МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

КУШНІР ДМИТРО ОЛЕКСАНДРОВИЧ

УДК 004.932.72'1(043)

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОЗОБРАЖЕННЯХ НА МОБІЛЬНІЙ ПЛАТФОРМІ В РЕАЛЬНОМУ ЧАСІ

123 – «Комп'ютерна інженерія»

12 – «Інформаційні технології»

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.
Кушнір Д.О.

Науковий керівник:
Парамуд Ярослав Степанович
кандидат технічних наук

Львів – 2023

АНОТАЦІЯ

Кушнір Д.О. Методи та засоби пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі в реальному часі. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп’ютерна інженерія» (12 «Інформаційні технології»). – Національний університет «Львівська політехніка», Львів, 2023.

Зміст анотації

Дисертація присвячена вирішенню актуальної науково-технічної задачі розроблення методів та засобів пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі у реальному часі.

У вступі обґрунтовано актуальність теми дисертаційних досліджень, сформульовано мету дослідження та науково-технічні завдання, необхідні для її досягнення, показано зв’язок дослідження з науковими програмами та темами, наведено наукову новизну отриманих результатів, їх практичну цінність та особистий внесок здобувача. Подано відомості про апробацію результатів роботи та особистий внесок автора та його публікації.

У першому розділі проведено аналіз існуючих підходів до інтеграції систем пошуку та розпізнавання об’єктів, а саме різновиди та архітектурні особливості моделей розпізнавання, та алгоритмів відстежування довільного класу об’єктів. Результати аналізу показали що інтеграція таких систем вимагає застосуванню певного набору фільтрів, спеціалізованих функцій активації, та алгоритмів відстеження об’єктів. У ході аналізу, як базову нейронну мережу обрано сімейство моделей згорткових нейронних мереж Yolo, як найбільш перспективну у галузі розпізнавання об’єктів. Додатково проведено аналіз існуючих мобільних систем для пошуку та розпізнавання об’єктів у реальному часі. Визначено, що великою проблематикою таких систем є відсутність ефективної платформи автоматичного тренування та

інтеграції моделей у мобільну платформу. Також однією з проблем є підвищення ефективності роботи таких систем, оскільки вони переважно мають обмежені апаратні можливості. Як висновок до першого розділу, сформовано набір методів та засобів для вирішення проблеми пошуку та розпізнавання у відеозображеннях на мобільній платформі у реальному часі та сформульовано завдання дисертаційного дослідження.

У другому розділі запропоновано метрики оцінювання результатів розпізнавання та відстеження об’єктів. Сформовано та описано загальну структуру моделі згорткової нейронної мережі Yolov4 для мобільної платформи. Використано модифікований метод кластеризації об’єктів розпізнавання на базі k-середніх++ для формування якорів розпізнавання. Розроблено методи фільтрації результатів розпізнавання. Розроблено 3 алгоритма відстеження об’єктів: алгоритмічний, алгоритмічний з навчанням з підкріпленням та алгоритм оперативного відстеження на базі мінімізаційного фільтру IOU, з використанням Угорського алгоритму як функції збіжності. Розроблено методи мемоїзації об’єктів відстеження. Запропоновано метод квантизації вихідних вагових коефіцієнтів згорткової нейронної мережі методом афінних перетворень.

У третьому розділі, згідно з запропонованими методами та засобами, розроблено алгоритми тренування моделі згорткової нейронної мережі, автоматичного анотування вхідних зображень та конвертування моделі у CoreML формат для мобільної платформи. Згідно обраних засобів масштабування та контейнеризації Docker, побудована структура системи автономого анотування, тренування та конвертації такої моделі. З даної структури можна виділити Docker контейнери для кожного модуля/сервіса, які використовують масштабовані апаратні можливості операційної системи. Описано взаємозалежності між кожним елементом такої системи. Запропоновано засіб інтеграції вбудованого модуля для відстеження рухомих об’єктів на мобільній платформі iOS. Інтеграція полягає у використанні бібліотеки JavaScriptCore для передачі даних між системою та модулем.

У четвертому розділі представлено розроблену архітектуру систем на мобільній операційній системі iOS та операційній системі Ubuntu та обґрунтовано вибір компонент таких систем. Представлено результати аналізу та апробації системи. Отримані результати дослідження підтвердили ефективність алгоритмів пошуку та розпізнавання у реальному часі.

Ключові слова: розпізнавання об’єктів, алгоритм відстеження об’єктів, фільтрація результатів розпізнавання, масштабоване середовище, функції активації, відеозображення, мобільна платформа, згорткова нейронна мережа, реальний масштаб часу, час пошуку об’єктів, час розпізнавання об’єктів, масштабована система Docker, сімейство моделей згорткових нейронних мереж Yolo, алгоритми кластеризації, Угорський алгоритм, афінні перетворення.

ABSTRACT

Kushnir D.O. Methods and means of searching and recognizing objects in video images on the mobile platform in real-time. – Qualification scientific work on the rights of a manuscript.

The thesis for the Philosophy Doctor (Ph.D.) degree in specialty 123 «Computer Engineering» (12 «Information Technology»). – Lviv Polytechnic National University, Lviv, Ukraine, 2023.

Abstract content

The Ph.D. thesis is devoted to solving the current scientific and technical problem of developing real-time methods to search and recognize objects in video images on a mobile platform.

The introduction substantiates the relevance of the topic of dissertation research, formulates the purpose of the study and the scientific and technical tasks necessary to achieve it, shows the connection of the study with scientific programs and topics, provides the scientific novelty of the results obtained, their practical value and the personal contribution of the applicant. Information about the work results' testing and the author's personal contribution and publication are presented.

The first section analyzes existing approaches to integrating search and object recognition systems, namely, varieties and architectural features of recognition models and algorithms for tracking an arbitrary class of objects. The analysis results showed that integrating such systems requires applying a particular set of filters, specialized activation functions, and object-tracking algorithms. During the analysis, the Yolo family of convolutional neural network models was chosen as the basic neural network, as the most promising in the field of object recognition. In addition, an analysis of existing mobile systems for searching and recognizing objects in real time was carried out. It was determined that a significant problem of such systems is the lack of an effective platform for automatic training and integrating models into the mobile platform. Also, one of the problems is increasing the efficiency of such systems since they mostly have limited hardware capabilities. As a conclusion to the first chapter, a

set of methods and tools for solving the problem of search and recognition in video images on a mobile platform in real time was formed, and the task of the dissertation research was formulated.

In the second section, metrics for evaluating the results of object recognition and tracking were proposed. The general structure of the Yolov4 convolutional neural network model for the mobile platform is formed and described. A modified method of recognition object clustering based on k-means++ was used to create recognition anchors. Methods of filtering recognition results have been developed. Three object tracking algorithms have been developed: algorithmic, algorithmic with reinforcement learning, and an operational tracking algorithm based on the IOU minimization filter, using the Hungarian algorithm as a convergence function. Methods of memoization of tracking objects have been developed. Finally, a method of quantizing the output weight coefficients of a convolutional neural network by affine transformations is proposed.

In the third chapter, according to the proposed methods and tools, algorithms for training the convolutional neural network model, automatic annotation of input images, and conversion of the model into CoreML format for the mobile platform are developed. According to the selected means of scaling and containerization of Docker, the structure of the system of autonomous annotation, training, and conversion of such a model was built. From this structure, Docker containers can be extracted for each module/service, which will offer scalable hardware capabilities of the operating system. The interdependence between each element of such a system is described. A means of integrating a built-in module for tracking moving objects on the iOS mobile platform is proposed. The integration takes place with the use of the JavaScriptCore library for data transfer between the system and the module..

The fourth chapter presents the developed system architecture of the iOS mobile operating system and the Ubuntu operating system and justifies the choice of components of such systems. The results of system analysis and testing are presented. The obtained research results confirmed the effectiveness of search and recognition algorithms in real time.

Keywords: object recognition, object tracking algorithm, results filtering, scalable environment, activation functions, video images, mobile platform, convolutional neural network, real-time map, object search time, object recognition time, scalable Docker system, Yolo cluster of convolutional neural network models.

Список публікацій здобувача за темою дисертації

Статті у фахових виданнях України:

- Kushnir, D., & Paramud, Y. (2020). The algorithm of Cyber-Physical system targeting on a movable object using the smart sensor unit. *Advances In Cyber-Physical Systems*, 5(1), 16-22. https://doi.org/10.23939/acps2020.01.016.
- Кушнір, Д. (2021). Методи та засоби покращення точності розпізнавання об'єктів на мобільній платформі iOS в реальному часі. *Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 3(1), 80-88. https://doi.org/10.23939/csn2021.01.080.
- Кушнір, Д., & Парамуд, Я. (2019). Методи пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі ios в реальному часі. *Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 1(1), 24-34. https://doi.org/10.23939/csn2019.01.024.
- Кушнір, Д., & Парамуд, Я. (2020). Алгоритм оперативного наведення засобів вимірюваьно – керувального вузла кіберфізичної системи на рухомий об'єкт. *Комп'ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 2(1), 44-52. https://doi.org/10.23939/csn2020.01.044.

Статті у наукових періодичних виданнях інших держав, які входять до міжнародних наукометричних баз:

- Kushnir, D. (2022) Methods and means for small dynamic objects recognition and tracking. *Computers, Materials & Continua*, 73(2), 3649-3655. https://doi.org/10.32604/cmc.2022.030016.

Матеріали міжнародних наукових та науково-практичних конференцій, збірники яких входять до міжнародних наукометричних баз:

- Kushnir, D., & Paramud, Y. (2020). Model for real-time object searching and recognizing on mobile Platform. 2020 IEEE 15Th International Conference On Advanced Trends In Radioelectronics, Telecommunications And Computer

Engineering (TCSET). *IEEE*, 127-130. https://doi.org/10.1109/tcset49122.2020.235407.

- Kushnir, D., Ocherklevich, O., & Paramud, Y. (2021). Deep Neural Network Model for Text Semantic Analysis Based on Word Embeddings. *2021 11Th International Conference On Advanced Computer Information Technologies (ACIT)*. *IEEE*, 718-721. https://doi.org/10.1109/acit52158.2021.9548393.

Матеріали міжнародних наукових та науково-практичних конференцій:

- Vavruk, E., & Kushnir, D. (2018). Mobile system for text recognition and translation with using Microsoft Cognitive API. *8th International youth science forum «Litteris et Artibus»*, 81-84. https://ena.lpnu.ua/handle/ntb/51711.

Наукові праці, які додатково відображають наукові результати дисертації:

- Ваврук, Є., & Кушнір, Д. (2018). Система розпізнавання та перекладу текстової інформації в мобільних додатках з використанням бібліотеки Microsoft Cognitive OCR. *Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні системи та мережі*, 1(905), 33-42. https://doi.org/10.23939/csn2018.905.033.
- Andrushchak, N., Vynnyk, D., Melnyk, M., Bajurko, P., Kushnir, D., Haiduchok, V., ... & Yashchyshyn, Y. Impact of optical Illumination on transmission of subterahertz electromagnetic waves by Bi 12 GeO 20 crystals. *Acta Physica Polonica A*, 4(141), 415-419. https://doi.org/10.12693/APhysPolA.141.415
- Borak, T., Kushnir, D., & Paramud, Y. (2022). Microprocessor Subsystem of the Smart House to Control the Multichannel Irrigation of the Room Plants. *Advances In Cyber-Physical Systems*, 7(1), 1-7. https://doi.org/10.23939/acps2022.01.001.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. ШНМ – штучна нейронна мережа;
2. ШІ – штучний інтелект;
3. ЗНМ – згорткова нейронна мережа;
4. РНМ – рекурентна нейронна мережа;
5. МН – машинне навчання;
6. ГН – глибинне навчання;
7. КЗ – комп’ютерний зір;
8. РО – розпізнавання об’єктів;
9. ШН – штучний нейрон;
10. ReLU – rectified linear unit;
11. Mish – a self-regularized Non-Monotonic Activation Function;
12. КА – карта активації;
13. k-means – кластеризація методом K-середніх;
14. TP – істинно позитивне стан;
15. FP – хибно позитивний стан;
16. FN – хибно негативний стан;
17. R – recall (чутливість);
18. P – precision (влучність);
19. F1 – F-міра;
20. Yolo – You Only Look Once;
21. R-CNN – Region-based Convolutional Neural Networks;
22. HOG – Histogram of Oriented Gradients;
23. SSD – Single Shot Detector;
24. mAP – значення середньої влучності;
25. CoreML – фреймворк для розгортання моделей ЗНМ на МП;
26. MOT17 – еталонна вибірка рухомих об’єктів для оцінки методів відстеження;
27. Backbone – хребет ШНМ;

28. Neck – шия ШНМ;
29. Head – голова ШНМ;
30. ID – кількість перемикань об’єктів.
31. СП – спеціалізована платформа;
32. МП – мобільна платформа;
33. МОС – мобільна операційна система;
34. AWS – Amazon Web Services;
35. PAAS – платформа як сервіс;
36. DDPG – Deep Deterministic Policy Gradient;
37. КФС – кіберфізична система;
38. ВКВ – вимірювально-керувальний вузол;
39. ВО – відстеження об’єктів;
40. CSP – Cross Stage Partial;
41. IOU – мінімізаційний фільтр Intersection Over Union;
42. ЗФС – згладжуючий фільтр стиснення;
43. ВМ – вбудований модуль;
44. UMD – Universal Module Definition;
45. RETR – Recognizer and Tracker.

ВСТУП

Обґрунтування вибору теми дисертації. Широке використання сучасних мобільних засобів, збільшення їх функціональних можливостей забезпечує розв'язання різного класу задач. Однією з таких задач є пошук та розпізнавання об'єктів у реальному часі. При цьому розвиток технологій та апаратних можливостей дозволив підняти якість пошуку та розпізнавання на новий рівень. Сучасні кіберфізичні системи (КФС) здатні ефективно розпізнавати та відстежувати об'єкти за допомогою різного набору методів та засобів. Водночас існуючі системи не завжди правильно відображають результати розпізнавання, а методи подальшого відстеження певного класу об'єктів мають значні похибки в результатах оцінки траєкторії руху таких об'єктів. Суттєвою специфікою є інтеграція таких методів та засобів на мобільний пристрій, який має набагато менше апаратних можливостей у порівнянні зі статичними КФС. При цьому важливо зберегти показники точності розпізнавання. Відповідно актуальною є наукова задача підвищення точності пошуку та розпізнавання певного класу об'єктів на мобільній платформі шляхом удосконалення та розроблення методів, засобів, моделей та алгоритмів ефективного розпізнавання та відстеження у реальному часі.

Тому актуальною є наукова задача підвищення точності пошуку та розпізнавання довільного класу об'єктів на мобільній платформі шляхом удосконалення відповідних моделей нейронних мереж та розроблення методів, засобів та алгоритмів ефективного розпізнавання та відстеження у реальному часі.

Зв'язок роботи з науковими програмами, планами, темами. Основні дослідження виконані на кафедрі електронних обчислювальних машин Національного університету "Львівська політехніка" та відповідають науковій тематиці кафедри: "Питання теорії, проектування та реалізації комп'ютерних систем та мереж, а також комп'ютерних засобів, вузлів, приладів і пристроїв вимірювальних, інформаційних, керуючих, телекомунікаційних та

кіберфізичних систем”.

Частина дисертаційних досліджень виконана в межах держбюджетної науково-дослідної роботи:

- «Інноваційне використання твердотілих і нанокомпозитних матеріалів для керування субтерагерцовим випромінюванням» (номер державної реєстрації 0119U100609; терміни виконання роботи: 01.2021 – 12.2021 р.).

Мета і задачі дослідження. Метою дисертаційної роботи є розробка та дослідження методів та засобів пошуку та розпізнавання об’єктів на мобільній платформі у реальному часі.

Для досягнення поставленої мети **потрібно вирішити такі основні завдання**:

- проаналізувати відомі методи, засоби, моделі та алгоритми пошуку та розпізнавання об’єктів;
- проаналізувати існуючі моделі пошуку та розпізнавання об’єктів на мобільній платформі;
- запропонувати модель для розпізнавання довільного класу об’єктів;
- запропоновувати методи кластеризації, квантизації, фільтрації та мемоїзації результатів розпізнавання;
- реалізувати алгоритми оперативного відстеження розпізнаних об’єктів;
- розробити засоби контейнеризації та масштабування елементів системи;
- розробити масштабовану систему пошуку та розпізнавання об’єктів;
- розробити автоматизований засіб конвертування розробленої моделі ЗНМ у CoreML формат для мобільних пристроїв;
- практично реалізувати та верифікувати систему пошуку та розпізнавання об’єктів на мобільній платформі у реальному часі.

Об’єктом досліджень є процес пошуку та розпізнавання об’єктів в відеозображеннях на мобільній платформі у реальному часі.

Предметом досліджень є методи, засоби, моделі та алгоритми пошуку та розпізнавання об’єктів в відеозображеннях на мобільній платформі у реальному часі.

Методи досліджень. Методи досліджень базуються на основних методах комп'ютерної інженерії та системного аналізу: *побудові моделі системи* пошуку і розпізнавання об'єктів на мобільній платформі методами математичного та комп'ютерного моделювання, та *аналізу цієї системи* на основі побудованої моделі методами аналізу, декомпозиції, синтезу та концептуального моделювання.

На етапі аналізу системи пошуку та розпізнавання об'єктів було визначено *її архітектуру, властивості, переваги та недоліки;* проаналізовано сучасні підходи до пошуку, розпізнавання та відстеження об'єктів. Визначено методи навчання та кластеризації вагових коефіцієнтів нейронних мереж. Проаналізовано сімейство моделей Yolo, та спроектовано архітектуру оригінальної моделі ЗНМ на базі моделі Yolov4. Були проаналізовані методи масштабування системи пошуку та розпізнавання, у тому числі за допомогою PAAS платформи Docker. Проаналізовано існуючі підходи до створення алгоритмів відстеження об'єктів у реальному часі. Проаналізовано методики інтеграції спроектованої моделі на мобільний пристрій iOS за допомогою фреймворку CoreML.

На етапі декомпозиції визначено завдання розробки *моделі розпізнавання об'єктів ЗНМ* для мобільної платформи; методів кластеризації якорів розпізнавання моделі ЗНМ; *методів фільтрації* для вихідної моделі ЗНМ за допомогою згладжуючих та мінімізаційних фільтрів; *методу оперативного відстеження* розпізнаних об'єктів, який полягає у використанні алгоритму відповідності точок на базі Угорського алгоритму; *методу квантизації вихідних вагових коефіцієнтів* ЗНМ на основні афінних перетворень; *алгоритму мемоїзації* відстежених об'єктів який базується на методах оперативного фільтрування об'єктів за граничною лінією; засобів масштабованого анотування вхідних зображень, тренування моделі ЗНМ та конвертування у формат CoreML; підхід до інтеграції вбудованого модуля відстеження об'єктів у мобільний додаток за допомогою фреймворку JavascriptCore.

На етапі синтезу були *створені двошарова та тришарова модель* ЗНМ

yolo_getr з змінною кількістю вхідних класів, зображень, ступеню квантизації та методом кластеризації якорів розпізнавання k-means++; створені *засоби контейнеризації та масштабування елементів системи* для масштабованого анотування вхідних зображень, тренування моделі ЗНМ та конвертування у Coreml фомат; створені засоби інтеграції вбудованого модуля відстеження на МП.

На етапі концептуального моделювання *були реалізовані* системи пошуку та розпізнавання на вбудованій мобільній системі Jetson Nano та мобільному пристрої на базі МОС iOS; були *впровадженні* розроблені методи на мовах JavaScript, Python та Swift; *проведено тренування* нейронних мереж типу yolo_getr та *тестування* роботи методів та засобів розпізнавання та відстеження у реальному часі по метриках ефективності та продуктивності на МП.

Наукова новизна одержаних результатів. Внаслідок теоретико-практичного підходу при розробленні системи пошуку та розпізнавання об’єктів на мобільній платформі у реальному часі:

Удосконалено метод кластеризації при інтеграції у модель згорткової нейронної мережі на мобільній платформі, який полягає у використанні метода к-середніх++ разом з мінімізаційними та згладжуючими фільтрами, який, у порівнянні з методом к-середніх, поліпшує точність розпізнавання у середньому на 5% за метриками mAP, F1 та Recall.

Отримав подальший розвиток метод квантизації моделі згорткової нейронної мережі на основі афінних перетворень для мобільної платформи, який дозволив збільшити продуктивність моделі у середньому на 7% за метриками реального часу у порівнянні з не квантизованими моделям при зменшенні показника метрики mAP до 4%.

Удосконалено метод збіжності для задач оперативного відстеження завчасно визначеного класу об’єктів, використовуючи Угорський алгоритм, що на відміну від існуючих аналогів, які використовують алгоритм K-вимірне дерево, поліпшує точність розпізнавання та відстеження рухомих об’єктів у середньому на 10% за метриками F1, MOTP та MT, що підтверджено еталонною

вибіркою MOT17.

Набув подальшого розвитку метод мемоїзації відстежених об’єктів з граничною лінією на мобільній платформі, який дозволяє збільшити точність відстеження рухомих об’єктів у реальному часі, що дозволило зменшити кількість перемикань ID на не менше ніж 6% у порівнянні із базовим методом відстеження без використання мемоїзації.

Вперше розроблено модель масштабованого створення згорткової нейронної мережі для мобільної платформи, яка полягає в автоматизованому анотуванні вхідного класу зображень, тренуванні моделі згорткової нейронної мережі та конвертації у CoreML формат для мобільної операційної системи iOS.

Практичне значення одержаних результатів. Практичне значення дисертаційного дослідження полягає у розробленні методів та засобів пошуку та розпізнавання об’єктів на мобільній платформі у реальному часі у вигляді масштабованого програмно-алгоритмічного забезпечення, яке може бути застосовано для ідентифікації, відстежування та рахування довільного класу об’єктів на мобільній КФС. У зв’язку з можливістю використовувати розпізнавання власних класів об’єктів, спектр використання системи може бути широкий: від систем розпізнавання руху малих рухомих об’єктів до систем відстежування людини без/з маскою під час пандемії.

Один і запропонованих методів фільтрації результатів розпізнавання об’єктів на зображенні використано у навчальному процесі кафедри «Електронних обчислювальних машин» Національного університету “Львівська політехніка” при викладанні дисципліни «Цифрова обробка сигналів» за темою «Опрацювання цифрових зображень» для студентів освітньо-кваліфікаційного рівня “бакалавр”, що навчаються за напрямком 123 “Комп’ютерна інженерія”, спеціалізацією 123.04 «Кіберфізичні системи».

Частина дисертаційних досліджень виконана в межах держбюджетної науково-дослідної роботи: «Інноваційне використання твердотілих і нанокомпозитних матеріалів для керування субтерагерцовим випромінюванням» (номер державної реєстрації 0119U100609; терміни виконання роботи: 01.2021

– 12.2021 р.).

Особистий внесок здобувача. Основний зміст роботи, всі теоретичні та практичні результати, висновки і дослідження, які представлено до захисту, одержані автором особисто. Особисто здобувачеві належать наступні наукові результати: метод кластеризації якорів розпізнавання з використанням мінімізаційних та згладжуючих фільтрів [1 – 3, 5], квантизація моделі ЗНМ методом афінних перетворень для мобільного пристрою [1, 4], алгоритм оперативного відстеження довільного класу об’єктів з методом збіжності на основі Угорського алгоритму [5], алгоритмічний метод пошуку та наведення на об’єкт [2], Автоматизований пошук та наведення на об’єкт методом навчання з підкріпленням [3], методи мемоїзації відстежених об’єктів [5], масштабована система пошуку та розпізнавання об’єктів на мобільній платформі [2 – 3, 5], спроектована та розроблена модель ЗНМ yolo_getr [6]. Ідеї, положення чи гіпотези інших авторів, які наявні в дисертації, мають відповідні посилання і використані лише для підсилення ідей та результатів здобувача.

Апробація роботи. Основні теоретичні положення та практичні результати дисертаційної роботи доповідалися і обговорювались на семінарах та конференціях:

- наукових семінарах кафедри «Електронні обчислювальні машини» Національного університету «Львівська політехніка» (2019-2023);
- VIII Міжнародному форумі науки “Litteris et Artibus – 2018”, Львів, Україна, 2018;
- XV Міжнародній конференції Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET-2020), Slavske, Ukraine, 2020;
- XI Міжнародній конференції International Conference on Advanced Computer Information Technologies (ACIT-2021), Deggendorf, Germany, 2021.

Публікації. У 11 наукових публікаціях повністю відображені основні результати дисертації, з них: 1 стаття у науковому періодичному виданні інших

держав, що входить до наукометричних баз Scopus та Web of Science, 4 статті у фахових виданнях України, 2 матеріали міжнародних наукових та науково-практичних конференцій, які індексуються у базі Scopus, 1 матеріал міжнародної наукової та науково-практичної конференцій, 3 наукові праці, які додатково відображають наукові результати дисертації.

Структура та обсяг роботи. Повний обсяг дисертації становить 181 сторінку, з яких 143 сторінки основного тексту. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел (125 найменувань) та додатків. Робота містить 73 рисунка, 11 таблиць, 6 алгоритмів та 2 додатки. Структура, мова та стиль викладу дисертації відповідає вимогам МОН України до оформлення дисертації. Робота написана українською мовою з використанням сучасної наукової термінології, стиль викладу матеріалу послідовний та логічний.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

У розділі проаналізовано:

- Методи пошуку та розпізнавання об'єктів, у тому числі з використанням штучних нейроних мереж;
- Методи навчання та висновування у задачах розпізнавання об'єктів у тому числі для задач навчання з вчителем, без вчителя та з підкріпленням;
- Алгоритми розпізнавання за допомогою Згорткових Нейронних Мереж у тому числі алгоритми Yolo. Розглянуто також методи та засоби поліпшення результатів розпізнавання;
- Існуючі мобільні платформи та засоби для виконання задач розпізнавання об'єктів, у тому числі мобільну плафторму iOS та фреймворк CoreML;
- Засоби масштабування системи пошуку та розпізнавання, у тому числі PAAS Docker;
- Існуючі алгоритми відстеження та наведення на рухомі об'єкти, у тому числі з використанням згорткових нейронних мереж. Проаналізовано основні типи таких методів.

Результати розділу опубліковано в працях автора [1 – 11].

1.1. Пошук та розпізнавання об'єктів: основні поняття та характеристики.

1.1.1. Загальні поняття.

Широке використання сучасних пристроїв, збільшення їх функціональних та апаратних можливостей забезпечує розв'язання різних класів задач. Однією з таких задач є пошук, розпізнавання та відстеження довільного класу об'єктів. Такі задачі відносять до галузі знань «Комп'ютерний Зір», яка є однією з підгалузей «Машинного Навчання» [1,12] (Рис. 1.1).

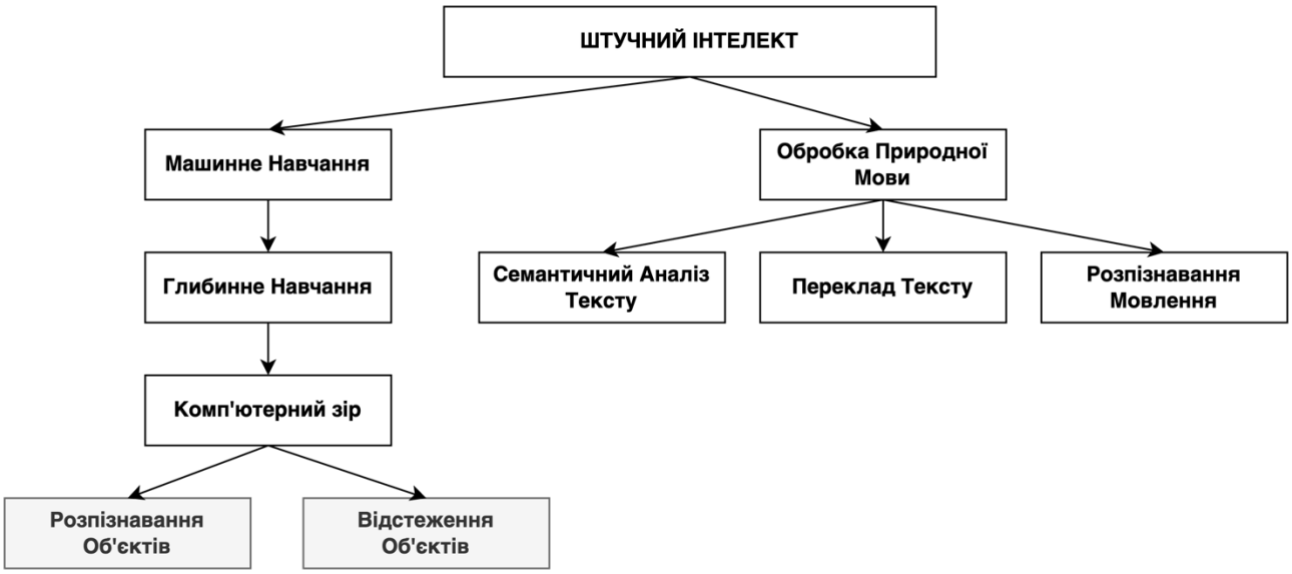


Рис. 1.1. Візуалізація зв’язків між галузями знань: Штучний Інтелект, Машинне Навчання, Глибинне Навчання та Комп’ютерний Зір.

Дамо загальні визначення кожному з понять:

- Штучний Інтелект (ШІ) – це такий інтелект, який демонструється машиною, на відміну від природного інтелекту, що притаманний людям та тваринам. Дослідження ШІ визначається як область дослідження інтелектуальних агентів, яка сприймає своє середовище і вживає заходів, що збільшують її шанси на досягнення поставленної мети [12];
- Машинне Навчання (МН) – це такий ШІ, який може автоматично адаптуватися з мінімальним втручанням людини;
- Глибинне Навчання (ГН) – це підмножина МН, яка використовує Штучні Нейронні Мережі (ШМН) для імітації процесу навчання людського мозку [13];
- Комп’ютерний Зір (КЗ) – галузь знань, яка визначає як ШНМ може класифікувати та розпізнати об’єкти з відеозображень чи фотографій.

Системи КЗ часто застосовуються для розпізнавання та відстеження певного класу об’єктів. Останні дослідження в цій області показали значний приріст ефективності таких систем [1 – 5].

1.1.2. Методи розпізнавання об’єктів.

Окремо варто виділити методи Розпізнавання Об’єктів (РО).

РО – це широкий термін для опису сукупності методологій, які передбачають ідентифікацію об'єктів у цифрових зображеннях (Рис. 1.2).

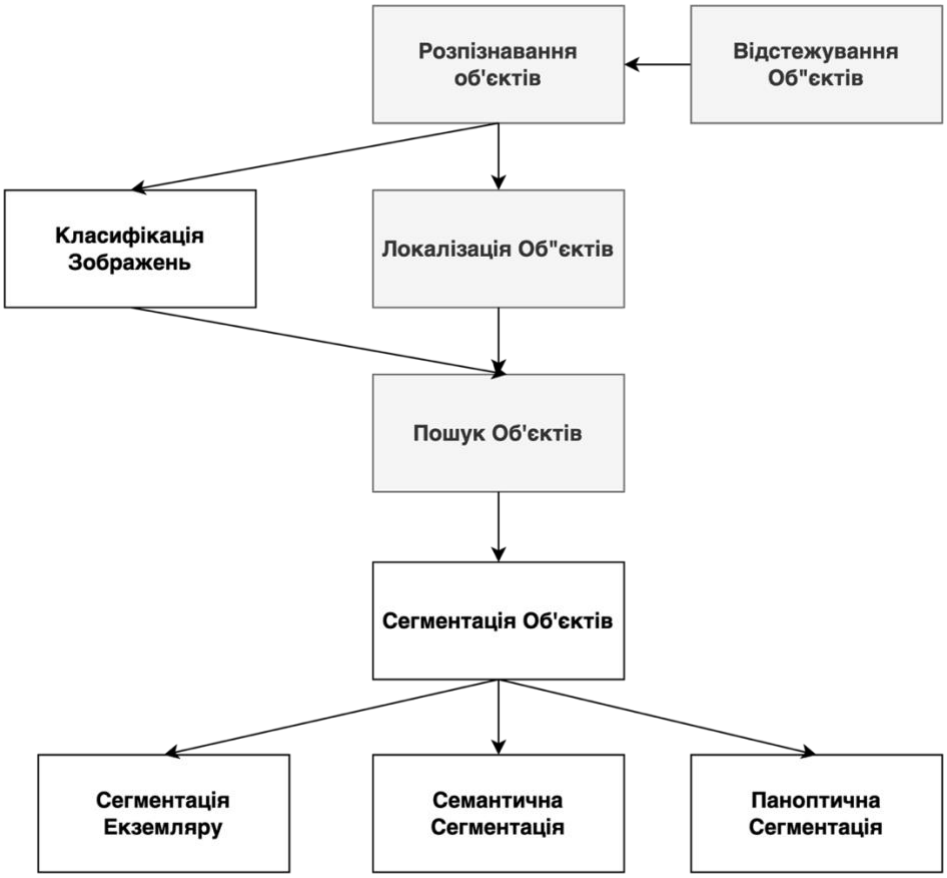


Рис. 1.2. Класифікація задач розпізнавання об’єктів.

Варто дати визначення що Клас – це множина об’єктів, які мають властивості «схожості». Для задачі РО обов’язково має бути визначений мінімум 1 клас.

РО можна розділити на дві окремі гілки: класифікацію зображень і локалізацію об'єктів.

- Класифікація зображень передбачає клас, заданий одним вхідним зображенням з одним об'єктом в межах зображення. Існують два типи класифікації: бінарна класифікація, та багато класова класифікація;
- Локалізація об'єкта - це методологія, яка використовується для визначення піксельної присутності об'єкта на зображенні та позначення місця розташування об'єкта за допомогою намальованої рамки розміру. Локалізація

об'єктів може бути додатково розкладена на пошук об'єктів, де об'єкти малюються рамками розміру та мітками, та сегментацію об'єктів, де зразки виділяються та відокремлюються від їх фону. Такі об’єкти також можна відносити до класів.

В рамках дисертаційного дослідження розглядаються задачі пошуку, локалізації, розпізнавання та відстеження об’єктів.

Існує велика кількість класифікаторів для виконання задач РО. Зазвичай, їх поділяють на три групи: порівняння за зразком, статистичні методи та нейронні мережі. Серед таких груп можна виділити наступні алгоритми: логістична регресія, відстань до найближчого сусіда (k-NN), найближче середнє значення (SVM), наївний Баєсівський класифікатор, дерево ухвалення рішень, Random Forest, екстремальне градієнте підсилення (XGBoost) та різні види ШНМ.

Ефективність згаданих алгоритмів була порівняна у досліджені [14]. Результати висвітлені и таблиці 1.1.

Таблиця 1.1. Порівняльний аналіз алгоритмів класифікації для задач РО.

Алгоритм / **Метрика**	**Логістична регресія**	**k-NN**	**SVM**	**Наївний Баєсівський Класифікатор**	**Дерево Ухвалення Рішень**	**Random Forest**	**XGBoost Класифікатор**	**ШНМ Тренована Вибірка**	**ШНМ Тестова Вибірка**
Точність (%)	83	85	83	80	80	83	85	89	83
Чутливість (%)	83	87	85	92	77	80	87	89	83
Влучність (%)	85	86	83	76	86	89	87	89	86
Специфічність (%)	80	83	83	87	74	78	84	88	79
***F*-міра (%)**	85	86	84	84	80	84	87	89	85

Метрика була згенерована за допомогою матриці невідповідностей (confusion matrix), та сформована на основі хибно позитивних (TP) та хибно негативних (TN) значень під час тестування. Розглянемо кожен з параметрів окремо.

- **точність (accuracy)** – частка правильно прогнозованих результатів з усієї вибірки (як негативні так і позитивні результати). Більш висока точність означає більш високий показник правильно класифікованих даних;
- **повнота або чутливість (recall)** – частка загального числа позитивних

зразків, з знайдених. Чим вище значення повноти, тим нижчий неправильно класифікований позитивний результат;

- **влучність або прогностична значущість позитивного результату (precision)** – показник правильно прогнозованих позитивних зразків серед усіх позитивних прогнозованих зразків. Вища точність означає більш високі правильні класифіковані дані для справжніх результатів;
- **специфічність (specificity)** – показник правильно прогнозованих негативних результатів серед усіх фактичних негативних зразків. Більш висока специфічність означає більш високі правильні класифіковані дані для негативних результатів;
- ***F-міра (F*$_1$ **score)** – є середнім гармонійним повноти та влучності. Це запобігає вибору неправильної моделі, якщо набір даних не розділений правильно [14].

Аналізуючи результати, можна побачити що алгоритми ШНМ для однакової тестової вибірки показують точніші результати по усім метрикам. Тому для подальших досліджень було обрано алгоритм ШНМ.

У той же час поява великомасштабних анотованих наборів даних зображень, таких як OpenImages, більш широка доступність обчислювальних засобів дозволила вивести якість розпізнавання за допомогою нейронних мереж на новий рівень.

1.1.3. Методи розпізнавання за допомогою нейронних мереж.

Одним з найефективніших засобів пошуку та розпізнавання об’єктів є ШНМ [15]. Під ШНМ (або Багатошаровим Перцептроном) розуміють системи які наслідуються від природного функціонування мозку тварин.

Традиційна архітектура ШНМ складається з шарів входу (дендрити), одного або декількох прихованих шарів, та вихідного шару (аксон). Кожен вузол (нейронне ядро) представлено як штучний нейрон (ШН), а з’єднання між ними (синапси) об’єднують входи та виходи суміжних нейронів (Рис 1.3).

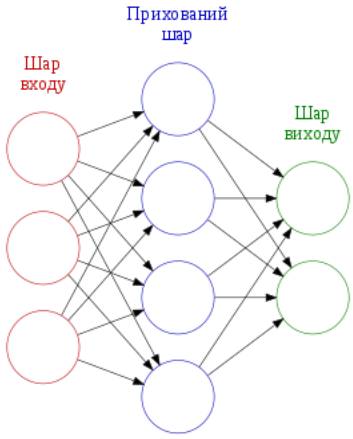


Рис. 1.3. Узагальнена модель ШНМ.

Варто визначити характеристики основного модуля обробки ШНМ – штучний нейрон (ШН). Його математична модель зображена на Рис. 1.4 [15].

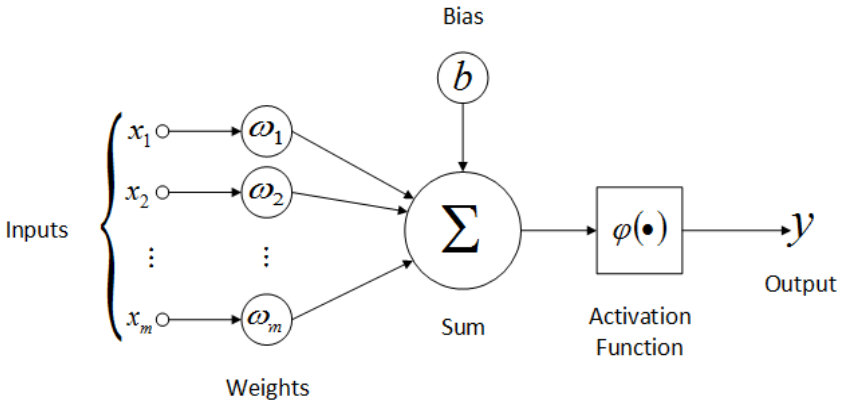


Рис. 1.4. Узагальнена математична модель ШН.

ШН використовує наступні математичні операції для входів x_i щоби подати результати на вихід y [15].

$$y = \varphi\left(\sum_i (\omega_i \times x_i) + b\right)$$

Де:

ω_i – вагові коефіцієнти (weights), які визначають силу зв'язку між нейронами;

x_i, y – входи та вихід ШН відповідно.

$\sum_i(\omega_i \times x_i)$ – перемноження вагових коефіцієнтів з вхідними сигналами. Зважена сума передається функції активації з урахуванням порогових значень;

b – порогове значення (bias), визначення порогових границь для функції активації;

φ – обрана функція активації ШН. Визначає значення вихідного сигналу, що передається наступним ШН. Головна особливість функцій активації – це їх монотонність.

Функції активації можна застосовувати різні, в залежності від поставлених задач:

- Sigmoid – це нелінійна, не бінарна, степенева функція. На відміну від лінійних функцій, функція сигмоїди намагається привести значення змінною до однієї з її сторін (або в від'ємній площині, або у додатній), що критично важливо в задачах класифікації зображень. Водночас, ця перевага є і недоліком: при наближені значення до кінця сигмоїди, градієнт змін приймає дуже малий діапазон можливих значень. Це призводить до проблем з градієнтом зникнення [4, 16];
- leaky ReLU – нелінійна функція активації. Повертає результат, при умові що вхідне значення є додатнім. В цьому випадку така функція працює як ефективне вирішення задачі апроксимації, що допомагає при необхідності об'єднати декілька шарів НМ [4, 17];
- Mish – функція яка є модифікацією функції активації Relu. При $x > 0$ є аналогічною Relu, при $x < 0$ є немонотонною функцією. Слугує вирішенням відомої проблеми «помираючого Relu» [4, 18];
- Softmax – функція, яка використовується до вихідного шару НМ у задачах класифікації зображень. Така функція може гарантувати, що сума всіх вихідних нейронів Нм рівна 1, а значення інтервалу рівне $[0, 1]$. Використовується на одному з рівнів функції активації Mish [4, 19].

Для виконання задачі дисертаційних досліджень на мобільній платформі, була обрана функція активації Mish, через її переваги при класифікації об'єктів на зображенні у ральному часі [4, 20].

Варто відзначити, що вхідний та вихідний шари ШН приймають лише одновимірні вектори. Таким чином, для обробки комплексних зображень, потрібно перевести їх з двох-вимірного простору у одновимірний. Цей процес називається згортанням (flattening) (Рис. 1.5) [21].

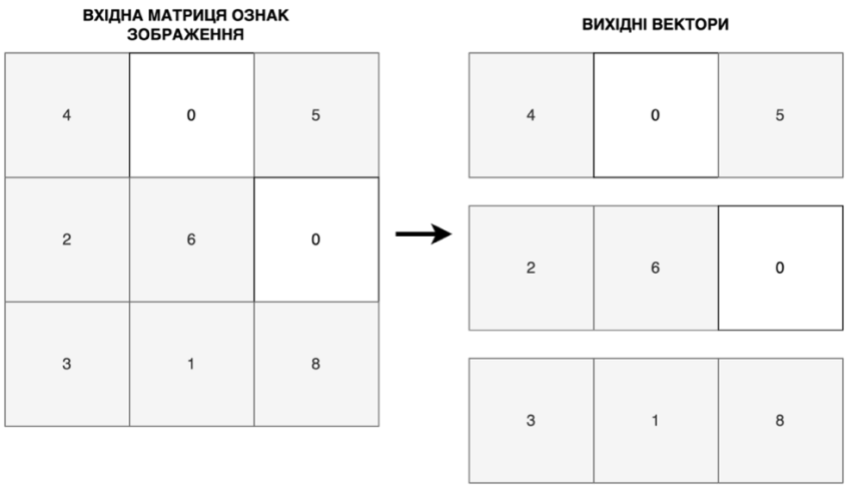


Рис. 1.5. Процес згортання ознак зображення у одновимірний вектор.

Результуючі вектори надходять на входи x_i штучного нейрона.

1.1.4. Методи навчання нейронних мереж та їх типи.

Для виконання поставлених задач пошуку та розпізнавання об’єктів, нейронна мережа має бути навчена.

Алгоритм навчання (тренування) – це метод або математична модель, при якому параметри ШНМ налаштовуються шляхом симуляції вхідного середовища. Це робиться шляхом поновлення ваг і рівнів упередженості мережі. З алгоритмічної точки зору, під тренуванням НМ розуміють вибирання однієї моделі з множини дозволених. Серед найвідоміших алгоритмів навчання варто виділити: градієнтний спуск (gradient descent) та його різновиди, сімейство алгоритмів Бройдена-Флетчера-Гольдрфабра-Шанно, алгоритм Левенберга-Марквардта та інші. Також існують оптимізації алгоритму градієнтного спуску, такі як Adaptive Movement Estimation (Adam)Adaptive Subgradient Methods for Online Learning and Stochastic Optimization (Adgard) та Nesterov Momentum [22-23].

Наступним етапом після навчання йде задача «висновування» (inference).

Висновування – алгоритм, який полягає у використанні навченої моделі для отримання висновків/прогнозів на тестовій вибірці. Ця фаза вимагає значно менше обчислювальних ресурсів та може виконуватись на графічному процесорі (GPU), через відсутність потреби у прорахуванні похибки. Зазвичай алгоритм висновування поділяють на три фази: зіставлення, вибірка та виконання.

Тип навчання залежить від вхідних та вихідних ознак. На сьогоднішній час, розрізняють три основні типи навчання: з підкріпленням, з наставником та без наставника (Рис. 1.6) [24].

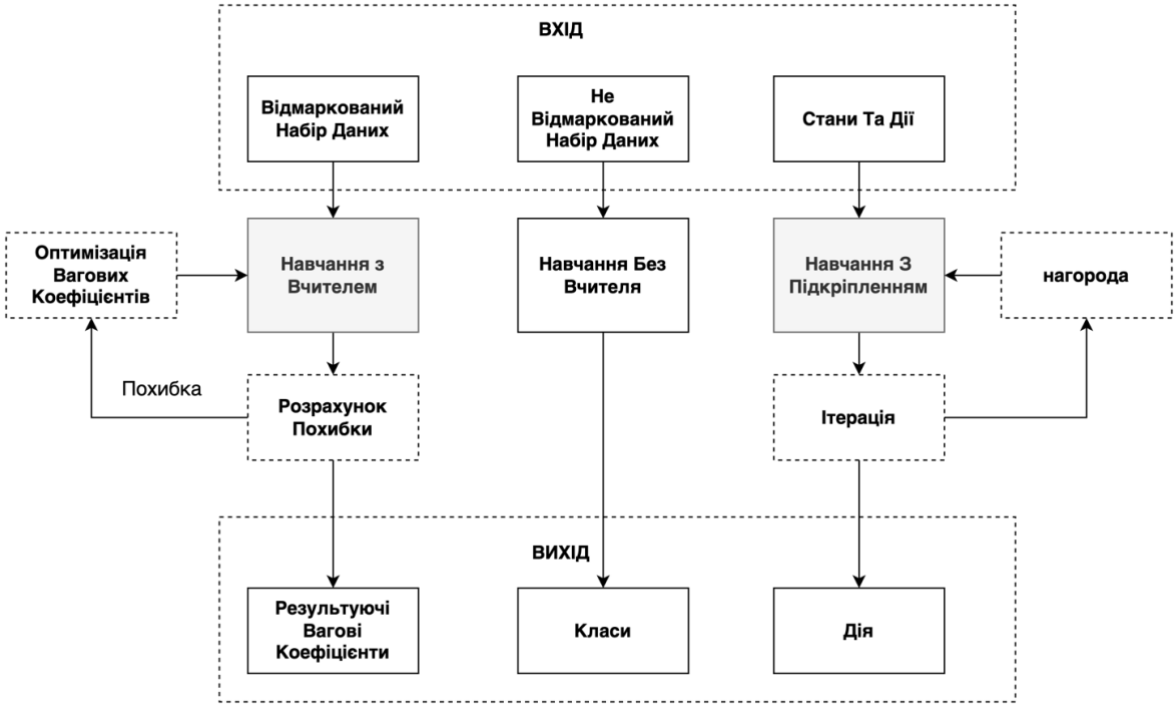


Рис. 1.6. Типи навчань нейронної мережі.

- Навчання з вчителем (supervised learning) – процес подачі на вхід завчасно відмаркованого набору даних. Кожен екземпляр подається на вхід ШНМ, проходить обробку та порівнюється з значенням цільового вектора, що являє собою потрібний вихід мережі. Далі за визначеним правилом вираховується помилка, що тягне за собою зміну вагових коефіцієнтів в середині ШНМ. При чому правила зміни визначаються обраним алгоритмом. Вектори навчальної вибірки подаються на вхід послідовно, обчислюються помилки, ваги оптимізуються для кожного вектору, поки помилка для всієї навчальної вибірки не досягне прийнятого низького рівня [24]. Також розділяють підгалузь навчання з учителем: часткове навчання з вчителем (semi-supervised learning), при якому для навчання враховуються як відмарковані дані, так і не відмарковані. При цьому використовується мала відмаркована вибірка для тренування моделі, яка буде використовуватись для визначення гіпотез маркуваня решти не відмаркованих зображень. Такий підхід дозволяє

підвищити точність пошуку та розпізнавання об’єктів, при умові що вхідна вибірка є не великою, та клас об’єктів є максимально суміжний [25-26];

- Навчання без вчителя (unsupervised learning) – процес подачі на вхід не відмаркованого набору даних. Алгоритм опрацьовує вагові коефіцієнти НМ таким чином, щоб у результаті отримувались вихідні вектори які є узгодженними. Таким чином, при отриманні близьких по значенню вхідних векторів, будуть формуватися приблизно однакові виходи. Такий тип навчання зазвичай підходить для задач в яких заздалегідь відомомою є навчальна вибірка, а на виході очікується виявлення відповідних закономірностей, взаємозв’язків що можна виявити між об’єктами [24]. Основні задачі, які покладаються на навчання без вчителя це: задачі кластеризації, які можуть грати допоміжну роль при розв’язанні задач РО, задачі узагальнення, задачі візуалізації даних. Кластеризація – це процес розподілу елементів даних на класи таким чином, що елементи одного класу є максимально близькими, а елементи різних класів є максимально різнорідними. Приклади метрик кластеризації: зв’язок, відстань та інтенсивність. Одними з прикладів алгоритмів кластеризації є кластеризація методом k-середніх (k-means) та нечітка кластеризація, при чому алгоритм k-means вважається швидшим [27];
- Навчання з підкріпленням (reinforcement learning) – це проміжний варіант між парадигмою навчання з вчителем та навчання без вчителя. На заміну блоку «вчителя» додається блок «критик», який аналізує реакцію середовища на вхідні дані та визначає евристичну похибку, яка додається до процесу навчання НМ [24].

Також для навчання НМ часто використовують методику передавального навчання (Transfer Learning), яка полягає у використанні вже отриманих знань при тренуванні суміжної задачі. Наприклад, знання отриманні при розпізнаванні 60 видів комах різних груп, можна використати при спробі розпізнати певний вид мурах. Зазвичай при такій методиці глибинні шари ШНМ (backbone) «заморожуються», в той час як верхні шари (head) активно беруть участь у

навчанні.

Якщо тренувальна вибірка відрізняється по типу від вже отриманих класів, використовується методика доопрацювання (Fine-Tuning) передавального навчання. Вона полягає як і у замороженні глибинних шарів ШНМ, так і оптимізації вхідних параметрів, таких як: темп навчання (learning rate), цикл (epoch), оптимізатор, тощо. Як результат, збільшується швидкість навчання, та усуваються проблеми тренування малої вибірки [28].

У ході дисертаційного дослідження використано усі визначені парадигми навчання НМ.

Навчання з вчителем широко застосовується при виконанні задач пошуку та розпізнавання об’єктів на зображеннях, оскільки застосовується велика маркована вибірка та повністю з’єднані шари ШНМ. Основні сучасні типи нейронних мереж, які застосовуються для задачі пошуку та розпізнавання об’єктів наведені на Рис. 1.7.

У дослідженні розглядаються лише методи ГН, оскільки такі методи показали себе ефективнішими при розпізнаванні об’єктів на зображенні у реальному часі в порівнянні з такими методами МН, як методом Опорних Векторів [29].

Рис. 1.7. Типи глибинних нейронних мереж для задач розпізнавання об’єктів.

Для виконання цілей дослідження, тип нейронної мережі варто вибирати в залежності від продуктивності, швидкодії та розміру вагових коефіцієнтів. В таблиці 1.2. продемонстровано основні відмінності між Багатошаровим Перцептроном (ШНМ), ЗНМ та Рекурентною Нейронною Мережею (РНМ) [30].

Таблиця 1.2. Порівняльний аналіз ШНМ, ЗНМ та РНМ.

Критерій	**ШНМ**	**ЗНМ**	**РНМ**
Використання	Задачі предиктивної аналітики	Задачі КЗ, у тому числі пошук та розпізнавання об'єктів	Задачі обробки природньої мови (NLP) [7]
Тип вхідних даних	Табличні дані	Набір даних (зображення)	Послідовний набір даних
Переваги	Висока відмовостійкість	Висока продуктивність, малі за розміром вагові коефіцієнти.	Можливість запам'ятовувати стани ШН, можливість використати багато входів та виходів ШН
Недоліки	Залежність від апаратного забезпечення, низька продуктивність	Великий набір тестових даних вимагається для тренування	Повільний і складний процес тренування

Для подальших досліджень обрано ЗНМ як нейронну мережу, яка використовується з високою ефективністю при вирішенні задача пошуку та розпізнавання об'єктів.

1.2. Алгоритми розпізнавання об'єктів за допомогою згорткових нейронних мереж.

Згорткові Нейронні Мережі (ЗНМ) – модель нейронної мережі, історично

наступна ітерація ШНМ, яка автоматизує ручну роботу виділяння ознак, натомість зосереджуючись на оптимізації архітектури нейронної мережі. Одними з видів оптимізації є зменшення мінімального обсягу попередньої обробки та відсутність додаткового зв'язку між нейронами сусідніх шарів. Такий підхід дозволяє ЗНМ виділяти окремі визначені ознаки розпізнанного об'єкта в залежності від ступеня важливості таких ознак (Рис. 1.8) [31].

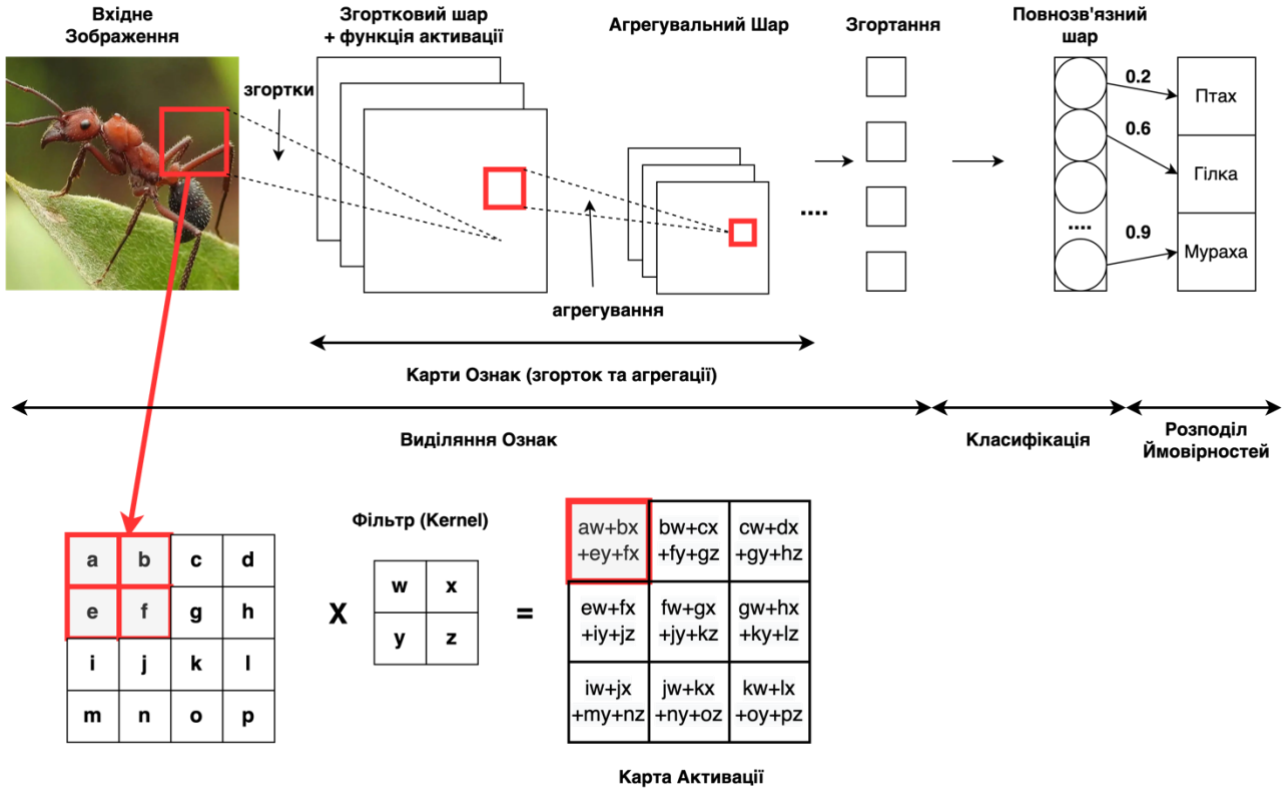


Рис. 1.8. Типова архітектура ЗНМ.

У більшості ЗНМ використовуються шари входу та виходу та три приховані шари. Розберемо приховані шари детальніше:

- Згортковий шар (Convolutional Layer). Перший прихований шар ЗНМ. Він складається з набору фільтрів (kernels) для кожного шару, функцію активації та параметрів, за якими буде відбуватися навчання. При чому розмір фільтрів є менший за розмір вхідного зображення. Кожен фільтр перемножується з фрагментом зображення та формує карту активації (КА). Цей процес повторюється для кожного фрагменту зображення. Результуюча згортка формується за рахунок комбінації згенерованих КА. Як результат, вихідні вагові коефіцієнти створюються шляхом об'єднання КА вздовж усієї

вимірювальної глибини. Можна вважати, що кожна компонента КА є вихідним результатом нейрона. Таким чином, кожен нейрон пов'язаний з невеликою локальною областю у вхідному зображені, а розмір рівний розміру фільтра. При чому усі нейрони у КА мають спільні параметри один з одним, що зумовлює ЗНМ вивчати фільтри, які мають максимальну реакцію на локальну область вхідних даних. В результаті, початкові згорткові шари фіксують низькорівневі характеристики (лінії) зображень, тоді як фінальні шари виділяють більш високорівневі ознаки (форми та об'єкти) [32];

- Агрегувальний шар (Pooling Layer). Наступний шар ЗНМ, який зазвичай йде після функції активації. Слугує для зменшення розмірів КА, зменшуючи кількість параметрів для навчання та загальний обсяг обчислень що виконуються в ЗНМ. Шар агрегації підсумовує ознаки виявленні в регіоні КА, створеної згортковим шаром. Отже наступні операції виконуються над узагальненими об'єктами, ніж над точно розташованими об'єктами, що робить модель більш стійкою у положені вхідних ознак на зображені. Розрізняють максимізаційне агрегування (max pooling) та усереднювальне агрегування (average pooling). У першому випадку виконується операція об'єднання, яка обирає максимальне значення елемента з КА, охопленої фільтром, генеруючи найімовірніші результати з попередньої КА. У другом випадку виконуються аналогічні операції, проте з об'єднанням середніх значень для КА [32];
- Повноз'єднаний шар (Fully Connected Layer). Запозичений від попередніх архітектур ШНМ (як видно з Рис. 1.8). З'єднує кожен ШН одного шару з ШН наступного шару. Також виконується операція згортання (Рис. 1.5).

Під час кожної кроку (stride), фільтр (kernel) проходить по висоті та ширині зображення, створюючи зображення цієї рецептивної області. Як результат формується КА, тобто двовимірне представлення зображення.

Формування вихідних ознак згорткового шару можна вирахувати за формулою [32]:

$$n_{out} = \left[\frac{n_{in} + 2p - k}{s}\right] + 1$$

Де:

n_{in} – кількість вхідних ознак; n_{out} – кількість вихідних ознак;

k – розмір згорткового фільтру;

p – розмір згорткових границь; (padding) s – розмір кроку

Як зазначалось, Методи класифікації в ЗНМ не відрізняються від згаданих методів ШНМ. Проте різні імплементації ЗНМ мають власні способи виділяння ознак. Розглянемо найпопулярніші архітектури ЗНМ.

Region-based Convolutional Neural Networks (R-CNN)

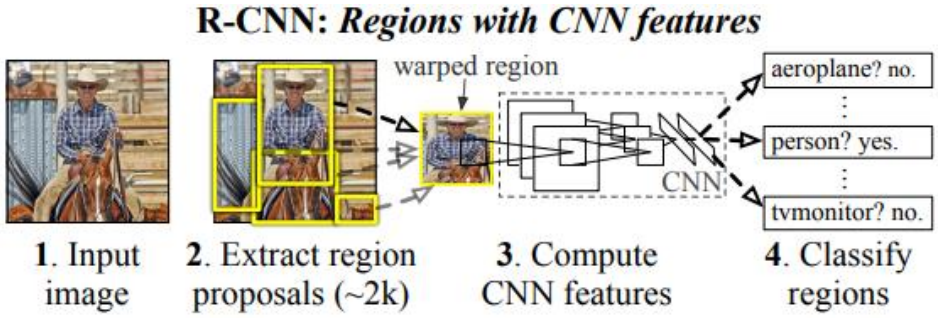


Рис. 1.9. Типова архітектура ЗНМ R-CNN.

R-CNN поєднує метод визначення можливих регіонів із ЗНМ. Для визначення регіонів зображення використовується алгоритм вибіркового пошуку який дозволяє генерувати 2000 різних регіонів які із високою ймовірністю можуть містити об’єкт. R-CNN допомагає локалізувати об’єкти використовуючи нейронну мережу глибинного навчання та навчання високоємної моделі з використанням досить малої кількості проанотованих даних. Цей метод має досить високу точність виявлення об’єктів завдяки використанню глибоких згорткових нейронних мереж для класифікації запропонованих регіонів. R-CNN дозволяє промасштабувати тисячі класів об’єктів не вдаючись до методів апроксимації, включаючи хешування [33].

Fast R-CNN

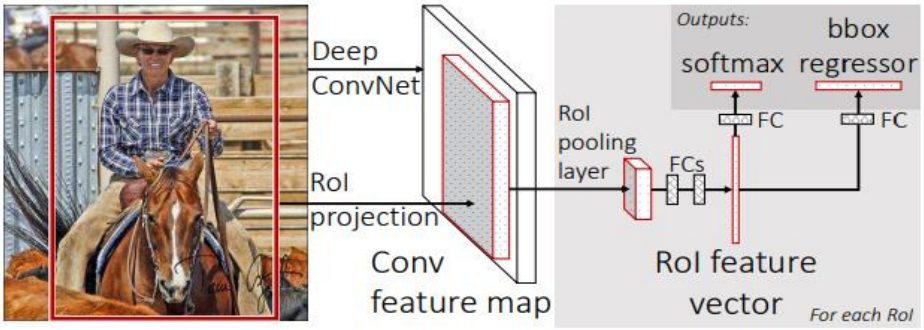


Рис. 1.10. Типова архітектура ЗНМ Fast-R-CNN.

Написаний мовою програмування Python та C++, Fast R-CNN це алгоритм МН для детектування об’єктів. Цей алгоритм виправляє недоліки R-CNN та SPPnet, а також покращує їх швидкість та точність [34].

Переваги Fast R-CNN:

- Вища якість детектування (mAP), ніж R-CNN та SPPnet;
- Навчання є одноступеневим, з використанням багатозадачної функції втрати;
- Навчання може оновити всі шари нейронної мережі;
- Не потрібно додаткової пам'яті для зберігання ознак зображення.

Faster R-CNN.

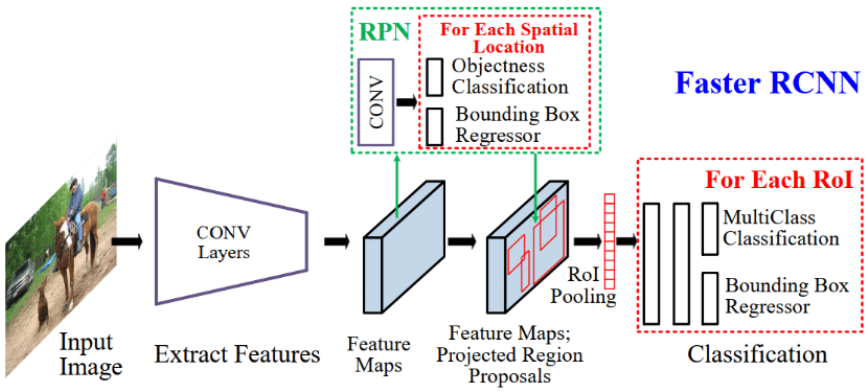


Рис. 1.11. Типова архітектура ЗНМ Faster-R-CNN.

Faster R-CNN – це схожий до Fast R-CNN алгоритм детектування об’єктів. Він складається із двох модулів: повної згорткової мережі що визначає можливі регіони об’єкта на зображенні та детектора Fast R-CNN який обробляє запропоновані регіони. У першому модулі використовується Region Proposal Network (RPN) – повна згорткова мережа, яка одночасно знаходить межі об’єкта

та ймовірність відповідності об’єкта певному класу. Ця мережа натренована генерувати запропоновані регіони із досить високою якістю [35].

Histogram of Oriented Gradients (HOG)

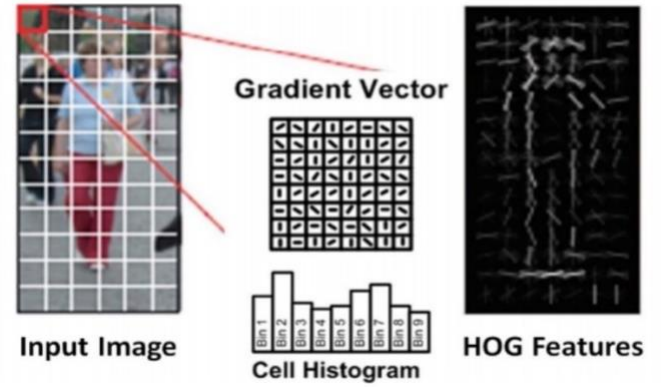


Рис. 1.12. Типова архітектура ЗНМ HOG.

HOG – це дескриптор ознак зображення, який використовується для виявлення об'єктів при обробці зображень та інших технологіях комп'ютерного зору. Метод HOG визначає напрямки градієнтів в локальних частинах зображення, таких як вікно детектування чи область інтересу. Однією з ключових переваг HOG-подібних ознак є їх простота і легкість розуміння інформації, яку вони несуть [36].

Region-based Fully Convolutional Network (R-FCN)

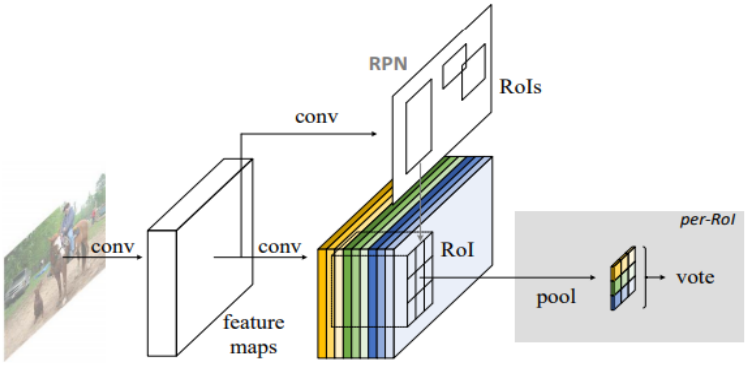


Рис. 1.13. Типова архітектура ЗНМ R-FCN.

R-FCN – це, як і Fast R-CNN чи Faster R-CNN, алгоритм детектування зображень що базується на регіонах. Але на відміну від Fast R-CNN та Faster R-CNN, які використовують ресурсоємні підмережі, R-FCN повністю згортковий і всі обчислення у цьому алгоритмі здійснюються для цілого зображення.

R-FCN складається із спільних, повністю згорткових архітектур, як це є

також у випадку з FCN, який, як відомо, дає кращий результат, ніж Faster R-CNN. У цьому алгоритмі всі шари нейромережі є згортковими і призначені для класифікації областей інтересу (ROI) на різні категорії об’єктів і фону [37].

Single Shot Detector (SSD)

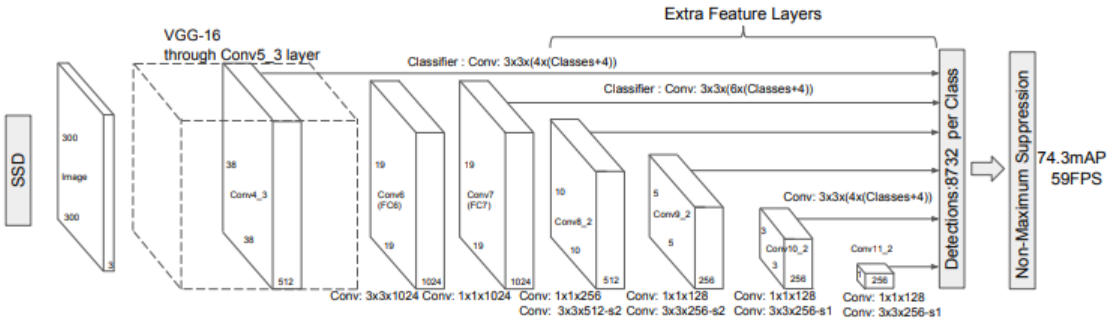


Рис. 1.14. Типова архітектура ЗНМ R-FCN.

SSD спроєктований для детектування об’єктів в режимі реального часу використовуючи одну ГН. У SSD вихідні області зображення дискретизуються у множину стандартних областей для різного співвідношення сторін. На етапі дискретизації відбувається масштабування відповідно до розташування КА. Мережа SSD об’єднує результати із різних карт ознак із різними розширеннями щоб обробляти об’єкти різного розміру [38].

Переваги SSD:

- повністю виключає етап генерації запропонованих областей об’єктів та подальші етапи передискретизації пікселів або ознак та інкапсулює всі обрахунки в одній нейромережі;
- легка в навчанні та проста для інтеграції в системи, які потребують модуля детектування;
- має приблизно таку ж точність детектування як і методи які використовують додатковий крок для генерації запропонованих об’єктів, при цьому цей метод є значно швидшим і пропонує єдиний підхід для навчання та інтерфейсу.

Spatial Pyramid Pooling (SPP-net)

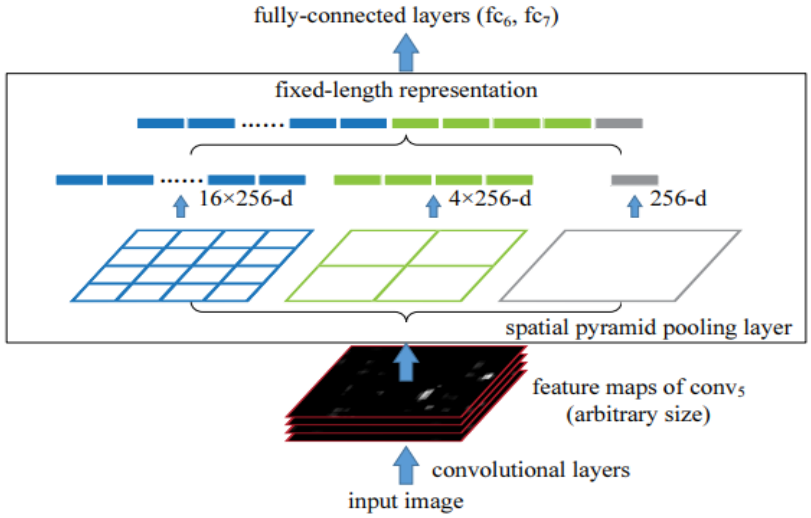


Рис. 1.15. Типова архітектура ЗНМ SPP-net.

SPP-net - це структура нейромереж яка може генерувати представлення фіксованої довжини, незалежно від розміру чи масштабу зображення. Метод SPP-net стійкий до деформації об’єктів і покращує попередні методи класифікації зображень на основі ЗНМ. Використовуючи SPP-net можна обрахувати карти ознак цілого зображення лише раз і після цього об’єднати ознаки у спільні регіони (частини зображення) що дозволяє генерувати представлення фіксованої довжини для тренування детекторів. Цей метод уникає повторення обрахунків згорткових ознак, що є його перевагою [39].

Yolo (You Only Look Once)

Yolo – це один з найпопулярніших алгоритмів детектування об’єктів що використовується науковцями всього світу. Стандартна модель Yolo виконує обробку зображення зі швидкістю до 35 FPS (кадрів у секунду) у реальному часі, тоді як версія з двома вихідними шарами – Tiny Yolo обробляє 125 кадрів в секунду, при цьому досягаючи подвоєного показника в порівнянні з іншими детекторами в реальному часі. Цей алгоритм перевершує інші методи детектування, включаючи DPM та R-CNN, як для зображень різних доменів (природа, мистецтво та ін.) [40].

Принцип роботи Yolo відрізняється від методів детектування об’єктів описаних вище. У методі Yolo єдина згорткова мережа визначає межі об’єктів і ймовірності належності їх до певних класів. Зображення ділиться на сітку

розмірністю SxS, в межах кожної сітки отримується певна кількість областей. Для кожної області нейромережа визначає ймовірність знаходження там об’єкту певного класу та значення зсуву області. Область яка містить об’єкт із ймовірністю вищою за задане значення використовується щоб визначити розміщення об’єкта на зображенні. Як і алгоритм SSD, Yolo позиціонується як ЗНМ, яка виконує фази виставлення ймовірностей регіонам розпізнавання та ймвовірностей класів за одну ітерацію (на відміну від моделей R-CNN) (Рис 1.16) [41].

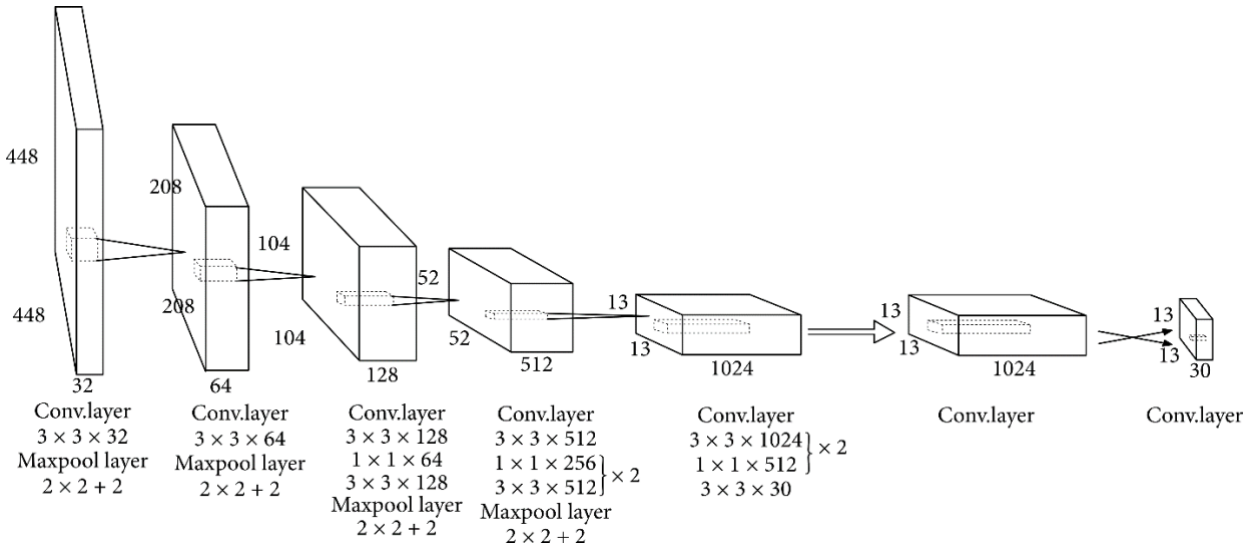


Рис. 1.16. Типова архітектура ЗНМ YOLO.

Для виконання дисертаційних досліджень, був обраний алгоритм ЗНМ Yolo як такий, який ефективно вирішує задачі розпізнавання об’єктів у реальному часі. У той час як інші моделі розпізнавання об'єктів, такі як ResNet, R-CNN або Faster R-CNN, можуть бути більш точними, моделі Yolo набагато швидше досягають розпізнавання об'єктів у реальному часі.

1.3. Аналіз сімейства моделей згорткових нейронних мереж Yolo

Коротко розглянемо типові архітектури ЗНМ Yolo:

- Yolov3 – алгоритм відстеження об’єктів у реальному часі для розпізнавання довільного класу об’єктів. Поліпшення у порівнянні з попередніми версіями Yolo полягають у наступному: додаванні оцінки ймовірності на усіх вихідних шарах ЗНМ, для покращення розпізнавання малих об’єктів. Це дозволяє виконувати операції визначення ймовірності та координат розпізнаного

об’єкта за один етап. Як хребет (backbone) для виділяння ознак було обрано Darknet-53 замість ResNet-152 [4, 41]. Як функцію активації використовує Relu [17];

- Yolov4 – оновлена версія Yolov3, яка показує поліпшення значення середньої влучності mAP (mean Average Percision) на 10% та FPS на 12% в порівнянні з попередньою моделлю [4, 42]. Для виділяння ознак в основній версії використовується CSPDarknet53 та блок Path Aggregation Network для агрегації параметрів між рівнями хребта ЗНМ. При чому як голова (head) для виконання задач висновування використовується yolov3. Як шия використовується Spatial Pyramid Pooling (SPP). Були введені такі поняття як bag of specials – плагіни для пост обробки, до яких належить функція активації Mish [9], а також bag of freebies – методи для поліпшення процесу тренування, наприклад фільтр CIOU (complete intersection over union) [4, 42].Структурна схема моделі yolov4 наведена на Рис. 1.17;

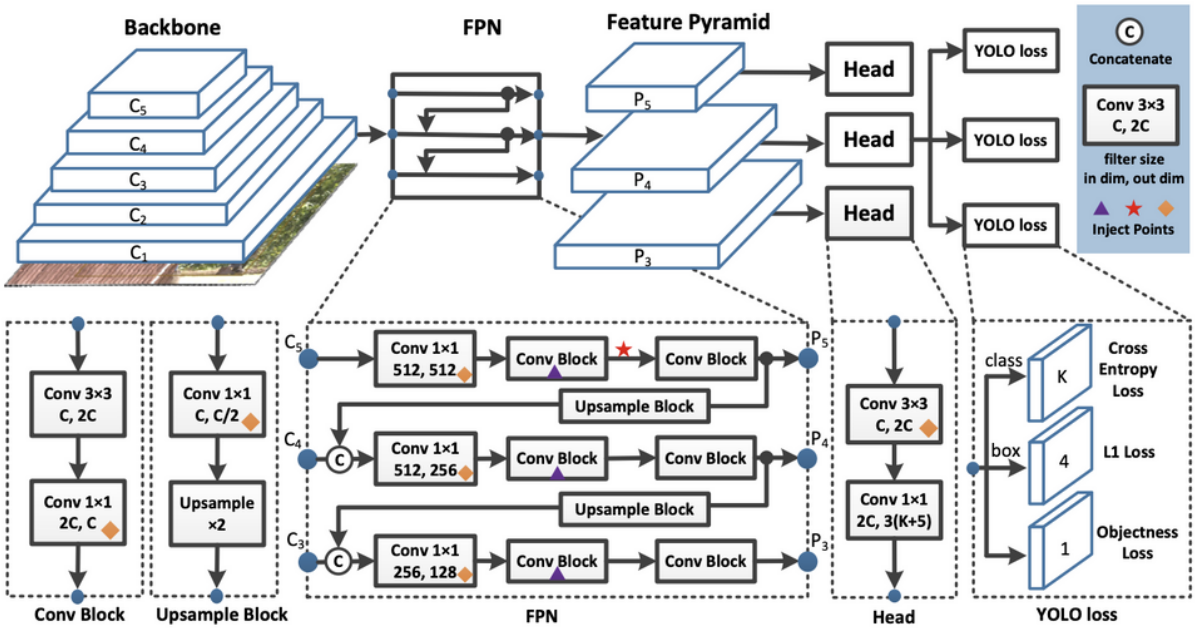


Рис. 1.17. Типова архітектура ЗНМ YOLOv4.

- Yolov4 Scaled – модифікована версія yolov4. Має додаткові шари backbone: ResNet та ResNeXt на базі основного хребта для виділяння ознак CSPDarknet53. Був доданий функціонал для масштабування потужностей моделі при використанні значних апаратних засобів [4, 43];

- Yolov5 – переписана версія моделі yolov4 з фреймворку Darknet мовою C на фреймворк для розробки моделей ШІ Pytorch на мові Python. При збільшенні масштабованих можливостей для розгортання моделей, було визначено що показник mAP є меншим у порівнянні з аналогічними моделями yolov4 [4, 44];
- YoloR – Продовження дослідження в рамках підвищення ефективності моделі yolov4. Пропонується ідея додати до механізму навчання не тільки навчання з учителеми, але і навчання без учителя (за аналогію з підсвідомістю людини) [45];
- YoloS– Модифікована версія Yolov4, де в якості backbone виступає модель Transofrmer замість ЗНМ (Рис. 1.18) [46].

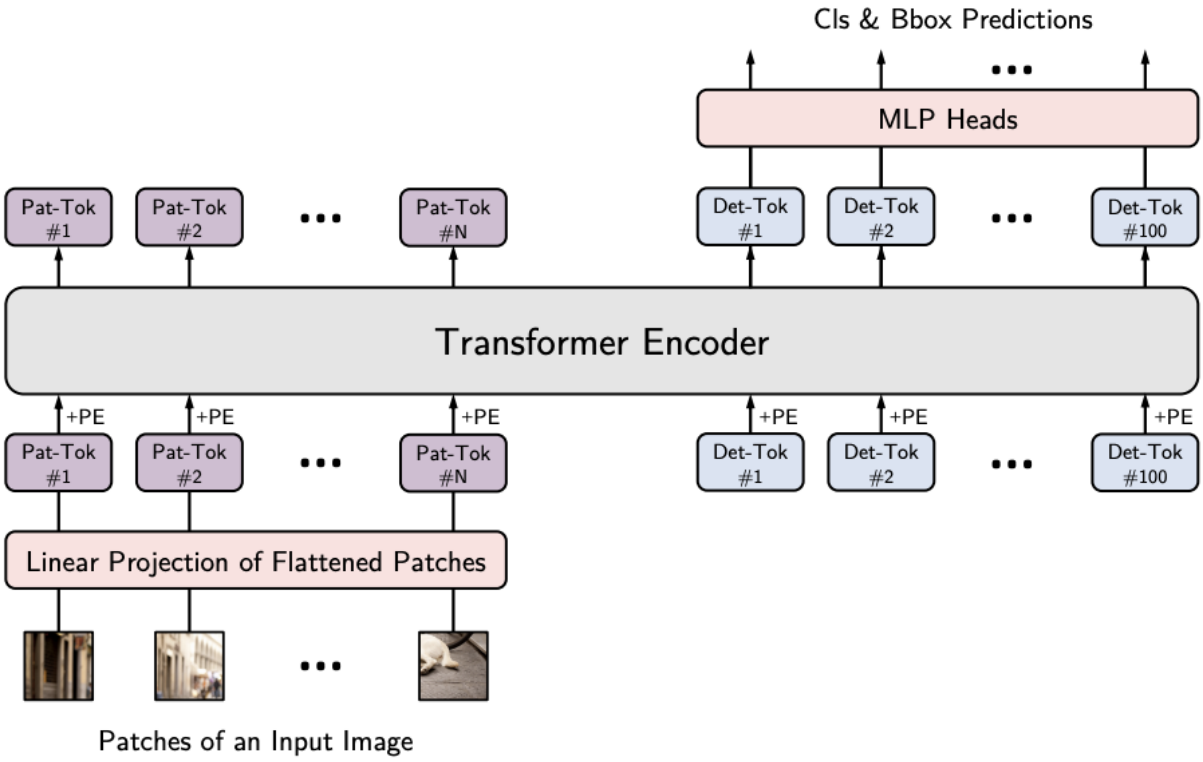


Рис. 1.18. Типова архітектура ЗНМ YoloS.

Узагальнена таблиця з результатами порівнянь роботи алгоритмів ЗНМ від різних дослідників продемонстрована в таблиці 1.3. Основними метриками для визначення ефективності ЗНМ були: mAP для кожного запиту, кількість кадрів у секунду (FPS) F-міра, точність та швидкодія.

Таблиця 1.3. Порівняльний аналіз алгоритмів ЗНМ для задач РО.

Посилання на дослідження	**Використаний Набір даних**	**Використані алгоритми**	**Результати дослідження**
Li et al., 2021 [47]	Remote sensing from GF-1 and GF-2 satellites. **Тренувальна вибірка**: 826 зображень. **Тестова вибірка**: 275 зображень. **Розмір Зображень**: 300 × 300, 416 × 416, 500 × 500, 800 × 800, 1000 × 1000	Faster R-CNN YOLO v3 SSD	YOLOv3 має більшу mAP та FPS ніж SSD та Faster R-CNN.
Benjdira et al., 2019 [48]	UAV dataset. **Тренувальна вибірка**: 218 зображень **Тестова вибірка**: 52 зображень **Розмір Зображень**: 600 × 600 to 1024 × 1024	Faster R-CNN YOLOv3	YOLOv3 має вищу F-міру та FPS ніж Faster R-CNN.
Zhao et al., 2019 [49]	Google Earth and DOTA dataset. **Тренувальна вибірка**: 224 Images **Тестова вибірка**: 56 Images **Розмір Зображень**: 600 × 600 to 1500 × 1500	SSD Faster R-CNN YOLOv3	YOLOv3 має вищі mAP та FPS ніж Faster R-CNN та SSD.
Kim et al., 2020 [50]	Korea expressway dataset **Тренувальна вибірка**: 2620 **Тестова вибірка**: 568 **Розмір Зображень**: не зазначено	YOLOv4 SSD Faster R-CNN	YOLOv4 має вищу точність SSD має вищу швидкість розпізнавання

Продовження таблиці 1.3.

Dorrer et al., 2020 [51]	Custom Refrigerator images **Тренувальна вибірка**: 800 зображень **Розмір Зображень**: не зазначено	YOLOv4 YOLOv5l	YOLOv4 має вищу mAP ніж YOLOv5l.
Rahman et al., 2021 [52]	Custom Electrical dataset **Тренувальна вибірка**: 5939 **Тестова вибірка**: 1400 **Розмір Зображень**: не зазначено	YOLOv4 YOLOv5l	YOLOv4 має вищу mAP порівнюючи з YOLOv5l.
Long et al., 2020 [53]	MS COCO dataset **Тренувальна вибірка**:: 118,000 **Тестова вибірка**: 5000 **Розмір Зображень**: не зазначено	YOLOv3 YOLOv4	YOLOv4 має вищу mAP порівнюючи з YOLOv3
Bochkovskiy et al., 2020 [54]	MS COCO dataset **Тренувальна вибірка**: 118,000 **Тестова вибірка**: 5000 **Розмір Зображень**: не зазначено	YOLOv3 YOLOv4	YOLOv4 має вищу mAP та FPS ніж YOLOv3
Ge et al., 2021 [55]	MS COCO dataset **Тренувальна вибірка**: 118,000 **Тестова вибірка**: 5000 **Розмір Зображень**: не зазначено	YOLOv3 YOLOv4 YOLOv5 YOLOv5I	YOLOv5 має вищу mAP ніж YOLOv3 та YOLOv5l YOLOv3 має вищий FPS ніж YOLOv4 та YOLOv5l

Аналізуючи існуючі впровадження алгоритмів Yolo та інших ЗНМ розглянутих у попередніх розділах, можна зробити висновок що Yolov4 має більшу точність у розпізнаванні за решту алгоритмів ЗНМ, хоча деякі джерела

стверджують що Yolov5 має перевагу. Причина різних результатів може залежати від наборів вхідних даних (вибірка, розмір зображень), модифікованих гіперпараметрах, тощо [56]. Алгоритми YoloR та YoloS на момент написання дисертаційного дослідження ще не були достатньо оцінені, тому не брали участі у аналізі.

Для подальших досліджень обрано алгоритм Yolov4, оскільки була продемонстрована висока точність, FPS та mAP такого підходу серед досліджених алгоритмів ЗНМ.

1.4. Особливості пошуку та розпізнавання об’єктів на мобільній платформі.

Мобільна платформа (МП) – пристрій або операційна система з обмеженими апаратними можливостями, в порівнянні з стаціонарними приладами.

Як було визначено у пункті 1.1.3, для виконання задач РО необхідно виконати два етапи: навчання, та висновування моделі ШНМ. Оскільки операція висновування, в порівнянні з навчанням, не вимагає великих апаратних затрат, її доречно виконувати на кінцевій платформі користувача, у тому числі і на МП. Дамо загальне поняття, на яких платформах можуть виконуватись обидві операції (Рис 1.19).

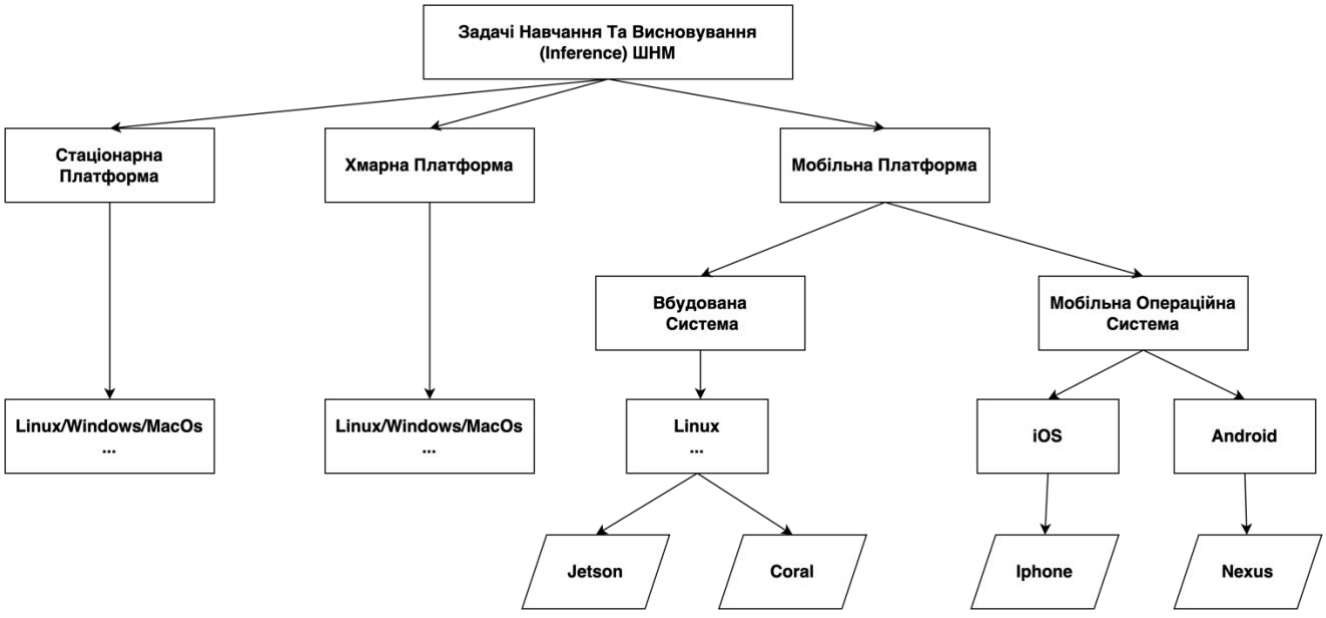


Рис. 1.19. Типи платформ, які підтримують задачі РО. Також зазначено деякі приклади операційних систем та прикладів пристроїв.

Дамо визначення кожному з понять:

- Мобільна Операційна Система (МОС) – операційна система для мобільних телефонів, розумних годинників, планшетів тощо;
- Вбудована Система (embedded system, ВС) – спеціалізований обчислювальний пристрій для виконання обмеженої кількості задач з обмеженнями системи реального часу (СРЧ);
- СРЧ – це така система, яка впливає на досліджене середовище в межах заданих часових проміжків. Тобто це система для якої важливим є час отримання результату. Метрика таких систем тісно пов'язана з обробкою відеозображень для задач РО [57];
- Відеозображення – технологія передачі візуальної інформації у вигляді потоку даних у СРЧ. Можна виділити наступні характеристики: FPS, роздільна здатність, співвідношення сторін екрану, ширина відео потоку;
- Хмарна Платформа (ХП) – модель забезпечення ресурсів комп'ютерної системи без прямого доступу користувача. Великі хмарні сервіси забезпечують різні рівні захищеності, масштабування, реплікації та резервування даних;
- Стаціонарна Платформа (СП) – не мобільна платформа, яка зазвичай зберігається стаціонарно у приміщенні. Зазвичай має більші апаратні можливості, порівнюючи її з МП.

Також варто виділити кросплатформні рішення для МОС. Такі системи дозволяють запускати один і той же додаток на таких платформах як Android та iOS, використовуючи різні типи компіляторів в залежності від типу МП, наприклад Just In Time Compilator (JIT) та Ahead Of Time Compilator (AOT) [8-9]. Такі підходи дозволяють значно полегшити розробку програмного забезпечення одночасно на декілька МП, проте специфічність задач РО на таких пристроях не завжди може бути забезпечена кросплатформними рішеннями.

Проведемо аналіз апаратних та програмних можливостей МП, ХП та СП для виконання задач РО (Таблиця 1.4) [58-60]. За метрику взято апаратні реалізації чипів, можливості використання різного роду процесорів, та програмні

реалізації. В залежності від платформи та компанії виробника чипу, використовують наступні типи процесорів для виконання задач РО:

- CPU – вузол керування який виконує інструкції, що складаються з комп'ютерної програми. Такий пристрій виконує основні арифметичні, логічні, керуючі та вхідні/вихідні (I/O) операції, визначені інструкціями в програмі;
- GPU – спеціалізована електронна схема, призначена для маніпулювання і зміни пам'яті для прискорення створення зображень в буфері кадру, який використовується для виведення на пристрій відеозображення;
- Neural Network Accelerator (NNA) – клас мікропроцесорів, розроблений для прискоренні обчислень ШНМ та інших алгоритмів МН. Відрізняються від GPU відсутністю вузлів фіксованого призначення для графіки і використовують арифметику низької розрядності;
- Vision Processing Unit (VPU) – клас мікропроцесорів, розроблений для прискоренні обчислень ШНМ в області КЗ. На відміну від GPU, архітектура пам’яті таких процесорів оптимізована для читання текстур і внесення змін до кадрових буферів в оперативній пам’яті;
- Neural Processing Unit (NPU) – спеціалізована електронна схема, призначена для алгоритмів ГН, призначений для роботи на МП Huawei;
- Tensor Processing Unit (TPU) – інтегральна схема специфічного застосування, призначена для прискорення обчислень ШІ. Розроблений Google [61].

Таблиця 1.4. Апаратні та програмні можливості МП, СП, ХП для задач РО.

Платформа	**Доступні апаратні засоби Для задач РО**	**Тип Процесорів**	**Програмні засоби**
МОС iOS	CPU, GPU, NNA Примітка: вбудована можливість CoreML автоматично перемикатися між CPU , GPU та NNA.	ARM (Apple Silicon, M1)	1) Засіб запуску ШНМ на iOS: CoreML (Apple): Підтримка акселерації - Metal Performance - Shaders (Metal) - BNNS Підтримка фреймворків:

Продовження таблиці 1.4

			- Tensorflow 1,2 - Pytroch 2) Tensorflow Lite (Google) 3) MNN - apple-cpu 4) Tensorflow.js, ONNX.js
МОС Android	CPU, GPU, NPU, VPU …	Coral, Qualcomm та інші	1) ONNX (Microsoft) 2) Tensorflow Lite, Tensorflow (Google) 3) Pytorch 4) MNN - intel-cpu - amd-cpu 5) Tensorflow.js, ONNX.js
ВС Nvidia Jetson	CPU, GPU, Neon coprocessor	ARM NVIDIA Carmel, NVIDIA Maxwell/Pascal architecture	1) TensorRT (Nvidia), Підтримка фреймворку CUDA 2) OpenCV Підтримка фреймворку CUDA 3) DeepStream 4) Tensorflow.js, ONNX.js
ВС Google Edge Coral	CPU, GPU, TPU coprocessor, VPU	ARM Cortex, Imagination PowerVR	1) Tensorflow Lite
СП Linux **ХП Linux** **Google Collab** **…**	CPU, GPU, TPU, NPU, VPU …	ARM, NVIDIA, Intel …	1) TensorRT (Nvidia), Підтримка фреймворку CUDA 2) OpenCV Підтримка фреймворку CUDA 3) DeepStream 4) Tensorflow (Google) 5) ONNX (Microsoft) 6) Pytorch 7) Tensorflow.js, ONNX.ji

Аналіз існуючих засобів показав, що перспективною МП для виконання задача РО є iOS через наявні можливості акселерації процесорів, та можливості

розумного перемикання між CPU, GPU та NNA. Аналіз показав, що для виконання задачі РО на МОС iOS необхідно використати фреймворк CoreML [62].

CoreML – фреймворк для алгоритмів МН від компанії Apple, головною особливістю якого є можливість портування моделей нейронної мережі під вимоги процесора. Таким чином, при виконанні етапу виснування можуть застосовуватись CPU, GPU та NNA. Структурна схема наведена на Рис 1.20.

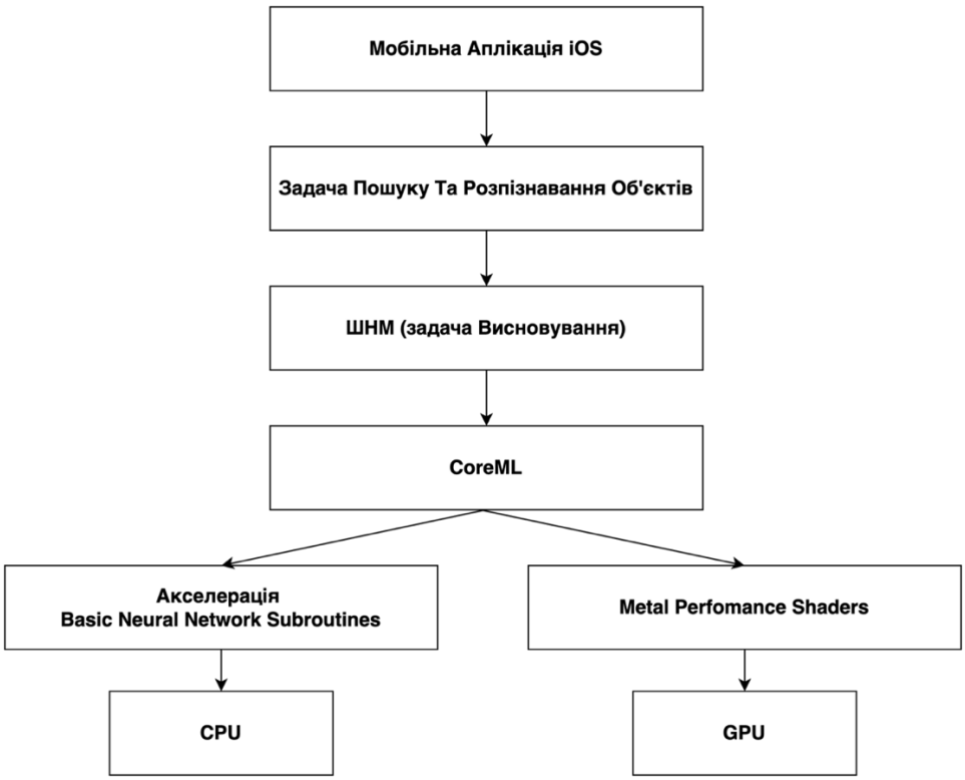


Рис. 1.20. Фреймворк CoreML та його компоненти для задач РО.

Для досягнення цілей РО, CoreML дає можливість використовувати наступні примітиви продуктивності (Perfomance Primitives):

- Basic Neural Network Subroutines (BNNS) – під модуль акселерації, який запускає примітиви шари ЗНМ на етапі висновування. Запускається лише на CPU. Головна ідея полягає у тому що не потрібно ре-імплементовувати шари ЗНМ з кожною новою ітерацією РО. Це важливо оскільки у типових ЗНМ 70% енергії йде на обробку шарів ЗНМ [63];
- Metal Perfomance Shaders (MPS) – технологія призначена для паралельної обробки даних з використанням графічного адаптера (GPU). Внутрішні

компоненти MPS автоматично визначають модель адаптера та групу, до якої він належить, що дозволяє застосовувати оптимальні інструкції для обробки даних з урахуванням обмежень та особливостей конкретного графічного ядра. Дозволяє реалізувати конвеєрну обробку вхідних даних, максимально ефективно утилізуючи переваги одночасної та незалежної роботи [64].

Аналіз літературних джерел показав що для виконання задачі висновування доречно використати МОС iOS та ВС Jetson, які показали високі показники швикдодії та продуктивності для задач РО в МП [65-68].

1.5. Аналіз систем масштабування засобів пошуку та розпізнавання об’єктів у реальному часі.

Масштабованість – здатність збільшувати свої можливості шляхом збільшення числа функціональних блоків, які виконують суміжні задачі.

Системи масштабування розділяють за рівнями відповідальності користувачів та провайдерів. Серед таких моделей обслуговування виділяються наступні типи (Рис 1.21).

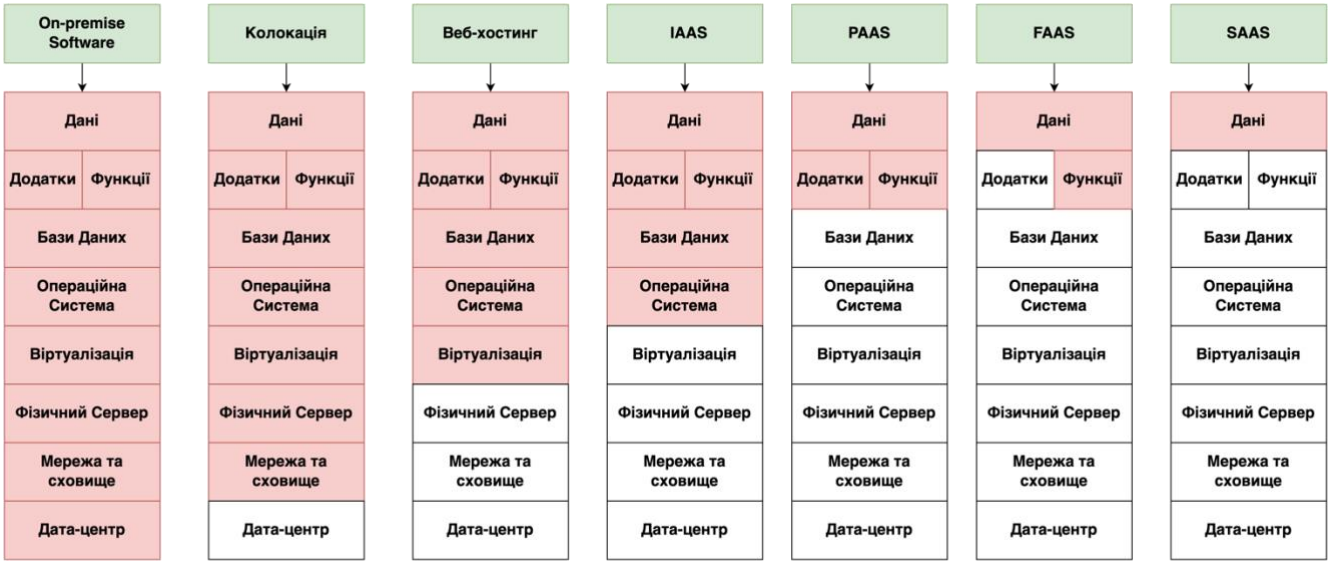


Рис. 1.21. Типи систем масштабування в залежності від задач які виконує споживач та провайдер. Червоним відмічені задачі, які виконує споживач, прозорим – задачі які виконує провайдер замість користувача.

Розглянемо кожну платформу окремо:

- On-premise software – стаціонарна IT інфраструктура де всі задачі виконуються

на одному або декількох пристроях в рамках одного приміщення/будівлі;

- Колокація – модель клієнтських послуг, що передбачає розміщення сервера на фізичному технічному майданчику провайдера, або надання провайдером в оренду користувачеві власного сервера. Вважається, що технічний майданчик провайдера – інтернет-вузол (дата-центр), обладнаний системами резервного живлення та кліматичного контролю і підімкнений до інтернету швидкісними каналами зв'язку. Послуга інтенсивно розвивається на ринках розвинених країн, де інтернет все частіше використовується як бізнес-середовище [69];
- Веб-хостинг – розширене поняття колокації. Фізична інформація зберігається на сервері провайдера;
- IAAS – Інфраструктура як сервіс. Користувач отримує від провайдера виділені сервери або віртуальні машини, дискові сховища і магістральна з’єднання до мережі інтернет. Клієнт оплачує послуги в залежності від операційних витрат, наприклад використання CPU чи GPU сервера. Програмне забезпечення встановлюється самостійно. Приклад застосувань це віртуальні машини та контейнери;
- PAAS – Платформа як сервіс. На відміну від IAAS, провайдер дає доступ користувачу до програмного забезпечення, наприклад встановлену операційну систему, середовище виконання, відлагодження та запуску. Задачі які покладаються на такі платформи є наступні: масштабування, кластеризації, розгортання додатку, оркестрація контейнерів. Прикладом такої системи є Docker, який також визначається як CAAS (Container as a service) [70];
- SAAS – Програмне забезпечення як сервіс. На відміну від IASS чи PASS, провайдер бере на себе усі задачі по розгортуванню, обслуговуванню та запуску додатків. Недоліком таких систем є обмеженість кастомізації. Прикладом таких систем є Amazon Web Services, Google Cloud Collab, Robolow, Kaggle тощо [71];
- FAAS – Функція як сервіс. Наступний після SAAS рівень абстракції, в якому повністю зникають фізична та програмна архітектура, окрім окремих функцій, які викликаються при потребі. Така модель дуже економна, оскільки користувач сплачує лише за серверний час, необхідний для виконання функції. Прикладом

реалізації є Serverless Computing. Прикладом таких систем від провайдерів є AWS Lambda, Google Cloud Functions, IBM Cloud Functions [72].

Існує безліч систем для масштабованого пошуку та розпізнавання об’єктів у реальному часі. Серед таких систем варто виділити Opendatacam та DeepStream Software Development Kit (SDK) [5].

Opendatacam - це система з відкритим вихідним кодом для постійного моніторингу об'єктів за допомогою моделі МН. Система може бути інтегрована на МП, такі як Jetson Nano, Raspberry Pi або операційна система Android, або виконана на машинах на базі Linux через веб-сервер. Така система може зчитувати дані з камери або через шину послідовного інтерфейсу камери (CSI) [73].

DeepStream SDK дозволяє виконувати програми безпосередньо на МП Nvidia, таких як Jetson Nano або Jetson Xavier NX. Крім того, він забезпечує більш швидку передачу даних між візуальним датчиком (Camera) через шину CSI. Однак він не підтримує багато варіантів налаштування моделі МН та параметрів відстеження об’єктів [74].

У той же час такі SAAS та FAAS системи від Amazon: Lambda, S3, ECS, Sagemaker чи Google Clould Collab від Google пропонують ефективні масштабовані рішення, їхня ефективність напряму залежить від витрачених коштів. І у більшості випадків IAAS чи PAAS системи цілком достатньо для виконання досліджень [75].

Для виконання цілей дисертаційних досліджень, було обрано PAAS систему масштабування та контейнеризації Docker [76], оскільки вона показала ефективність при розв’язку задач РО та ГН [77].

Docker – платформа для управління ізольованими Linux контейнерами на рівні окремих процесів. Такі процеси згодом можна переносити і клонувати у виді контейнера до інших серверів. Основні особливості: ізольованість середовище та масштабованість системи контейнерів. Головна відмінність такої технології від віртуальної це рівень ізольованості на рівні операційної системи, а не апаратного забезпечення, та швидкість виконання.

Узагальнена структурна схема системи Docker наведена на Рис. 1.22.

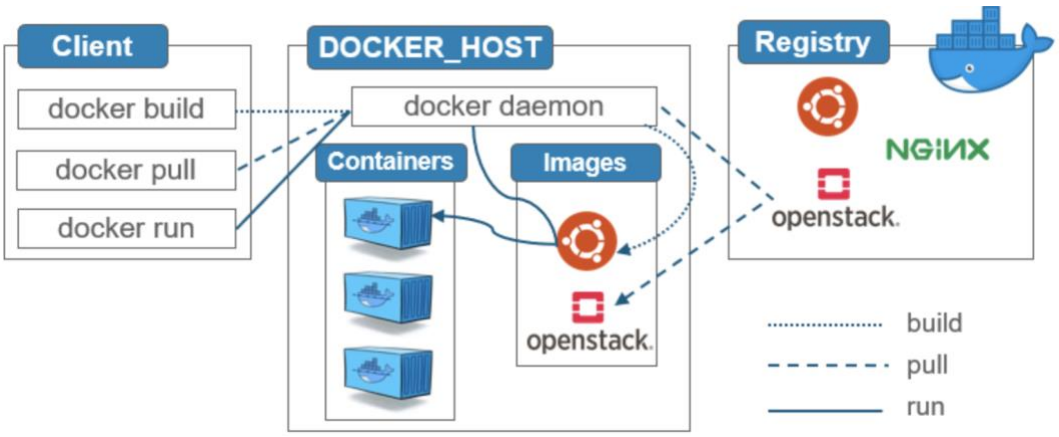


Рис. 1.22. Узагальнена структурна схема Docker

Де:

- Container (контейнер) – виконуваний пакет програмного забезпечення;
- Image (образ) – незмінний файл (образ) через який розгортаються контейнери;

Для виконання задачі тренування обрано On Premise Software СП Linux, на якій зручно та швидко реалізувати потрібну систему масштабування.

1.6. Алгоритми оперативного відстеження класу розпізнаних об’єктів.

1.6.1. Загальні поняття.

Відстеження об’єктів (ВО) – процес пошуку одного чи більше об’єкта на відеозображенні. Головне завдання алгоритмів відстеження – це послідовний аналіз кадрів відео для оцінення параметрів руху, та положення вихідного об’єкта [78]. Серед основних проблем, які виникають при виконанні задачі відстеження можна виділити: визначення траєкторії руху при високому рівні FPS у реальному часі; визначення положення об’єкта на цільовому відеопотоці, при високій кількості вхідних об’єктів, які можуть перекривати один одного. Отже, при створенні моделі ВО важливо визначити, як може змінюватись об’єкт при русі у реальному часі [5, 78].

Системи ВО можна розділити на два основні етапи:

Target Representation and Localization (представлення та локалізація об’єкта).

Цей етап є послідовним та висхідним, тобто наступні кроки ВО не зачіпають попередніх. При цьому складність алгоритму для обчислень на цьому етапі є достатньо малою [78]. Серед прикладів таких алгоритмів можна виділити:

- Відстеження Точкових Особливостей Сцени (Point feature tracking): Полягає у отриманні максимально точних послідовностей координат проекції точок вхідного відеопотоку [79];
- Відстеження Контурів (Contour Tracking): полягає у пошуку граничної межі об'єкта. Ітераційно модифіковує початковий контур, ініціалізований з попереднього кадру, на його нове положення в поточному кадрі. Цей підхід розвиває контур за рахунок мінімізації енергії контуру, використовуючи алгоритм градієнтного спуску (gradient descent) [79].

Filtering and Data Association (фільтрація та об’єднання даних про об’єкт).

Цей етап є послідовним та низхідним, тобто спершу проводиться обробка більш загальної інформації про об’єкт, а лише потім обчислюються гіпотези передбачуваного руху об’єкта. При цьому складність алгоритму для обчислень на цьому етапі є достатньо високою [80]. Серед прикладів таких алгоритмів можна виділити:

- Фільтр Калмана – або лінійна квадратична оцінка, це алгоритм, який використовує серію вимірювань, які спостерігаються протягом часу, включаючи різного роду шуми та неточності, та створює оцінку невідомих змінних, яка є більш точною у порівнянні з одинарними вимірюваннями. Активно використовується з 1960-х років, у тому числі при розробці програми траєкторії польоту на місяць [81];
- Фільтр частинок – рекурсивний алгоритм для численного вирішення проблеми оцінювання, особливо для не-гаусових та нелінійних процесів. На відміну від фільтра Калмана, не залежить від методів локалізації та апроксимації [79].

1.6.2. Відстеження об’єктів за допомогою згорткової нейронної мережі.

У той же час розглядають наступні підходи до ВО для задач РО у реальному часі за допомогою методів ГН та моделей ЗНМ (Рис. 1.23) [82].

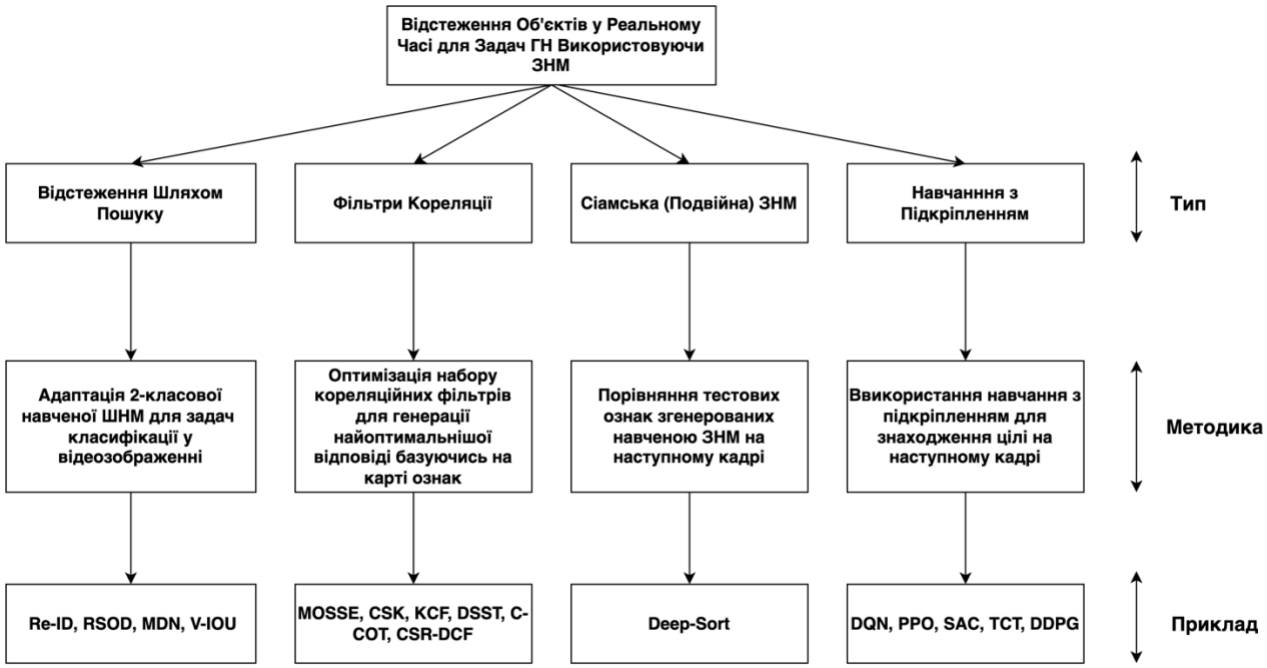


Рис. 1.23. Типові приклади ВО для задач ГН за допомогою ЗНМ.

З розвитком підходів ГН, зростає попит на методи пошуку та відстежування об’єктів за допомогою ЗНМ. Наприклад у змаганні на найефективніший алгоритм 2018 Visual Object Tracking (VOT 2018), усі призові місця отримали алгоритми, які використовують ГН [82].

Такі алгоритми умовно можна розділити на 4 групи: Відстеження шляхом пошуку (Tracking by Detection), Фільтри Кореляції, Сіамські (Подвійні) ЗНМ, Навчання з підкріпленням.

Відстеження шляхом пошуку – підхід для багато-класового відстеження, який полягає у відбиранні ознак з вхідного зображення, при аналізі регіонів такого зображення. Позитивні та негативні випадки в навчальних вибірках відбираються на основі оцінки Intersection Over Union (IoU) з основною істиною. Типовою проблемою для таких систем є обмежена продуктивність базового детектора, який може призвести до хибно-позитивних (FP) і пропущених виявлень. Розглянемо приклади реалізації такого підходу.

Зокрема, модель повторної ідентифікації (Re-ID) [83] заснована на

оптимізованій хребті (backbone) DenseNet121 з функцією втрат. Ця модель застосовує блок Squeeze-and-Excitation (SE), щоб автоматично отримати важливість кожної функції каналу та присвоїти їй відповідну вагу. Ознаки (features) переносяться в глибокий шар шляхом регулювання відповідних ваг, що знижує передачу зайвої інформації в процесі повторного використання функції в DenseNet121. Запропонована модель використовує додаткові переваги ознак середніх особливостей ЗНМ для підвищення здатності ознак функцій [83].

У той же час алгоритм Виявлення Малих Об'єктів у Реальному Часі (RSOD) [84] покращує точність виявлення невеликих об'єктів за допомогою

- використання карт ознак меншого шару, що містить більш дрібнозернисту інформацію для прогнозування місця розташування;
- злиття локальних і глобальних особливостей неглибоких і глибоких ознак функцій в Feature Pyramid Network (FPN) для підвищення здатності до вилучення більш репрезентативних функцій;
- присвоєння ваг вихідним особливостям FPN і їх адаптивне об'єднання;
- покращення шару збудження в механізмі уваги Squeeze-and-Excitation (SE), щоб точніше налаштувати реакції функцій кожного каналу [84].

Такі підходи можна вважати хорошим початком для реалізації власних методів і засобів алгоритму відстеження. Однак такі системи застосовувалися тільки для завдань автомобільного руху, де об'єкти зазвичай рухаються в строгих схемах. Тому додатково можна розглянути системи відстеження руху мурах [85-86]. Наприклад, у даній публікації [85] автори представляють свій фреймворк виявлення та відстеження руху мурах. Вони пропонують:

- прийняття двоступеневої структури виявлення об'єктів з використанням ResNet-50 (Residual Network with 50 Layers) як хреба (backbone) та кодування позиції областей, що представляють інтерес для точного знаходження мурах;
- використання моделі ResNet-50 для розробки дескрипторів траєкторії мурах;
- побудова довгострокових послідовностей розпізнавання та об'єднання їх з інформацією про рух для досягнення онлайн-відстеження.

Розглянемо також таку нейронну мережу як MDNet (Multi-Domain Convolutional Network Tracker). У задачах ВО існують деякі бажані властивості для навчання цільового представлення, такі як: інваріантність щодо освітлення, масштабу, перспективи та розмиття руху. Метою використання мультидоменного навчання є вивчення дискримінаційної моделі, яка вивчає спільне представлення цілі в різних областях. Щоб досягти цього, MDNet навчається в автономному режимі з великим набором відеопослідовностей, де кожна послідовність розглядається як домен. Для точної локалізації об’єкта під час відстеження використовується техніка регресії обмежувальної рамки на обмежувальних прямокутниках із високими показниками.

Для алгоритмічного та ефективного пошуку на базі мінімізаційного фільтру IOU був запропонований алгоритм V-IOU [87].

Він полягає у динамічному перекриванні локалізованих регіонів двох об’єктів за допомогою IOU.

У той же час, у статтях [88] пропонується онлайн-структура тестів Multiple Object Tracking (MOT) для відстеження руху мурах. Цей фреймворк поєднує в собі як розпізнавання, так і відстеження руху таких малих об’єктів. Методи та засоби цього фреймворку дозволяють ефективно запобігати перериванню фрагментів траєкторії, та ID (True Positive Id's), що є частою проблемою, оскільки мурахи можуть змінювати траєкторію руху [86].

ЗНМ на базі фільтрів кореляції – полягає у навчанні моделі кореляційного фільтра, для відокремлення об’єкта від заднього фону для кожного кадру відеопотоку у реальному часі. При цьому параметри такого фільтру підбираються таким чином, щоб результуюча згортка виглядала як розподіл Гаусса з піком у центрі об’єкта відстеження. Для навчання використовується вибірка з мінімізованою помилкою між розподілом Гаусса та вихідним результатом за допомогою методу найменших квадратів [90].

Кореляційні фільтри – це клас класифікаторів, які оптимізовані для створення різких граничних значень кореляційного виходу, головним чином для досягнення точної локалізації цілей у сценах.

Історично, можна описати розвиток застосування кореляційних фільтрів наступним чином:

Використовуючи сірі зображення, фільтр мінімальної вихідної суми квадратичних помилок (MOSSE) [91] вперше застосовує кореляційний фільтр у галузі відстеження. Цей фільтр легко обчислити, і він може швидко відстежувати об’єкти, але він не гарантує точного відстеження змін зовнішнього вигляду об’єкта. Після цього, у 2012 році, Henriques et al. [92] запропонував circulant structure tracking with kernels (CSK). Згодом, у 2014 році, Danwelljan et al. [93] запропонував щоб Kernels correlation filter (KCF) додатково налаштував характеристики каналу для багатоканальних функцій і ввів функції CN для відстеження. Функція CN покращує дискримінаційну здатність фільтра. Однак адаптивність фільтра до обертання, поза межами поля зору та швидкого руху все ще потребує вдосконалення. Наступним етапом була робота Danelljan et al. [94], яка пропонує discriminative scale space tracker (DSST) використовуючи піраміду ознак для вирішення проблеми багатомасштабної варіації, а також запропонував вдосконалений алгоритм DSST [95]. Зі швидким розвитком методів ГН, алгоритм C-COT [96.] дозволяє ефективно представити інформацію про просторову позицію з поверхневими функціями ЗНМ, що є комбінацією кореляційної фільтрації та ЗНМ. Алгоритм переміг у конкурсі VOT2016. Подібно до C-COT, алгоритм CSR-DCF [97] також застосовує функції ЗНМ до алгоритму кореляційної фільтрації, що покращує стійкість алгоритмів [98].

Сіамські (Подвійні) ЗНМ – це різновид ШНМ що складається з двох ідентичних нейронних підмереж з однаковими наборами ваг. Даний вид мереж дозволяє порівняти вектор ознак двох об'єктів з метою виділити їх семантичну подібність або відмінність. Сіамська нейронна мережа є нелінійним відображенням даних з метою наблизити один до одного схожі об'єкти і рознести різні об'єкти на максимально можливу відстань. Сіамські мережі отримали свою назву від сіамських близнюків, що фізично приросли один до одного, через використання відразу двох підмереж, що розділяють один набір ваг.

Прикладом таких ЗНМ є Deep Sort [99]. Цей підхід використовується для

задач відстеження людей (Human Tracking). Принцип роботи базується на відомому фільтрі Калмана та відстані Махаланобіса.

Відстань Махаланобіса — це така міра відстані між векторами випадкових величин, що узагальнює значення евклідової відстані. Евклідова відстань – відстань між двома точками в евклідовому просторі. Застосовується у випадках, коли евклідова відстань для середнього значення не може дати вірного розподілення для нових вимірів, при сильній кореляції. Визначається формулою [100]:

$$d(X,Y,S) = \sqrt{(X-Y)^T \times S^{-1} \times (X-Y)}$$

де X, Y — вектори, де різниця визначається між новою точкою (X) та середнім значенням для кожної змінної (Y), S — коваріаційна матриця, T — операція транспонування.

Коваріаційна матриця – це квадратна матриця, яка визначає коваріацію між кожною парою елементів заданого випадкового вектора.

Задача відстані Махаланобіса полягає у: забиранні коваріацій змінних, прийняття дисперсії змінних рівній 1, та використанні евклідової відстані для трансформації даних. Таким чином, чим вище кореляція між змінними, тим швидше можна скоротити відстань, так як буде відбуватись множення обернене найбільшому значенню [100].

Таким чином, алгоритм DeepSort використовує поняття відстані Махаланобіса та фільтру Калмана для того щоб переносити інформацію від одного кадра до наступного. На першій фазі, за допомогою методів РО, визначаються регіони та класи об’єкту розпізнавання. На наступній фазі, застосовуються такі алгоритми відстеження як Угорський алгоритм, для того щоб зв’язати певні об’єкти з об’єктами які раніше відслідковувались за допомогою фільтрів Калмана.

Також у Deep Sort, у порівнянні з попередніми рішеннями [101] додається нова метрика – appearance, яка будується на основі Сіамської ЗНМ. Вона полягає у тренування окремої ШНМ на базі картинок скупчень людей. Така ШНМ вирішує проблему, коли один об’єкт (людина) може закривати на декілька

ітерації кадрів інший об’єкт (іншу людину), при цьому відбувається заміщення ідентифікатора. Така ШНМ дозволяє «поглибити» пам’ять об’єктів алгоритму.

Недоліком алгоритму Deep Sort є використання лише для задач Human Tracking. Для відстеження іншого класу об’єктів необхідно перенавчати RE-ID ШНМ, що не завжди можливо.

Навчання з Підкріпленням для задач ВО – як зазначалось у пункті 1.1.4, підхід навчання з підркіпленням полягає у навчанні знаходити оптимальний шлях в середовищі, базуючись на отриманій нагороді. Агент, як правило, отримує спостереження в дискретних часових кроках з винагородами і вибирає дію з набору доступних опцій [3].

Існують наступні алгоритми для навчання системи з неперервним простором дій в задачах пошуку та ВО :

- Deep Q–Network (DQN) – алгоритм який комбінує Q–навчання з глибинним нейронними мережами, що дозволяє вирішувати високо–масштабовані задачі в ігровій та робо-технічній індустрії [102];
- Deep Deterministic Policy Gradient (DDPG) – модель, розроблена OpenAi, з використанням моделі «Actor–Critic». Цей підхід тісно пов’язаний з DQN: при виконанні команд, Actor (актор) намагається навчитися рухатись у певному напрямку, в залежності заданих умов Critic(критика) [3, 103];
- Proximal Policy Optimization (PPO) – суть алгоритму полягає у наближеній оптимізації стратегії, тобто коли відбувається пошук поля стратегії (field policy), без призначень значень до пар «стан-дія» [104];
- Soft Actor–Crititc (SAC) – розширена версія алгоритму PPO, зі збільшенням рівня ентропії градієнту стратегії (policy gradient), що дозволяє діяти в більш випадкових ситуаціях вхідного середовища [105];
- Target Candidate Track (TCT) – полягає у безперервому слідкуванні за об’єктами, прив’язуючи попереднє та поточне положення об’єкта у сміжних кадрах. Базується на фільтрі Калмана. Як алгоритм збіжності між двома розпізнанними об’єктами використовується Угорський алгоритм [3, 106].

Припускаючи результати авторів, тестова структура МОТ може бути використаною для перевірки продуктивності алгоритмів відстеження.

Серед ефективних алгоритмів оптимізації задачі відстеження, можна виділити алгоритми Kd-tree [107] та Угорський алгоритм [108]. Варто відзначити, що усе частіше алгоритми відстеження використовуються у кіберфізичних системах.

1.6.3. Використання алгоритмів відстеження у кібер-фізичних системах.

Останні роки розробка кіберфізичних систем (КФС) стає дедалі популярнішою. Під кіберфізичною системою розуміють комплексне поєднання кібернетичних компонентів та фізичних процесів, які дозволяють організувати виконання обчислювальних процесів, захищене збереження та обмін службовою та вимірювальною інформацією, здійснення впливів на фізичні процеси. Інтеграція таких компонентів в одну систему дає змогу створювати складні та ефективні технічні та сервісні інструменти [3, 109].

У той же час, важливою частиною кожної КФС є вимірювально–керувальний вузол (ВКВ). Таким чином КФС – це набір з ВКВ, які обладнані засобами спостереження та здатні взаємодіяти з фізичним середовищем виконуючи задачі [3, 110].

Для виконання задачі оперативного відстеження та наведення на рухомий об’єкт доречно використати один з модулів ВКВ у КФС. Такий модуль можна використовувати у системах з Unmanned Aerial Vehicles (UAV, Дрони) для наведення на об’єкт у реальному часі.

UAV впливають на наше суспільство багатьма способами. Тепер ми можемо виконувати завдання, які раніше були неможливі або вимагали значного втручання людини. Вони повністю змінили спосіб збору й обробки інформації про віддалені райони. Більшість дронів керуються людьми, але з розвитком ШІ, задачі керування, відстеження та наведення виходять на новий рівень [2 – 3].

Існує багато прикладних завдань, де UAV можуть відстежувати чи наводитись на об’єкти [111-115].

Від стеження за людиною, використання GPS-координат на величезних відстанях [111] до систем, які намагаються спонукати тварин грати [113] (Рис.

1.24), а також систем відстеження та виявлення літаків [114].

Рис. 1.24. Ліворуч – приклад системи відстеження дронів GPS (Global Positioning System) із використанням персонального мобільного пристрою як локатора. Праворуч – UAV система аналізу грайливої пози тварини.

Серед інших задач, які можуть бути використані у КФС, є системи дефектоскопії матеріалів з використанням мікроконтролера та крокового двигуна для обертання об’єкта дослідження та аналізу можливих виявлень дефектів на тестованих вробах. Як результат, система розпізнавання довільного класу об’єктів дозволяє ефективно оцінювати якість виробу у системах неруйнівного контролю виробів [115].

У той же час визначені системи КФС системи вимагають великих обчислювальних потужностей для задач РО, а у випадку використання ХП існує проблема затримок у мережі. Тому доречною є задача мінімізація обчислювальних ресурсів та зменшення залежності від доступу до мережі.

1.7. Постановка задачі

У ході аналізу літературних джерел, безумовно визначено методи та засоби реалізації поставлених задач.

Суть дисертаційного дослідження полягає у реалізації системи РО та ВО на МП. Враховуючи особливості МП, а саме обмеженість обчислювальних ресурсів, а також вразливість засобів зв’язку, необхідно реалізувати вбудоване рішення для задач РО та ВО.

Необхідно розробити методи ти алгоритми ефективного пошуку та розпізнавання об’єктів на такій платформі у реальному часі з використанням

відеозображень та інших засобів.

Також однією із задач є забезпечення масштабування систем навчання та висновування для таких пристроїв, в залежності від потреб користувачів.

Важливою задачею є реалізація дружнього інтерфейсу користувача для користування системою.

Систему необхідно апробувати на фізичному пристрої та протестувати результати її роботи.

Висновки до першого розділу.

У розділі проаналізовано існуючі типи галузі знань які стосуються пошуку та розпізнавання об’єктів: машинне навчання, глибинне навчання та комп’ютерний зір.

Класифіковано задачі розпізнавання об’єктів: розпізнавання локалізацію, та пошук. Проаналізовано алгоритми класифікації для задач розпізнавання: штучні нейронні мережі, логістичну регресію, баєсівський підхід, дерево ухвалення рішень та інші. Визначено, що штучні нейронні мережі показують точніші результати по усім метрикам.

Проаналізовано архітектуру штучних нейронних мереж. Виділено основні функції активації: Sigmoid, Leaky-Relu, Mish, Softmax.

Визначено два основні етапи навчання в штучних нейронних мережах: навчання та висновування. Висвітлено три основних типи навчання штучних нейронних мереж: навчання з вчителем, без вчителя та з підкріпленням. Описано методики передавального навчання, для модифікації результатів навчання.

Проаналізовано типи штучних нейронних мереж: багатошаровий перцептрон, згорткові нейронні мережі, рекурентні нейронні мережі. Розглянуто відомі алгоритми Згорткових нейронних мереж: R-CNN, Faster-R-CNN, SSD, Yolo та інші. Описано різновиди та модифікації моделі Yolo: Yolov3, Yolov4, YoloR, Yolov5 та інші.

Розглянуто основні підходи до інтеграції систем пошуку та розпізнавання на мобільні платформи. Проаналізовано основні види мобільних платформ: мобільні операційні пристрої та вбудовані системи. Описано методи інтеграції

методів пошуку та розпізнавання на мобільну операційну систему iOS за допомогою фреймворку CoreML.

Визначено засоби масштабування систем пошуку та розпізнавання. Описано типи таких систем в залежності від задач які покладені на споживача та провайдера: IAAS, PAAS, FAAS, SAAS та інші. Визначено, що PAAS система Docker найбільше підходить для вирішення задачі пошуку та розпізнавання для мобільних платформ.

Описано основі підходи до відстеження об’єктів у реальному часі на відеозображеннях. Визначено чотири типи алгоритмів відстежування з використання штучних нейронних мереж: відстеження шляхом пошуку, кореляційні фільтри, сіамські нейронні мережі та навчання з підкріпленням. Описано імплементації таких алгоритмів, визначення їх переваги та недоліки. Проаналізовано інтеграцію систем розпізнавання та відстеження у кіберфізичні системи.

З урахуванням проаналізованих матеріалів, посталено вимоги до створення мобільної платформи для пошуку та розпізнавання об’єктів у реальному часі.

У розділі здійснено аналіз літературних джерел [12-113].

РОЗДІЛ 2
МЕТОДИ ТА МОДЕЛІ РОЗПІЗНАВАННЯ ТА ВІДСТЕЖЕННЯ ДОВІЛЬНОГО КЛАСУ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ МОДЕЛІ ЗНМ YOLO

У розділі розглянуто метрики оцінювання результатів розпізнавання та відстеження об'єктів. Сформовано та описано загальну структуру моделі ЗНМ Yolov4 для МП. Використано модифікований метод кластеризації об'єктів розпізнавання на базі k-середніх++. Розроблено методи фільтрації результатів розпізнавання. Розроблено 3 типи алгоритмів відстеження об'єктів: алгоритмічний, алгоритмічний з навчанням з підкріпленням та алгоритм оперативного відстеження на базі мінімізаційного фільтру IOU. Розроблено методи мемоїзації об'єктів відстеження. Визначення метод квантизації вихідних вагових коефіцієнтів ЗНМ.

Результати розділу опубліковано в працях автора [1– 6].

2.1. Метрики оцінювання результатів розпізнавання та відстеження об'єктів.

Задачі розпізнавання.

Для задач РО з використанням навчання з учителем та незбалансованими класами застосовуються метрики матриці невідповідностей.

Це метод розбивання об'єктів розпізнавання на 4 категорії, в залежності від комбінації позитивності відповіді та алгоритму. До таких категорій відносять: істинно позитивні (TP), істинно негативні (TF), хибно позитивні (FP) та хибно негативні (FN) стани.

Введемо так званий мінімізаційний фільтр перетину регіонів розпізнавання [1].

$$\boldsymbol{IOU}(\boldsymbol{A},\boldsymbol{B})(\text{відношення перетинань та об}'\text{єднань}) = \frac{A \cap B}{A \cup B} \qquad (2.1)$$

Ця метрика визначає наскільки регіон розпізнавання та еталонний об'єкт співпадають по внутрішньому об'єму.

Результат розпізнавання вважається істинно позитивним (TP) виявленням, якщо IOU дорівнює або перевищує 0.5. FP – це такий стан, що має IOU зі значеннями нижче 0.5. FN це такий стан, при якому істинно позитивні об'єкти не були виявлені, або були нижчі за заданий поріг. FP у свою чергу визначає стан при якому хибно негативні результати інтерпретовані як позитивні.

Для оцінки результативності розпізнавання, застосовуються наступні метрики:

$$\boldsymbol{R}\ \left(\frac{recall}{\text{чутливість}}\right) = \left(\frac{TP}{TP+FN}\right) \tag{2.2}$$

$$\boldsymbol{FNR}\ (\text{істиннонегативний рівень}) = 1.0 - R \tag{2.3}$$

$$\boldsymbol{P}\ \left(\frac{precision}{\text{влучність}}\right) = \left(\frac{TP}{TP+FP}\right) \tag{2.4}$$

$$\boldsymbol{FPR}\ (\text{хибнопозитивний рівень}) = 1.0 - P \tag{2.5}$$

Чутливість вказує, яка кількість TP об'єктів була виділена класифікатором. Влучність вказує яка доля об'єктів, виділених як TP справді є істинною. FPR вказує очікувану тривалість FP стану. FNR – частка усіх FN, які все ще дають позитивні результати. Чим менші значення FNR та FPR, тим більша продуктивність моделі.

Варто відмітити, що чутливість та влучність не залежать від співвідношення розмірів вхідних класів. При умові що доля об'єктів TP значно менша за кількість об'єктів TF класу, дані показники метрики будуть показувати коректну роботу тестованих алгоритмів.

Існує 2 основних способи отримання одного критерія якості, використовуючи чутливість та влучність: F міра та показник середньої влучності.

$$\boldsymbol{F1}(F\ \text{міра}) = 2\ \times \left(\frac{P \times R}{P+R}\right) \tag{2.6}$$

Особливості отримання середньо гармонійного значення для F міри, полягає у тому що така міра близька до нуля. Таким чином досягається вища точність метрики, при не правильному розподілені вибірки.

У випадку необхідності не тільки передбачати клас об'єкта, а і виконувати ранжирування, тобто виконання задач РО пошуку та локалізації,

використовується метрика середньої влучності.

$$\boldsymbol{AP}\ (\text{середня влучність}) = \int_0^1 P(R)\ dR$$

Ефективність вибірки вимірюється шляхом вивчення як показників чутливості так і показників влучності, при аналізі продуктивності певних категорій.

Для тестованої вибірки Pascal VOC, формула набуває наступного вигляду:

$$\boldsymbol{AP_{Pascal\ VOC}} = \frac{1}{11}\sum_{Ri} P \times R_i$$

при умові IOU ≥ 0.5 значення вважається TP, та при локалізації об'єктів однакового класу, перший розпізнаний вважається TP, решта – TF. В такій вибірці для обрахунку середнього значення використовується обрахунок 11 точкової інтерполяції середнього значення влучності.

У той же час для потреби обрахунку середніх класифікаційних показників для всіх категорій, використовують метрику mAP [1]:

$$\boldsymbol{mAP}\ (\text{середня вибіркова влучність}) = \frac{\sum_{i=1}^{n} AP_i}{n} \qquad (2.7)$$

Така метрика визначає рівень впевненості (confidence) для об'єктів розпізнавання. Тому такий метод доречно використовувати на етапі тренування ЗНМ, для оцінки ефективності навчання. Також її можна застосувати для порівняння реалізованих моделей ЗНМ Yolov4 на етапі висновування.

Важливо відзначити, що для виконання задач на МП, які мають обмежені апаратні можливості, варто додати метрику продуктивності завантаження GPU та CPU у реальному часі. Введемо поняття які визначають суть реального часу: FLOPs (кількість операцій з рухомою комою за 1 секунду), FPS (кількість кадрів на секунду), час відображення результатів на екрані після початку опрацювання $\boldsymbol{t_{frame}}$, медіана часу для виконання операції передбачення об'єкта $\boldsymbol{t_{predict}}$, медіана часу для завантаження моделі у тестований пристрій $\boldsymbol{t_{load}}$.

Отже, для досягнення цілей дослідження, необхідно:

- Визначити швидкість операції висновування розробленої ЗНМ на

МП для задач РО. Швидкість має бути близькою до реального часу, що рівно приблизно 24 FPS;

- Визначити використання CPU, GPU та інших пристроїв акселерації з використанням метрик FLOPs, $\boldsymbol{t_{predict}}, \boldsymbol{t_{frame}}, \boldsymbol{t_{load}}$.

Підсумовуючи, важливо забезпечити ефективність та точність розроблених методів та засобів для задач РО на МП у реальному часі.

Задачі відстеження.

Задачі багатоцільового відстеження (МОТ) полягають у відстеженні шляху об'єктів у відеозображення у реальному часі. Така задача має за собою ціль відстежувати регіони розпізнавання кожного об'єкта в продовж обробки одного кадру. Така задача є нетривіальною, оскільки є ймовірність отримати одразу декілька істинних регіонів розпізнання будуть перекриватись і підходити одразу до декількох гіпотез відстежування.

Для оцінки якості виконання задач відстеження об'єктів, доречно ввести наступні метрики: МОТА (Багатоцільова точність / Multiple Object Target Accuracy) та МОТР (Багатоцільова влучність / Multiple Object Tracking Precision (МОТР) [116].

$$\boldsymbol{MOTP} = \frac{\sum_f \sum_{n,m} d_{tnm}}{\sum_f TP_f} \tag{2.8}$$

$$D \in \mathbb{R}^{N \times M}, d_{nm} \in [0,1]$$

МОТР метрика використовується для оцінки влучності позиціювання розпізнаних об'єктів. Це загальна кількість опрацьованих позиційних помилок d для усіх кадрів f виконаних за час t. Де d_{nm} – це матриця відстані фундаментальної істини для об'єктів відстеження (n,m). Для задач РО таку відстань обраховують використовуючи мінімізаційний фільтр IOU. МОТР вказує здатність алгоритму відстеження точно відстежувати позицію відстеженого об'єкта, незалежно від якості такого алгоритму до розпізнавання.

$$\boldsymbol{MOTA} = 1 - \frac{\sum_f (FP_f + FN_f + MME_f)}{\sum_f TP_f} \tag{2.9}$$

МОТА метрика базується на кількості пропущених об'єктів (FP_f), кількості

хибних спрацювань (FN_f) та частоті помилок призначення ідентифікаторів (MME). Кожен раз, коли фундаментально істинний об'єкт змінюється, збільшується коефіцієнт MME з зміною структури призначення відстежених об'єктів [116].

Чим більше значення t, тим менша кількість відстежуваних об'єктів. Таким чином, чим більший період роботи метрики, тим більша кількість FP і зменшується кількість FN, MME. В граничних випадках жоден з об'єктів не буде розпізнаний, оскільки FP буде рівно 100%, а FN=MME=0%. Тому доречно використовувати обидві метрики MOTP та MOTA для порівняльного аналізу відстежуваних об'єктів у реальному часі.

2.2. Опис моделі розпізнавання об'єктів Yolov4.

Процес РО при використанні ЗНМ можна розділити на наступні фази (Рис. 2.1):

- Хребет (backbone) ЗНМ – нейронна мережа для виділяння ознак з вхідного зображення, виконуючи аналіз лише узагальненої інформації високого рівня. Популярним рішенням слугують залишкові нейронні мережі (ResNet);
- Шия (neck) ЗНМ – окремий блок нейронної мережі необхідний для агрегації інформації від окремих шарів з попередніх блоків. Можливі приклади реалізації це: Path Aggregation Network (PAN), Feature Pyramid Network (FPN) та Spatial Pyramid Pooling (SPP);
- Голова (head) ЗНМ – блок для отримання виділених ознак з хребта ЗНМ, та обробки результатів враховуючи показник втрат (loss). Використовується для задач РО: пошуку та локалізації. Як зазначалось у літературному розділі, виділяють як двохрівневі (Sparse Prediction), які виділяють по окремості регіони розпізнавання і по черзі класифікують, так і однорівневі класифікатори (Dense Prediction). До таких класифікаторів відносять такі ЗНМ як Faster RCNN та Yolo відповідно.

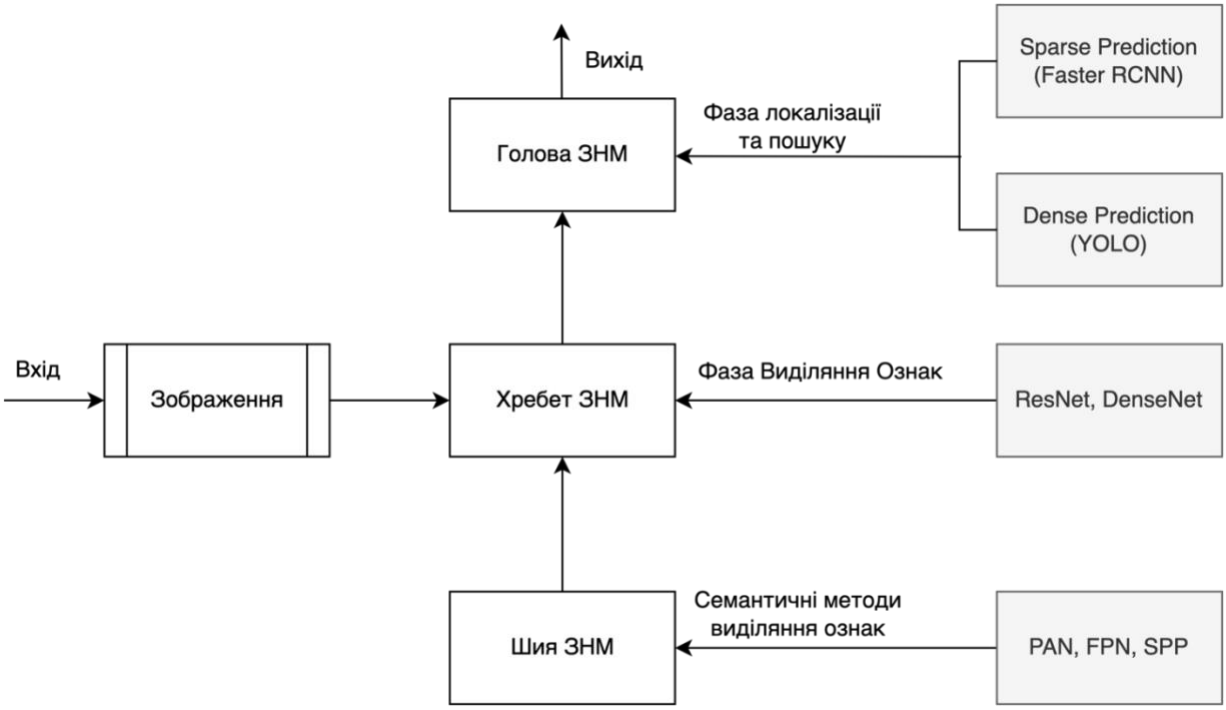


Рис. 2.1. Схематичне представлення фаз РО.

В ході аналізу літературних джерел було обрано архітектуру ЗНМ Yolov4.

Загалом, вибираючи підходи архітектури таких ЗНМ як Yolov4, важливо визначити критерії по яким виконувати вибірку.

Серед таких критеріїв можна виділити: роздільну здатність ЗНМ (розмір вхідного зображення); кількість згорткових шарів; кількість вхідних параметрів; кількість вихідних шарів (фільтрів); методи для збільшення рецептивного поля (bag of specials); методи реалізації хребтів та ший ЗНМ.

Розглянемо кожну з фаз РО окремо для архітектури моделі Yolov4.

Хребет ЗНМ. Хребет Yolov4 складається з декількох блоків з'єднання Cross Stage Partial (CSP) (Рис. 2.2). Такий метод полягає у додаванні обгортки на залишковою нейронною мережею (ResNet) та розподілення потоку градієнтів по різним шарам ЗНМ. Таким чином розподілення може мати більшу кореляцію, при перемиканні фаз конкатенації та переходу.

Підхід залишкових нейронних мереж полягає у пропуску декількох наступних шарів (з використанням подвійного чи потрійного пропуску або використані функцій активації). Такі пропуски необхідно інтегрувати для мінімізації проблеми зникання градієнту та запобігання проблеми виродження.

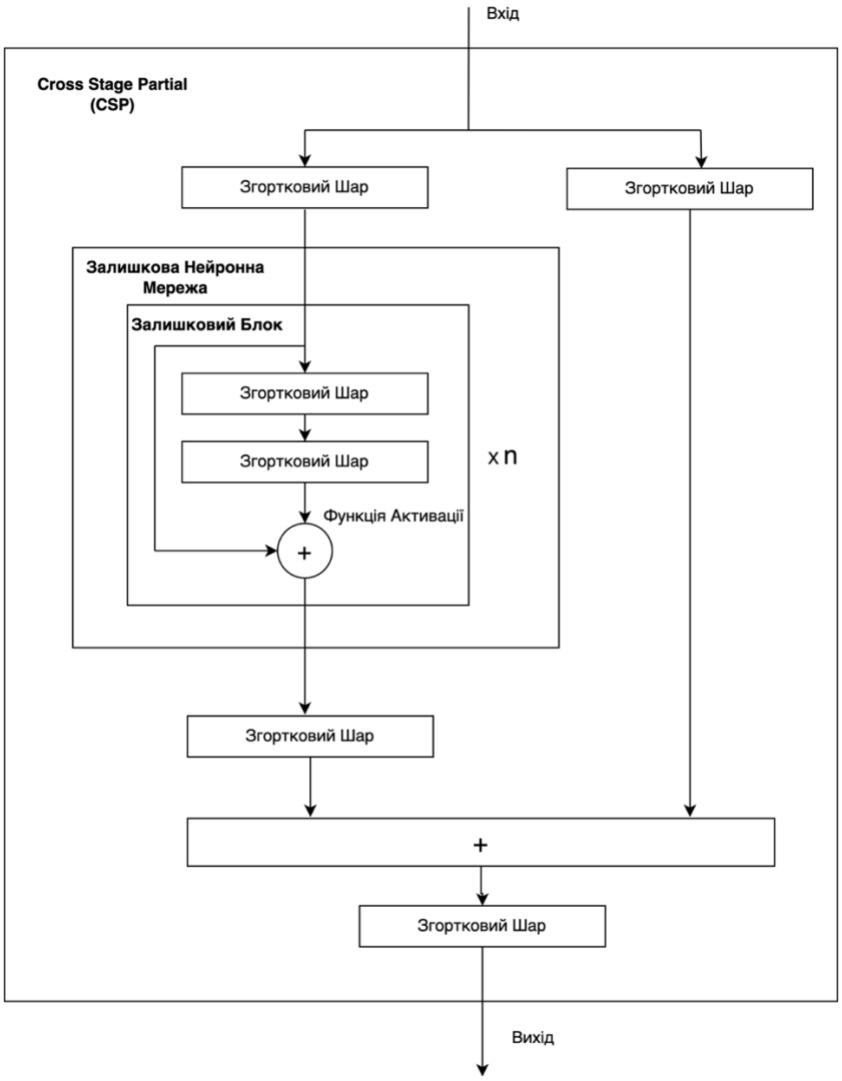


Рис. 2.2. Структурна схема CSP.

Вибираючи найоптимальнішу модель хребта ЗНМ, варто опиратись на такі параметри як: вхідна роздільна здатність ЗНМ; розмір рецептивного поля, кількість параметрів; BFLOP’s (мільярди операцій FLOPS); FPS. Продемонструємо аналіз хребтів ЗНМ, які використовують CSP модуль (таблиця 2.1).

Таблиця 2.1. Параметри хребтів ЗНМ для задач РО

Хребет ЗНМ	**Роздільна здатність ЗНМ**	**Розмір рецептивного поля**	**Параметри**	**BFLOP’s**	**FPS**
CSPResNext50 [117]	416x416	425x425	19.8 M	30	61
CSPDarknet53 [54]	416x416	725x725	24.1 M	53	66
CSPDarknet53 -tiny [54]	416x416	325x325	12.3 M	18	132

Для тестової вибірки вибрано ЗНМ з роздільною здатністю 416x416 пікселів, оскільки такий розмір зображень достатній для задач РО на МП. В загальному, чим більше вхідних параметрів чи BFLOP’s означає менший показник FPS. Також вимогою є висока роздільна здатність рецептивного поля для обробки роздільної здатності ЗНМ. Підсумовуючи, чим більше вхідних параметрів, тим більше буде можливостей розпізнати об’єкти на одному зображенні.

За результатами аналізу, виявлено що CSPResNext50 має вищу точність при виконанні задачі класифікації зображень, проте при розпізнаванні об’єктів ефективнішими є CSPDarknet53 та його підвид CSPDarknet53-tiny.

Розглянемо детальніше архітектури хребта ЗНМ Yolov4 CSPDarknet53 та CSPDarknet53-tiny. Це трьохшарова та двохшарова ЗНМ відповідно. Структурні схеми таких ЗНМ наведено на Рис. 2.3 та Рис. 2.4.

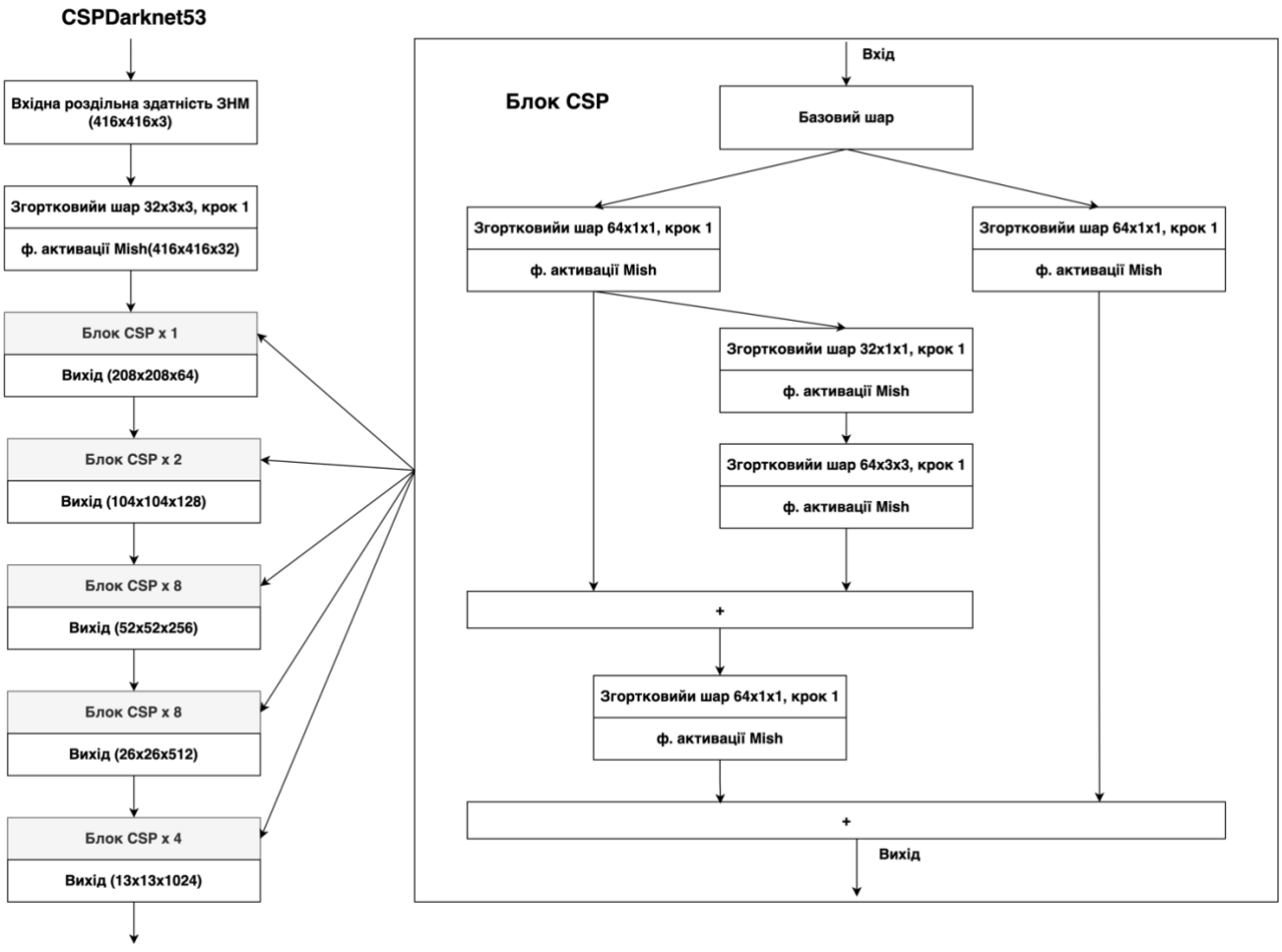


Рис. 2.3. Структурна схема хребта ЗНМ CSPDarknet53 Yolov4.

Модулі CSPDarknet53 можна розділити на дві групи: згорткові блоки та та CSP блоки. Згортковий блок складається з згорткових шарів, розмірність яких вимірюється висотою та шириною зображення та кількістю каналів, наприклад RGB.

Додатково використовується кернел (фільтр) для виконання операції згортки згорткового шару розміром 3 з кроком (stride) 1. Як функція активації використовується Mish. CSP блок розділяє карту ознак базового шару ЗНМ на дві частини, опрацьовує їх та об’єднує їх через міжфазову ієрархію, розділюючи градієнтний потік.

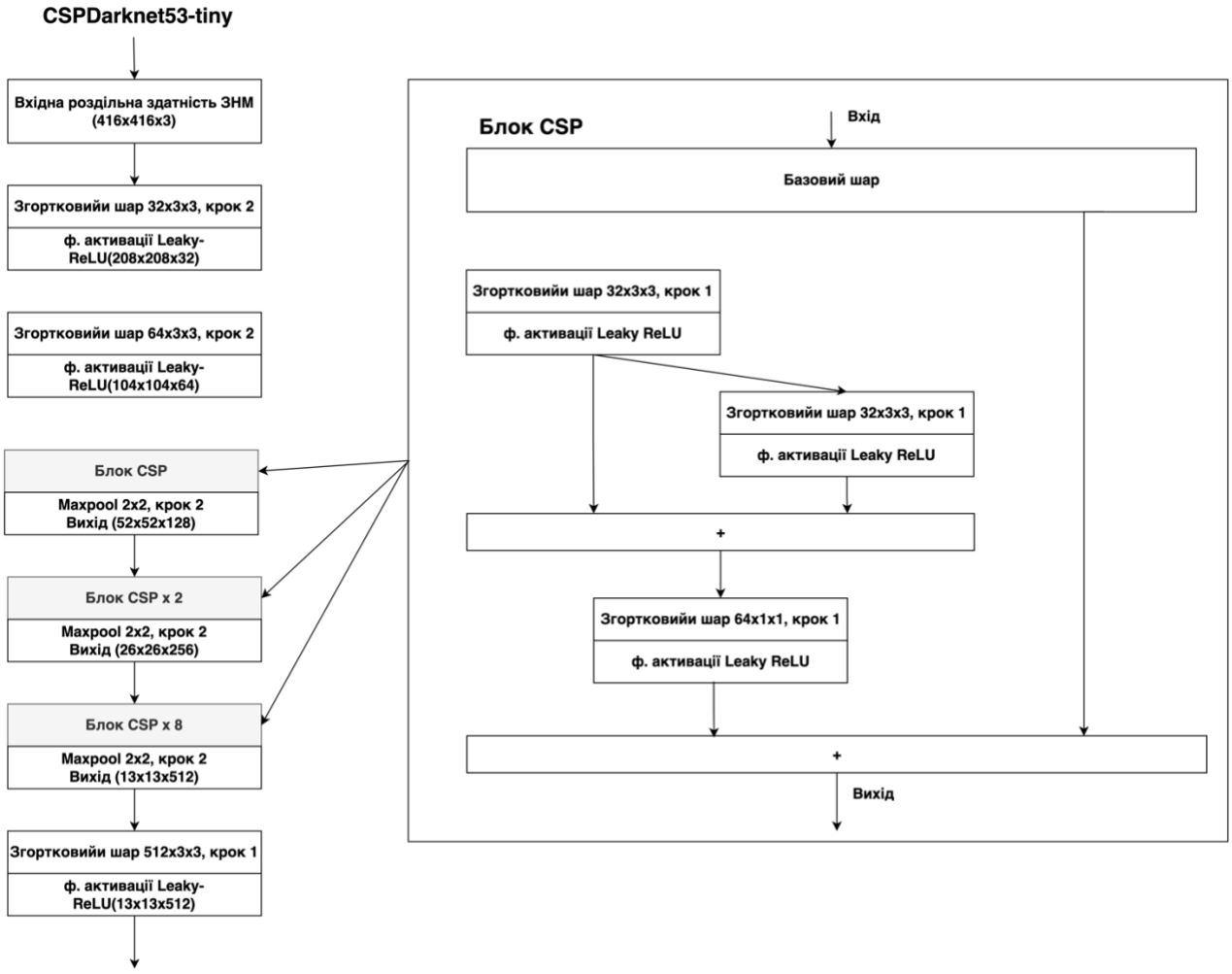


Рис. 2.4. Структурна схема хребта ЗНМ CSPDarknet53-tiny Yolov4.

Для задач РО на МП вкрай важливо зменшити апаратні вимоги до пристрою, при цьому критично не зменшуючи результати точності розпізнавання.

CSPDarknet53-tiny це полегшена версія ЗНМ CSPDarknet53, з двома вихідними шарами. Як функція активації, доречно використати функцію активації Leaky ReLU, оскільки при меншій вихідній точності, на ЗНМ зменшується апаратне навантаження.

Шия ЗНМ.

Шия ЗНМ Yolov4 реалізована за допомогою технології Path Aggregation Netowork (PAN) [118]. Цей метод полягає у агрегації параметрів з різних шарів хребтів ЗНМ для різних шарів голови ЗНМ на фазі висновування з використанням адаптивних карт ознак та аугментації шляхів. Аугментація скорочує інформаційний шлях, роблячи доступною локалізовані об’єкти на верхніх шарах ЗНМ (класифікаторі). У той же час адаптивна карта ознак відновлює втрачену інформацію між кожним прогнозованим значенням та шарам ознак. Таким чином PAN забезпечує цілісність ознак на етапі навчання ЗНМ.

Голова ЗНМ.

Задачі пошуку та локалізації в ЗНМ yolov4 виконується на останньому шарі такої ЗНМ. Приклад реалізації моделі пошуку та локалізації наведений на Рис. 2.5.

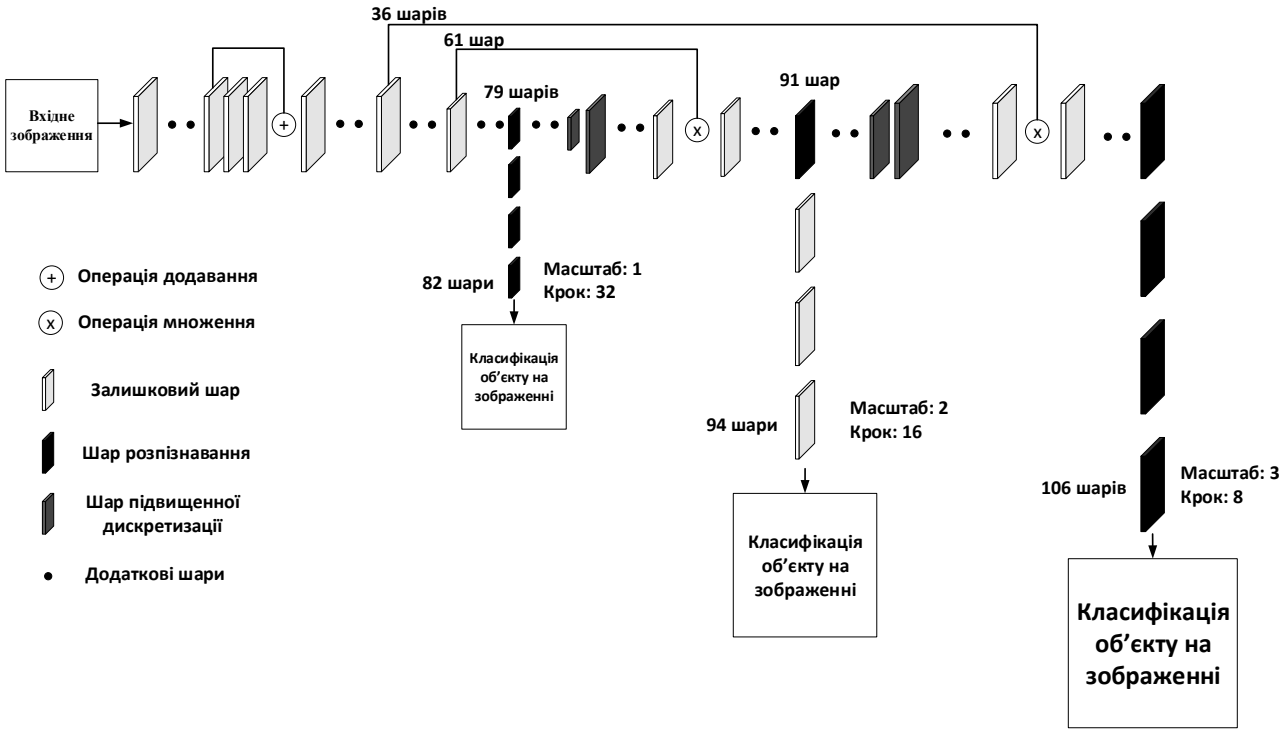


Рис. 2.5. Структурна схема голови ЗНМ CSPDarknet53 Yolov4 [1, 6].

На прикладі ЗНМ зазначено 3 вихідні шари, кожному з них передбачаються регіони для різних об'єктів у вигляді двомірного масиву. Розміри комірок масиву мають наступні значення: 8, 16 і 32.

На вхід ЗНМ подається вхідне зображення, розмірністю 416x416 пікселів. В результаті, вихідні матриці (сітки) матимуть наступні розміри: 52x52 , 26x26 та 13x13 (416/8 = 52, 416/16 = 26 і 416/32 = 13).

При запуску моделі ЗНМ, на виході генерується наступний вектор: *[1, 1, 18, 25, 25]*. Де 18 – це значення, яке отримується за допомогою формули [6]:

$$\overline{\boldsymbol{V}} = L \times \left(\left(t_x + t_y + t_w + t_h + p_o\right) + \left(p_1 + p_2 + .. + p_n\right)\right)$$

Де, $\overline{\boldsymbol{V}}$ – результуючий вектор;

L – вихідні шари мережі (регіони);

t_x, t_y, t_w, t_h – координати x, y, ширина та висота передбачуваного регіону;

p_o – коефіцієнт передбачення появи регіону (згладжуючий фільтр Object confidence). Визначає поріг знаходження об’єкта всередині передбачуваного прямокутника розпізнавання;

$p_1 .. p_n$ – вхідні класи, де n – кількість.

Наступним етапом є обробка отриманого вектора $\overline{\boldsymbol{V}}$. Для кожного передбачуваного регіону потрібно отримати розподілення вірогідності для заданого діапазону класів. Зробити це можна зробити за допомогою логістичної функцій softmax, для багатомірного випадку тобто для 2 та більше класів:

$$\boldsymbol{S}(\boldsymbol{z_i}) = \frac{exp^{z_i}}{\sum_{n=1}^{N} exp^{z_n}}$$

$$\boldsymbol{z} = w^T x - \theta$$

Де

z – вхідний вектор;

N – кількість класів;

M – кількість вхідних ознак;

$\boldsymbol{w^T}$**–**транспонована матриця вагових коефіцієнтів вхідних ознак розмірністю K * M;

$\boldsymbol{\theta}$ – вектор з пороговими значеннями розмірності N.

Для отримання координат і розмірів регіонів розпізнавання, необхідно скористатися формулами [6]:

$$\boldsymbol{b_x} = s(t_x) + c_x$$

$$\boldsymbol{b_y} = s(t_y) + c_y$$

$$\boldsymbol{b_w} = p_w exp^{tw}$$

$$\boldsymbol{b_h} = p_h exp^{th}$$

$$\overline{\boldsymbol{Detection}} = (t_x, t_y, t_w, t_h, t_o, c_{0,1\ldots n})$$

$$\boldsymbol{p_o} = IOU_b^{truth}\overline{Detection}$$

де b_x, b_y, b_w, b_h – передбачені координати x, y та ширина і висота регіону розпізнавання;

t_x, t_y, t_w, t_h виходи нейронної мережі, з об’єктом розпізнавання;

s – функція активації (у yolov4 використовується функція сигмоїди);

c_x та c_y – верхні крайні ліві початкові координати сітки;

p_wта p_h – значення висоти та ширини якорів розпізнавання для заданих регіонів. У даному випадку їх 3.

Після отримання координат і розмірів регіонів і відповідні ймовірності для всіх знайдених об'єктів на зображенні ($\overline{Detection}$), можна починати відмальовувати їх поверх картинки.

2.3. Метод кластеризації об’єктів розпізнавання.

Якорі розпізнавання (anchor boxes) – це методика розбиття вхідного зображення на сітку клітин, до яких прикріпляється об’єкт регіон розпізнавання, визначаючи позицію, ширину та висоту по відношенню до центру клітини сітки. Якір – це методика визначення таких координат. На Рис. 2.6. схематично показано визначення розмірів якорів, регіонів розпізнавання та розміри клітини сітки вхідного зображення.

Якорі розпізнавання задаються або вручну базуючись на наборі вхідних даних зображень, або ж виконуючи автоматичне визначення розмірів, базуючись

з розмірів регіонів розпізнавання. Yolov4 використовує метод к-середніх для виділення найоптимальніших розмірів якоря розпізнавання. Для мінімізації надлишкових якорів доречно використати мінімізаційний фільтр IOU (формула 2.1).

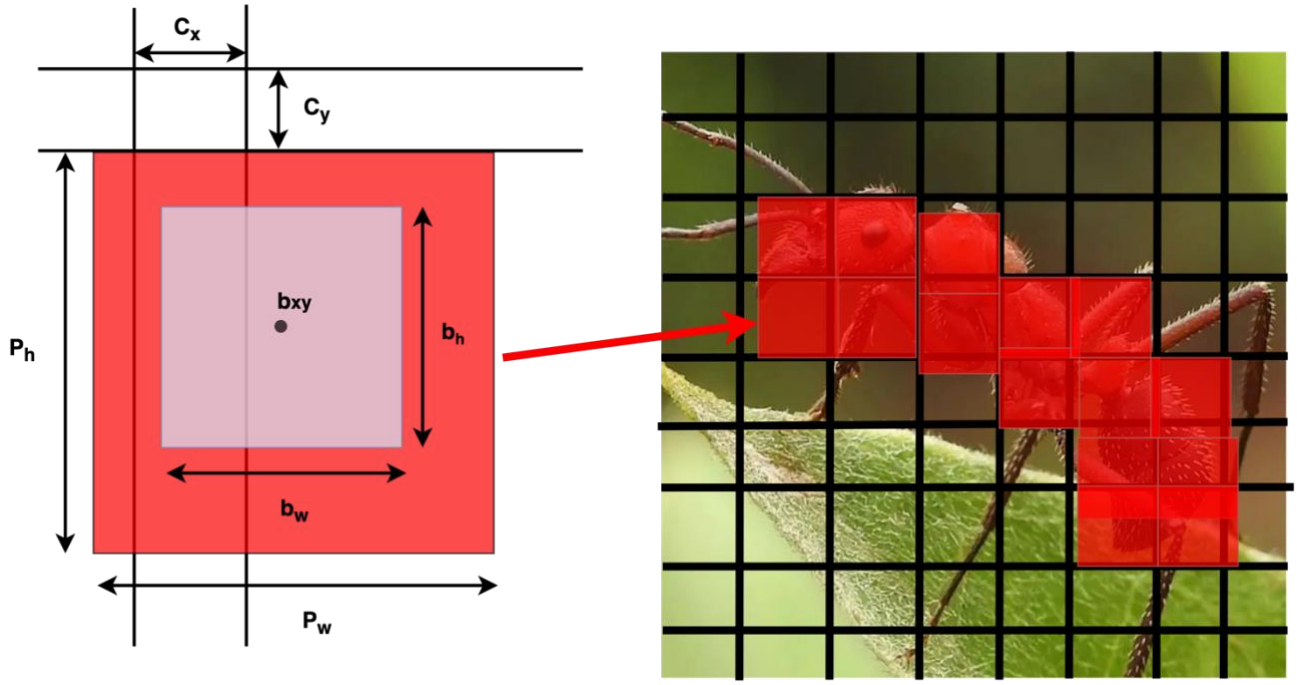


Рис. 2.6. Схематичне представлення генерації якорів та регіонів розпізнавання для розпізнаного класу об’єкта на вхідному зображенні.

Отриманий результат точності b_{xy} показує точність передбачення класу об’єкту для певного регіону.

При автоматичному визначенні якорів розпізнавання Yolov4 використовує метод кластеризації з навчанням без учителя k-середніх.

Суть кластеризації для задач РО у ГН полягає у виділянні спільних по розташуванню та кольору на зображені регіонів. Процес виділянні таких регіонів називається кластеризацією.

Принцип роботи k-середніх полягає у ітеративному використанні евклідової відстані для обрахунку відстані між вручну заданою кількістю кластерів. Така відстань на етапі очікування обраховується як відстань від початкового центру кластера (центроїда) до центру кожного об’єкта. Наприклад, для двох кластерів така відстань може прийняти вигляд:

$$\sqrt{(A_2 - A_1)^2 + (B_2 - B_1)^2}$$

де A_1, A_2 та B_1, B_2 – точки в координатній сітці кожного кластера.

При цьому на етапі максимізації початковий центр зміщається по координатам в сторону найменшої можливої відстані (алгоритм 2.1).

Алгоритм 2.1. Алгоритм K-середніх.

1. Визначення кількості кластерів **k;**
2. Випадкове розташування **k** центроїдів;
3. Повторити;
4. **Очікування**: визначення координат кожного об'єкту до найближчого центроїда k;
5. **Максимізація**: визначення нових координат центроїда k для кожного класу;
6. **Допоки** координати центроїда не змінюється.

Недоліком такого алгоритму є потреба завчасно знати кількість кластерів; значення результату кластеризації та часу виконання (O(n)) залежить від вибору початкових центроїдів. Якщо вибірка початкових центроїдів для кожного кластеру є повністю випадковою, це може призвести до помилок збіжності.

Тому для виконання задачі генерації якорів розпізнавання пропонується використати модифікований алгоритм k-середніх++ [119], який покладений вирішити проблему випадкового розташування центроїдів.

Суть алгоритму полягає у пріоритизації точок, які знаходяться на максимально далекій відстані від центроїда, для уникнення ситуації перекривання двох точок (алгоритм 2.2).

Алгоритм 2.2. Алгоритм K-середніх++.

1. Визначення першого центроїду випадковим чином серед усіх точок **X**;
2. Повторити;
3. Для кожної нової точки обрахувати відстань **D** з ймовірністю **P**;

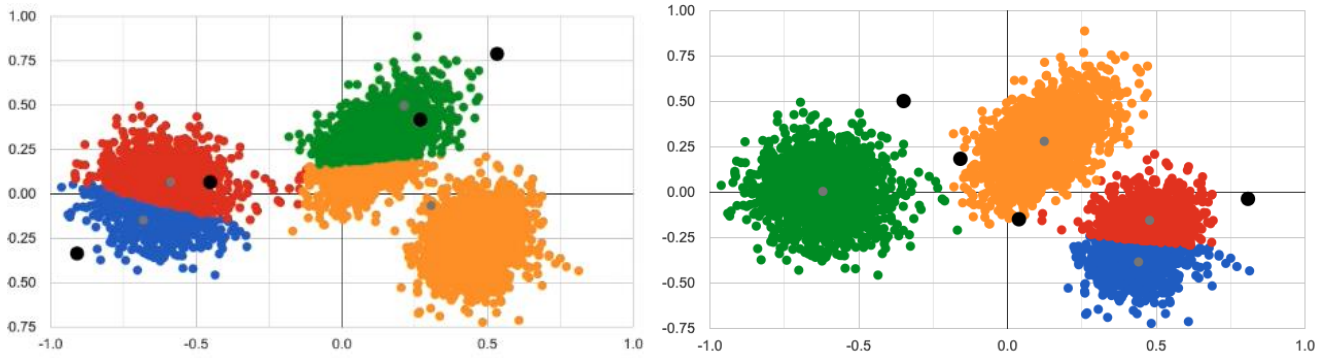
$$x \in X, \qquad P_i = \frac{D(x_i)^2}{\sum_{i=1}^{n}(x_i)^2}$$

4. Вибір наступного центроїда з набору точок таким чином, щоб ймовірність

ймовірність вибору точки у якості центроїда була прямо пропорційна до її евклідової відстані до найближчого попередньо вибраного центроїда;

5. **Допоки** усі центроїди не будуть знайдені;
6. Виконання алгоритму k-середніх (алгоритм 2.1).

Для порівняння якості кластеризації алгоритмів к-середніх та к-середніх++ було згенеровано набір випадкових даних у кількості 2000 точок з координатами $X,Y \in [-1;+1]$. Для прикладу, згенеровано 4 кластери. Результати продемонстровані на Рис. 2.7.



(а) (б)

Рис. 2.7. Результат генерації алгоритму к-середніх (а) та к-середніх++ (б) для 4 кластерів. Чорні точки – початкові координати центроїдів. Сірі точки – центральні координати кластерів.

Як видно з результатів аналізу, при поточній організації початкових центроїдів (а), кластеризація пройшла неуспішно, оскільки дані з одного кластеру перериваються з даними іншого кластеру. Разом з тим, для алгоритму к-середніх++ (б), точки в кластерах розподілені коректно.

Варто відзначити, що при більших обчислювальних затратах у порівнянні з алгоритмом к-середніх, помилки при збіжності значно мінімізуються.

Для задач РО при формуванні якорів розпізнавання, алгоритм к-середніх++ збільшує точність вибору якорів, що може значно покращити визначення помилки при класифікації на зображенні.

Для досліджуваної моделі Yolov4 з 3 вихідними шарами, методика кластеризації за допомогою алгоритму к-середніх++ виглядатиме наступним чином (алгоритм 2.3).

Алгоритм 2.3. Поліпшена методика генерації якорів розпізнавання для Yolov4 з використанням к-середніх++.

1. Визначення висоти та довжини прямокутників розпізнавання з усіх регіонів розпізнавання. Визначення початкової ітерації i = 0;
2. **Повторити** ітерацію i;
3. Вибір якоря розпізнавання як початкової точки (центроїду) вхідного кластеру з усіх регіонів розпізнавання
4. Обрахування відстані $D(x_i)$ між центроїдом всіх регіонів розпізнавання та центроїдом існуючих якорів розпізнавання. Вирахувати ймовірність $P(x_i)$ для кожного регіону розпізнавання, які були обрані як наступні центроїди:

$$x \in X, \qquad P_i = \frac{D(x_i)^2}{\sum_{i=1}^{n}(x_i)^2}$$

   Чим далі регіон розпізнавання від від початкового центроїду, тим більша ймовірність його вибору;
5. Використання мінімізаційного фільтра IOU для кожного регіона та якоря розпізнавання для вибору найбільш ймовірних якорів розпізнавання для поточного регіону розпізнавання. Чим вище це значення тим ймовірніше об’єкт відноситься до шуканого класу;
6. **Допоки** регіони розпізнавання не змінюються;
7. Отримання фінальних якорів розпізнавання.

Для меншої кількості вихідних шарів ЗНМ алгоритм виглядатиме аналогічно.

2.4. Методи фільтрації результатів розпізнавання.

Як визначено у пункті 2.2, за результатом розпізнавання, генерується набір розпізнаних регіонів об’єктів певних класів. Для збільшення якості розпізнавання, доречно застосувати методику фільтрації вихідних результатів розпізнавання.

Згладжуючий фільтр стиснення (ЗФС)

При умові, коли до одного об’єкта присвоюються одразу декілька регіонів розпізнавання з високою ймовірністю $\boldsymbol{p_o}$, виникає проблема нагромадження прямокутників з об’єктами на зображенні. Тому важливо максимально стиснути

кількість результатів розпізнавання для конкретного об’єкту до мінімуму (Рис. 2.8).

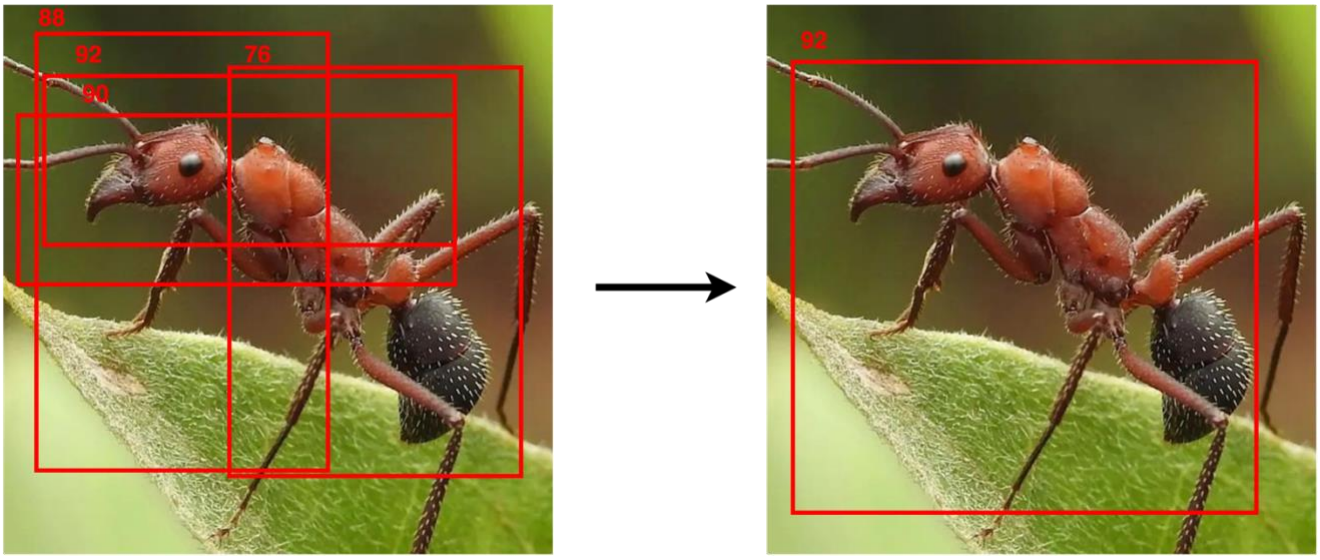


Рис. 2.8. Використання ЗФС для зменшення кількості регіонів розпізнавання.

Для обрахунку ЗФС необхідно: визначити регіони розпізнавання з найбільшою ймовірністю для поточного об’єкта; виконати фільтрацію при значенні мінімізаційного фільтру IOU (формула 2.1) для першого на черзі регіону більше заданого порогу (алгоритм 2.4).

Алгоритм 2.4. Методика фільтрації ЗФС.

1. Визначити поріг confidence threshold **T**;
2. Визначити список згладжених регіонів розпізнавання **Result** рівним 0;
3. Отримати відсоток розпізнавання **PR** для набору регіонів розпізнавання $\boldsymbol{Boxes}$;
4. Ітерація по $\boldsymbol{Boxes}$;
5. **Виконати** ітерацію по регіону розпізнаваня $\boldsymbol{b_i} \in \boldsymbol{Boxes}$;
6. Визначити прапорець фільтрації $\boldsymbol{f}$ рівним **0**
7. Наступна ітерація по $\boldsymbol{Boxes}$;
8. **Виконати** ітерацію по регіону розпізнаваня $\boldsymbol{b_j} \in \boldsymbol{Boxes}$ для кожного $\boldsymbol{b_i}$;
9. Якщо значення $\boldsymbol{IOU_{b_i}}$ рівне $\boldsymbol{IOU_{b_j}}$ та значення є більшими за T;
10. Якщо значення $\boldsymbol{PR_{b_j}}$ більше $\boldsymbol{PR_{b_i}}$;

11. Визначити прапорець фільтрації $\boldsymbol{f}$ рівним **1;**
12. Завершити ітерацію $\boldsymbol{b_j}$;
13. Якщо f є рівним 0
14. Збереження $\boldsymbol{b_i}$ у список **Result;**
15. Завершити ітерацію $\boldsymbol{b_i}$;
16. Отримати фіналізований список потенційних регіонів розпізнавання **Result**.

Задача алгоритму ЗФС – максимально звузити набір вихідних регіонів розпізнавання відкидаючи найменш ймовірні. Визначення максимального порогу confidence threshold *T та порогу мінімізаційного фільтру IOU* визначається користувачем системи.

Принцип мінімізаційного фільру IOU полягає у визначенні наближення розпізнаного об'єкта до істино позитивного еталонного регіону з вибірки (згенерованого або вручну, або за допомогою методів кластеризації) (Рис. 2.9).

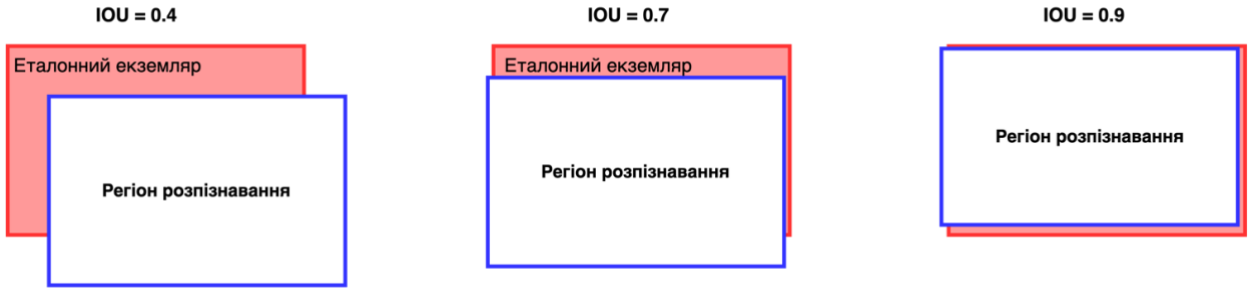


Рис. 2.9. Принцип збіжності регіонів розпізнавання за допомогою мінімізаційного фільтру IOU.

Варто відзначити що такі методи фільтрації як ЗФС виділяють регіони розпізнавання для кожного визначеного класу об'єкта по окремості. Проте виникає проблема, коли декілька регіонів розпізнавання певних класів можуть не попасти у фінальну вибірку Result, через зависокий пороги T та IOU**.** Тоді потенційно ймовірнісні регіони розпізнавання з високим відсотком розпізнавання можуть бути вилучені.

У такому випадку доречно залишати потенційні регіони розпізнавання у вибірці, зменшуючи їх відсоток розпізнавання пропорційно до значення порогу IOU. Таку задачу можна розписати як лінійне рівняння:

$$f = 1, b_i \in Boxes, \qquad Result = b_i - IOU$$

У такому випадку такі регіони розпізнавання зберігаються, з присвоєнням меншої кількості відсотку розпізнавання. При цьому ці регіони можуть брати участь у наступних ітераціях для інших об’єктів цього класу.

Порогові фільтри IOU та Object Confidence [1]. Як зазначалося вище, використовуються для відображення відфільтрованих регіонів розпізнавання на кадрі відеозображення за допомогою ЗФС. Чим менше значення порогу IOU, тим зменшується можливість накладань регіонів розпізнавання. Object Confidence визначає граничну межу ймовірності розпізнавання конкретного об’єкта.

Фільтр розміру кадру (ФРК) [5]. Для задач РО на МП з обмеженою роздільною здатністю екрану важливо обмежити формування завеликих прямокутників розпізнавання які виходять за піксельну сітку. ФРК фільтрує об’єкти розмір яких у відсотковому відношенні вище за розмір екрану. Обраховується за формулою:

$$(detection_w \times detection_h) \leq \left(\frac{frame_w \times frame_h}{area_{koef}}\right) \times 100$$

де:

detection$_{wh}$ – роздільна здатність (висота та ширина) розпізнаного прямокутника (регіону) розпізнавання;

frame$_{wh}$ – вхідна роздільна здатність (висота та ширина) з оптичного сенсора;

area_koef – коефіцієнт площі який вказує поріг вилучення розпізнаних об’єктів.

Загалом варто відзначити що усі фільтри застосовуються на фінальній стадії розпізнавання – висновування.

2.5. Методи оперативного відстеження розпізнаних об’єктів.

2.5.1. Алгоритмічний метод відстеження та наведення на об’єкт.

Як визначено в аналітичному розділі, для задач відстеження та наведення використовують модуль ВКВ. Такий алгоритм часто використовують як частина КФС з використанням UAV (дронів). Алгоритм відстеження доцільно розділити

на дві фази: обрахунок відстані від центру розпізнавання до об’єкта та обрахунок кута до локалізованого об’єкта. Схематична репрезентація таких підходів до відстеження та наведення на об’єкт наведена на Рис. 2.10.

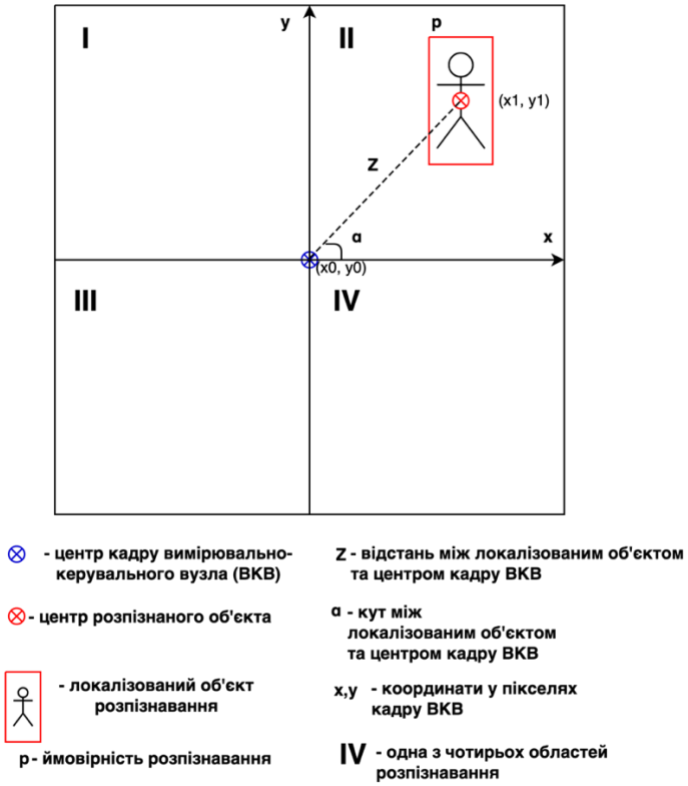


Рис. 2.10. Діаграма активності для алгоритму відстеження та наведення на рухомий. об’єкт. Визначення положення об’єкта відносно області розпізнавання за допомогою кута *a*.

Відстань між центром інформаційно кадру ВКВ та розпізнаним регіоном розпізнавання можна обрахувати за формулою [2]:

$$hypot = \sum_{i=1}^{\sqrt{n}} v_i^2 = \sqrt{v_1^2 + v_1^2 + .. + v_n^2}$$

$$d_x = A_X - B_X \quad d_y = A_y - B_y$$

$$dist = hypot(d_x, d_y)$$

де:

v – вхідний вектор;

$A_{x,y}$ – координати центру інформаційно кадру ВКВ;

$B_{x,y}$– координати центру регіону розпізнавання;

$d_{x,y}$ – координати для обрахунку відстані між $A_{x,y}$ та $B_{x,y}$;

hypot – функція, гіпотенуза або квадратний корінь суми добутків вхідних векторів;

dist – результуючий вектор. Відстань між двома точками у картезіанський системі координат. Відстань між центром ВКВ та об'єктом розпізнавання.

Проте виникає проблема обрахунку коефіцієнтів відстаней до об'єкту, оскільки для різних екранів користувачів з використанням різних пропорцій можуть виникати різні результати.

Для визначення цих пропорцій необхідно обрахувати значення, яке представляє собою відсоток площі, який займає прямокутник розпізнавання об'єкту по відношенню до розміру екрану. Це значення обраховується за формулою [2]:

$$area_P = \left(\frac{area}{frame_h \times frame_w}\right) \times 100$$

де:

$area$ – прямокутник з об'єктом розпізнавання;

$frame_h$ – висота екрану;

$frame_w$– ширина екрану;

$area_P$ – відсоткове співвідношення розмірів екрану пристрою та розмірів прямокутника з об'єктом розпізнавання.

Звідси отримуємо алгоритм руху дрона по траєкторії назад та вперед:

- Якщо значення *area* менше значення screen на 25%, рухаємо дрон уперед;
- Якщо значення *area* більше значення screen на 25%, рухаємо дрон назад;
- При умові що *area* між 30% та 60% значення screen – дрон не рухається.

Варто відзначити, що результуючий вектор *dist* використовується як допоміжний метод для визначення напрямку наведення модуля ВКВ. Такий вектор повинен прагнути максимально наближатись до центру інформаційного кадру ВКВ.

Узагальнена структурна схема алгоритму відстеження та наведення продемонстрована на Рис. 2.11.

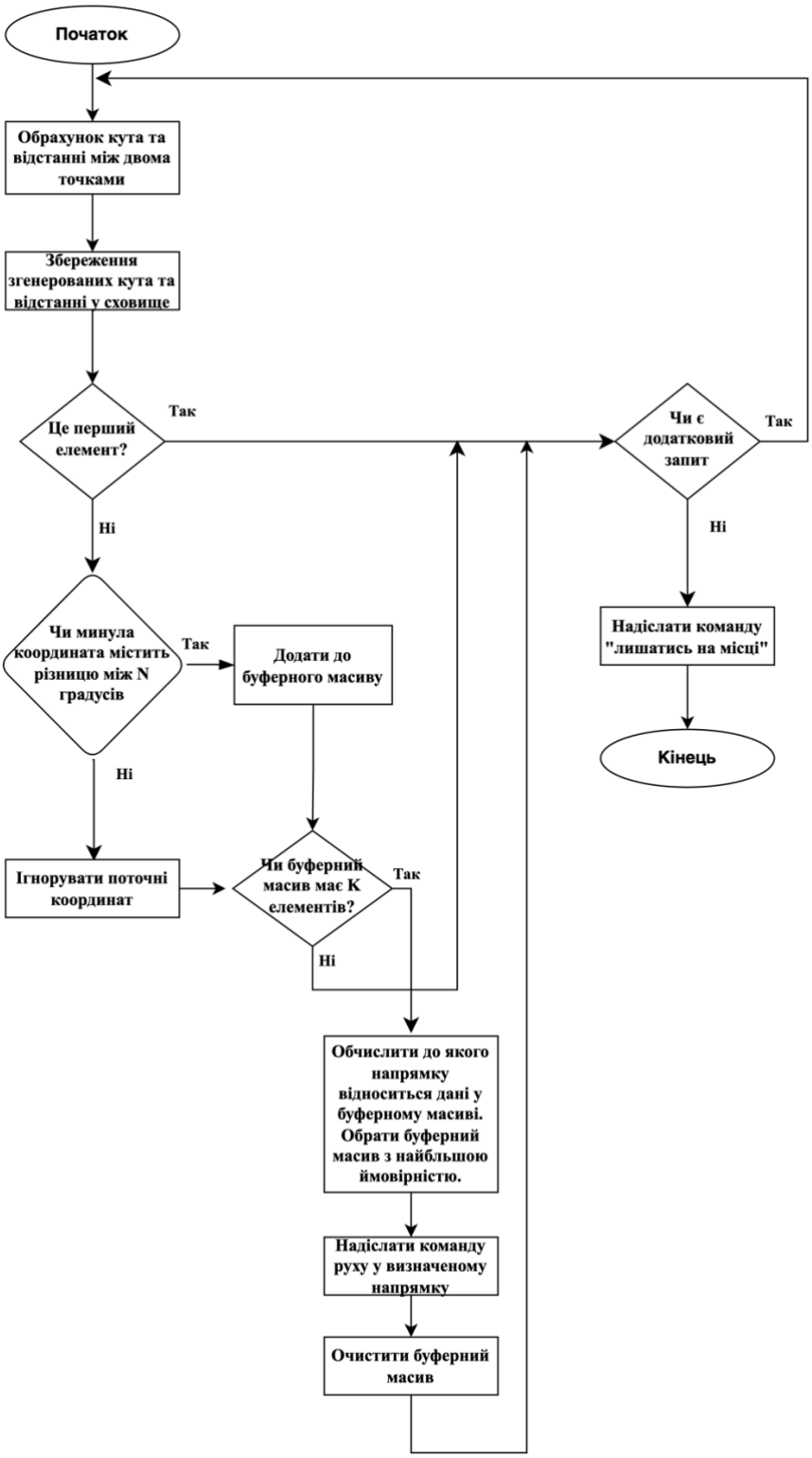


Рис. 2.11. Схема алгоритму відстеження та наведення.

Запропонований алгоритм відстеження та наведення використовує значення куту angle та вектора dist для збереження у буферному масиві.

Кут відхилення між центром прямокутника з об’єктом розпізнавання та центром інформаційного кадру ВКВ обраховується за формулою [2]:

$$d_x = B_X - A_X$$

$$d_y = A_y - B_y$$

$$angle = \frac{arctan(d_x, d_y) \times 360}{2 \times \pi}$$

де:

$A_{x,y}$ – координати центру інформаційно кадру ВКВ;

$B_{x,y}$ – координати центру регіону розпізнавання;

$d_{x,y}$ – координати для обрахунку кута відхилення між двома точками для обрахунку арктангенсу у картезіанському просторі;

angle – кут а радіанах між центром інформаційного кадру ВКВ та об’єктом розпізнавання.

Наступним кроком, при умові різниці у куті angle більше ніж заданій N у порівнянні з попереднім значенням у буферному масиві для поточної області розпізнавання – додавання поточного кута у список потенційних напрямків руху. Якщо така поведінка при визначення напрямку руху повторюється K разів, відправляється команда на рух UAV у цьому напрямку. Значення на скільки потрібно здійснити рух залежить від позиції останніх координат у буферному масиві. Для прик структурна схема алгоритму відстеження та наведення ладу, UAV може бути повернутий за годинниковою стрілкою на 20 градусів, та опущений униз на 6 сантиметрів.

Для тестування роботи алгоритму доречно скористатись симульованим середовищем з випадково згенерованими об’єктами. Вхідні значення: роздільна здатність інформаційного кадру ВКВ: 960x720 пікселів; для тестування було згенеровано 500 випадкових об’єктів. Різні кольори визначають різні класи об’єктів. Кількість запусків: 2. Варто зауважити, що поява кожного об’єкта на екрані відбувається послідовно, один за одним. Деякі об’єкти можуть належати до одного спільного класу. Результати алгоритмічного метода відстеження та наведення представленні на Рис. 2.12.

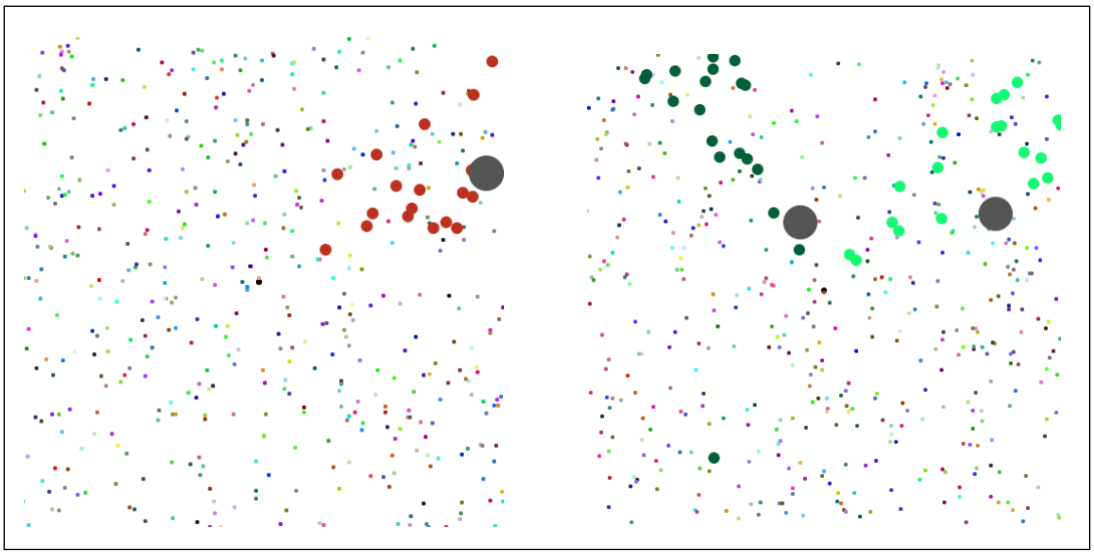


Рис. 2.12. Симуляція метода відстеження та наведення на рухомі об’єкти з 500 випадково згенерованими сутностями. Найбільша крапка визначає фінальний напрямок руху КФС.

Алгоритм розділяє інформаційний кадр на 4 рівні секції і при наявності великої кількості об’єктів в одній з цих областей, розраховує шлях відповідно від координат суміжних об’єктів. Найбільша точка визначає фіналізований напрямок руху для конкретного класу. При умові якщо декілька об’єктів належать до одного класу, обирається найближчий елемент до початкової визначеної точки цього класу.

У наступному підрозділі буде розглянуто подальшу ітерацію даного алгоритму з використанням навчання з підкріпленням.

2.5.2. Метод відстеження та наведення на об’єкт з використанням навчання з підкріпленням.

При виконанні задачі відстеження існує проблема оперативного скерування ВКВ у правильному напрямку. Така проблема може виникати при недостатніх апаратних можливостях системи, коли ОС не встигає опрацьовувати вхідні алгоритмічні задачі. У цьому підрозділі висвітлюється модифікований алгоритм відстеження та наведення з використанням навчання з підкріпленням.

Координати x,y,z обраховуються за допомогою методу відстеження та наведенні на базі алгоритму навчання з підкріпленням DDPG. Цей підхід базується на обчисленні відстані між центром середовища спостереження (модуля ВКВ), та

центром розпізнаного об’єкта. Відповідно від цих результатів буде відбуватися рух КФС на прикладі UAV(дрону) у певному напрямку [3].

Для вирішення поставленої задачі доцільно розділити на декілька кроків:

Задання середовища для слідкування.

Відносні координати розташування об’єкта обраховуються використовуючи різницю між центром середовища спостереження, та центром об’єкта розпізнавання. Ці дані використовуються для моделі навчання з підкріпленням.

Середовище для навчання дрона складається з наступних частин:

- Середовище спостереження: *0 <= x <= 960; 0 <= y <= 720*;
- Середовище дії (action): *–60 < action1 <60; –60 < action2 <60*. Де action1 контролює дії руху вліво та вправо, а action2 контролює дії руху вниз та вверх по висоті;
- Центр екрану: x=480, y=360;
- Швидкість дрону вираховується в діапазоні *–100 < Speed < 100*.

Наведені вище параметри необхідні для визначення координат руху x та y. Схематичне представлення середовища керування дрона наведене на Рис. 2.13.

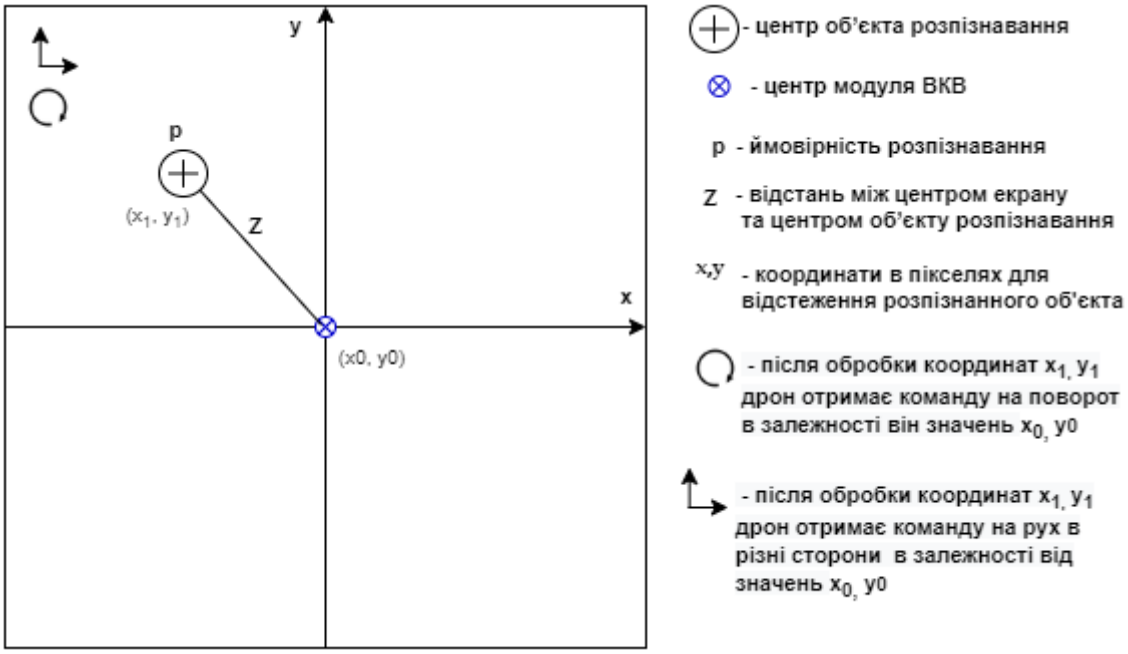


Рис. 2.13. Середовище керування дрона.

Навчання моделі на базі DDPG агента.

На основі аналізу літературних джерел, для вирішення поставленої задачі – відстеження та відслідковування об’єктів, обрано модель DDPG, оскільки така модель підходить для оцінки стратегії дії (руху дрону і різних напрямках). На Рис. 2.14 зображена структурна схема розробленої моделі навчання з підкріпленням.

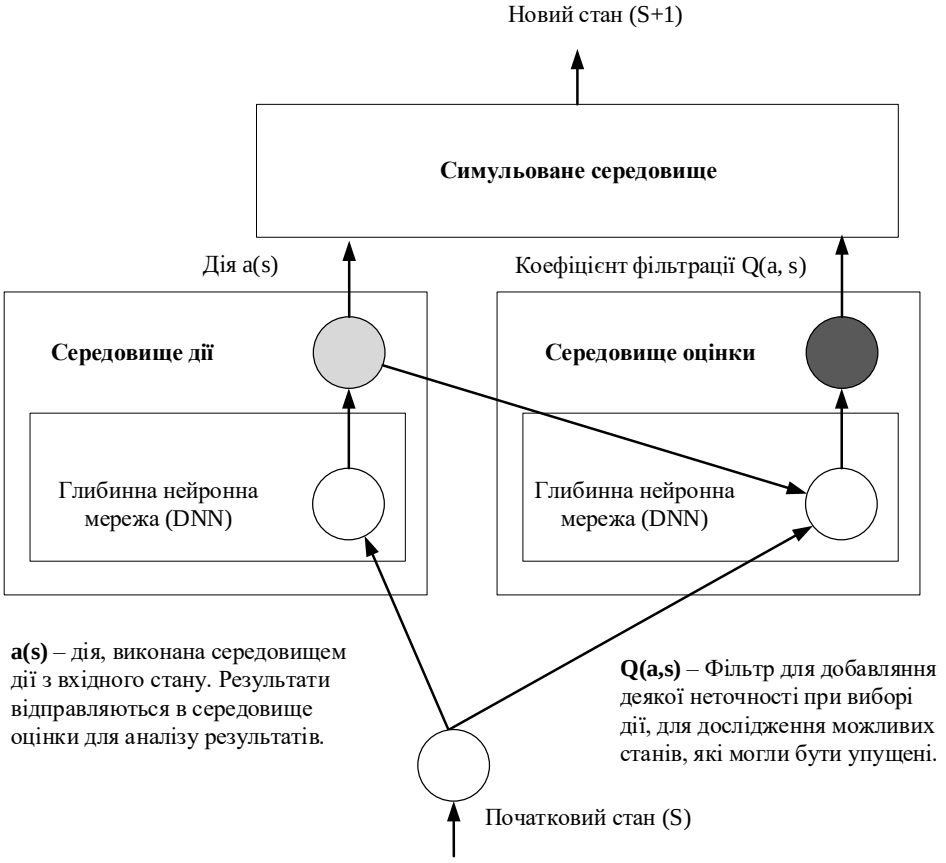


Рис. 2.14. Структурна схема запропонованого алгоритму навчання з підкріпленням на базі агенту DDPG.

Для навчання DDPG агента, необхідно визначити функцію оцінки (винагороди), яка матиме на вході координати x та y центру кожного регіона розпізнавання. Через декілька ітерацій оцінки модель повинна навчитися розуміти чи вона навчається (рухається у правильному напрямку) чи деградує.

Попередня обробка даних.

Впродовж процесу тренування, згенеровано випадковим чином регіони розпізнавання, розмірністю *0 <= x <= 960; 0 <= y <= 720* пікселів.

Для задачі визначення часу, напрямку та швидкості пересування UAV (дрона) використано агент на базі алгоритму «Continious Mountain Car» [3].

Тренування агента ділиться на незалежні епізоди, кожен з яких виконується покроково. Алгоритм кожного кроку (step function) наступний [3]:

- Оновлення позицій враховуючи вхідні дії для генерування наступного стану;
- Обрахунок нагороди;
- Перевірка чи стан кінцевий (done);
- Отримання нагороди, наступного стану та прапорця завершення роботи.

Для того щоб визначити значення стану (чи він кінцевий), необхідно щоб виконувалась хоча б одна з наступних умов:

- кількість завершених кроків > 15;
- x чи y набувають значень більших за максимально встановлені в середовищі спостереження (у даному випадку це 960x720);
- координата центру екрану співпадає із координатою центру регіону об’єкта розпізнавання.

Ітерація завершується, коли стан є кінцевий. При цьому виконується функція скиду значення.

Алгоритм функції скиду наступний:

- обнулення координат та задання нових випадковим чином в межах середовища спостереження;
- ініцалізація кінцевого стану (done);
- ініціалізація внутрішнього стану.

Для того щоб визначити коефіцієнт функції оцінки (dist) використано наступну формулу [3]:

$$d_x = A_x - A_y$$

$$d_y = B_x - B_y$$

$$dist = \sqrt{d_x^2 + d_y^2}$$

Де:

$A_{x,y}$– координати центру екрану;

$B_{x,y}$– координати центру прямокутника з об’єктом розпізнавання;

$d_{x,y}$ – dx,y-координати для обрахунку відстані між двома точками;

dist – результуючий вектор. Коефіцієнт функції оцінки.

Кінцева формула функції оцінки була оптимізована методом спроб та помилок і в результаті набула наступного вигляду [3]:

$$dist > 100, \quad reward += -(dist \times 0.30)$$
$$dist < 100, \quad reward += (dist + 90)$$

Де:

reward – функція нагороди;

dist – результуючий вектор. Коефіцієнт функції оцінки.

Кожну ітерацію, функція оцінки буде збільшуватись, або зменшуватись відповідно від того чи в правильному напрямку рухається тренування.

Результати тренування для різних параметрів наведені на Рис. 2.15.

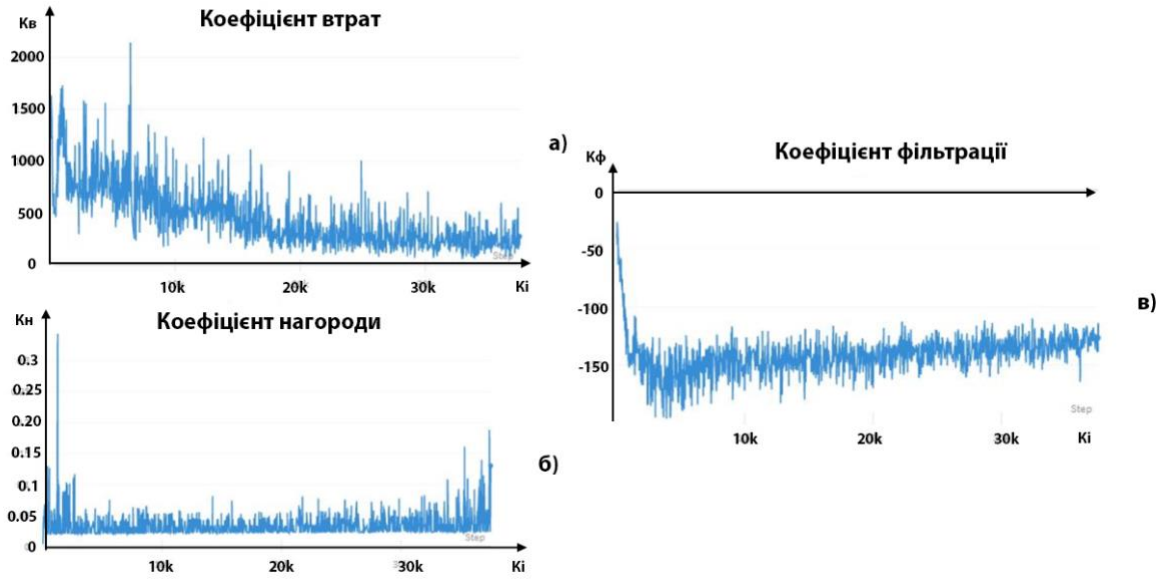


Рис. 2.15. Результати аналізу параметрів при тренуванні моделі навчання з підкріпленням на базі DDPG з кроком у 40 тисяч: а) коефіцієнт втрат; б) коефіцієнт нагороди (дія); в) коефіцієнт фільтрації.

На Рис. 2.15 коефіцієнт втрат визначає наскільки система помилялася при тренуванні. Коефіцієнт нагороди (дія) вказує які значення нагороди подавалися

моделі під час тренування. Коефіцієнт фільтрації визначено для додавання неточності у результати треннування моделі DDPG.

В результаті тренування з вхідною функцією оцінки була отримана модель, яка натренована виконувати дії (рухи в різні сторони) в залежності від вхідних даних.

Отримані результати відсилаються у вигляді команди по протоколу UDP до КФС UAV, яка виконує відповідний маневр. При знаходженні декількох об’єктів на екрані, UAV вибирає найбільш вірогідний і слідує за ним. Дослідження проведено одночасно для наступних параметрів:

1) **x, y** – центральні координати розпізнаного об’єкту. Задача системи наблизитись до координат **center** – центр середовища розпізнавання UAV.

Результуючі координати передаються агенту DDPG. Вихідні результати вимірювань продемонстровані на рис. 2.16. Лінії штрихова та пунктирна прямують до координати 490 – центру екрану).

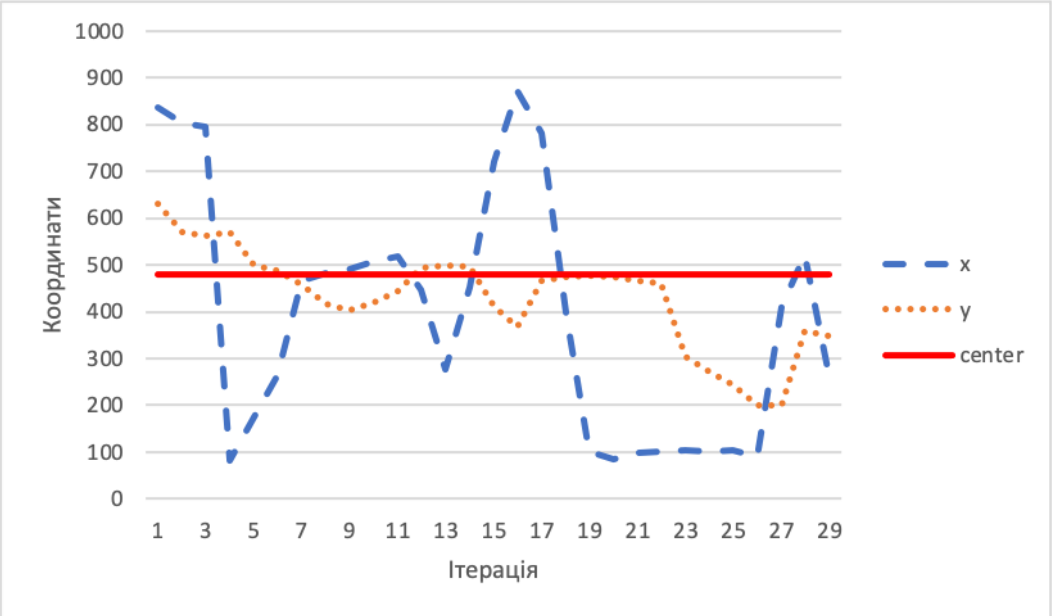


Рис. 2.16. Графік змін координат x,y при виконанні тестових ітерацій.

2) **area_p** – коефіцієнт відстані між середоовищем та об’єктом розпізнавання. Результати вимірювань продемонстровані на Рис. 2.17.

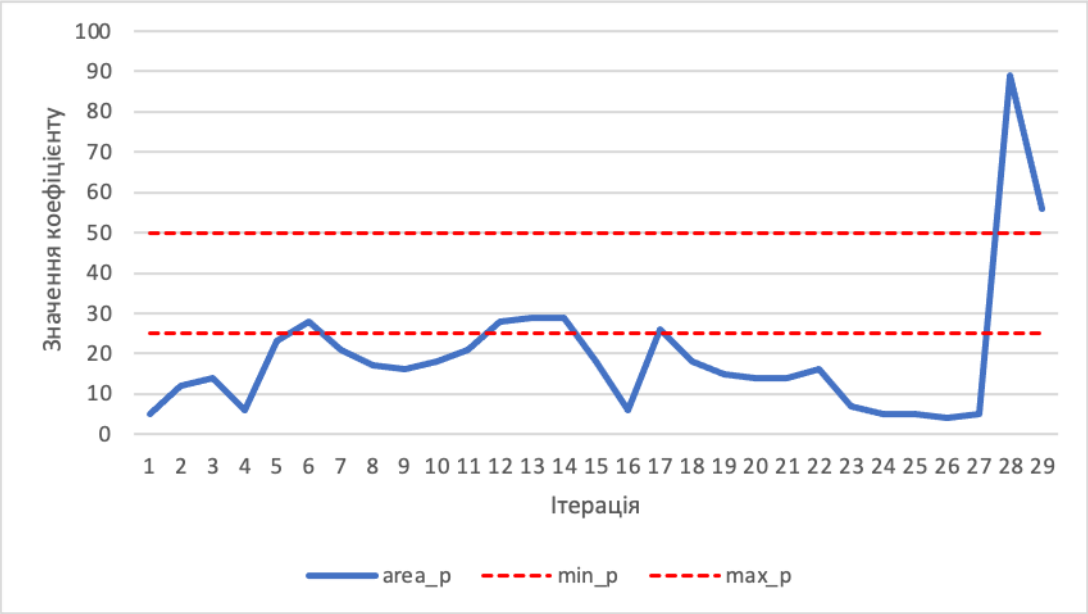


Рис. 2.17. Графік змін координат коефіцієнта відстані area_p.

В результаті тесту, значення area_p безперервно намагається опинитись між двома штрихпунктирними лініями (min_p, max_p), якщо об'єкт змінює відстань до дрона.

3) **Ax, Ay** – це вихідні коефіцієнти руху об’єкта, які відправляється як UDP команди до дрону після обробки агентом DDPG. Результати вимірювань продемонстровані на Рис. 2.18. Ax – зміна напрямку руху дрону по/проти годиннкиової стрілки. Ay – зміна напрямку руху дрону вверх та вниз.

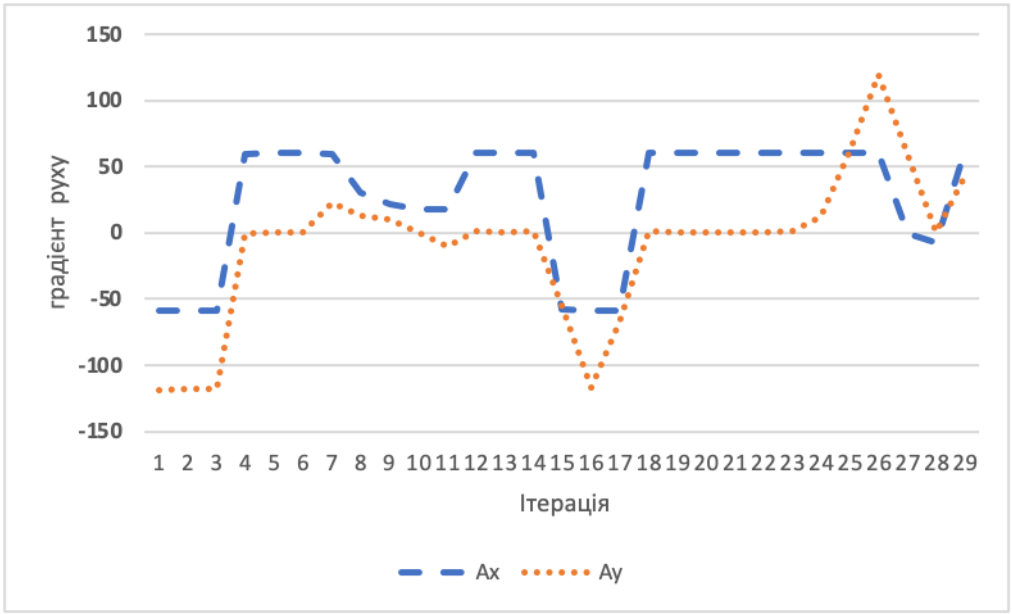


Рис. 2.18. Графік зміни коефіцієнтів руху Ax, Ay при виконанні тестових ітерацій.

Результати експериментальних досліджень показали, що система коректно реагує на зміну розташування об’єкта, відсилаючи команди руху вперед/назад (рис. 2.17), вверх/вниз та відносно руху годинникової стрілки (рис. 2.18). Плюсові градієнти вказують на рух за годинниковою стрілкою та вверх, від’ємні – проти годинникової стрілки та вниз. Існує деяка затримка (0.5 – 1 секунди) пов’язана з швидкістю передачі команд до UAV, проте це може бути вирішено використанням більш високошвидкісного WIFI адаптера [3].

2.5.3. Метод оперативного відстеження об’єктів на базі IOU.

Задачі відстеження великої кількості рухомих та статичних об’єктів на МП потребують значної швидкодії алгоритму, оскільки роздільна здатність екрану користувача є обмежена в розмірах, при високій кількості розпізнаних об’єктів. При цьому швидкодія у такому випадку має мати більший пріоритет за точність розпізнавання. У цьому підрозділі розглядається модифікований алгоритм оперативного відстеження об’єктів V-IOU [5].

Алгоритм V-IOU полягає у використанні мінімізаційного фільтра IOU (формула 2.1) для перекривання локалізованих регіонів декількох розпізнаних об’єктів. Цей метод дозволяє перевірити чи той самий розпізнаний об’єкт лишається на екрані, з присвоєнням йому унікального ідентифікатора. При цьому у сучасних системах розпізнавання таких як yolov4, які мають достатню якість визначення об’єктів, хибні передбачення, які продукуються значеннями ХП та ХН можуть бути проігнорованими. Схематично процес генерації ідентифікаторів зображено на рис. 2.19.

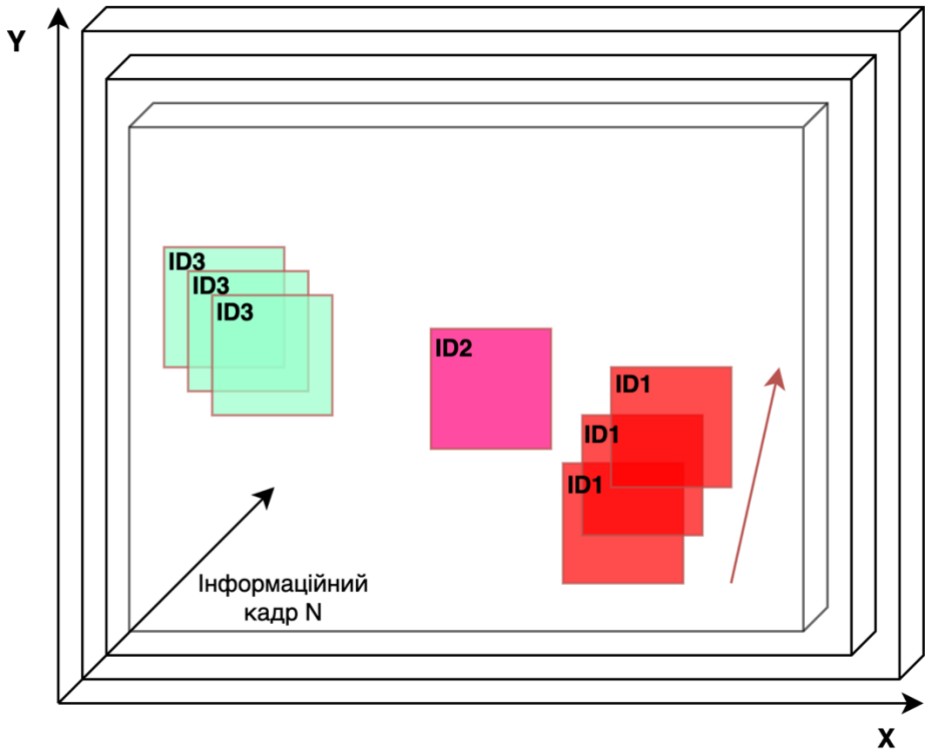


Рис 2.19. Схематичне представлення алгоритму V-IOU. Де X,Y – координатна сітка інформаційних кадрів де розміщені регіони розпізнавання; N – послідовність інформаційних кадрів; ID1,ID2, ID3..ID_n – Унікальні ідентифікатори, які присвоюються спільному відстеженому об’єкту.

Кожен об’єкт визначає свою приналежність до певного ID_n в залежності від ступеня перекриття обох об’єктів шляхом визначення найбільшого IOU. Такий підхід є достатньо ресурсозатрантим, хоча і показує значну ефективність при виконанні задачі відстеження.

Тому для оптимізації задачі подібності двох об’єктів застосовують такі алгоритми як K-D дерево та Угорський алгоритм. Такі алгоритми дозволяють зменшити кількість перерозподілень $ID_{n.}$ при втраті об’єкта на декілька інформаційних кадрів, при високому ступені ХП значень.

Досліджено, що Угорський алгоритм збіжності відстежених об’єктів показує ліпші результати при роботі моделі пошуку і розпізнавання Yolov4 у порівнянні з алгоритмом збіжності K-D дерево. Різниця у точності полягає у

специфіці роботи алгоритмів. Головна задача Угорського алгоритму полягає у знаходженні мінімальної ціни збіжності об’єктів, у той час як алгоритм K-D дерева лише передбачає позитивні (ІП) значення.

Для тестування метода відстеження об’єктів у реальному часі, використано еталонну вибірку MOT17 з алгоритмами збіжності K-D дерево та Угорським алгоритмом. Результати аналізу продемонстровані на Рис. 2.20 та Рис. 2.21 [5]. Методи оцінки якості відстеження також висвітлені у пункті 2.1.

	IDF1	IDP	IDR	Rcll	Prcn	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	IDt	IDa	IDm
MOT17-09-SDP	48.8%	64.2%	39.3%	57.3%	93.5%	26	7	17	2	211	2274	45	78	52.5%	0.149	34	18	7
MOT17-05-FRCNN	44.7%	61.6%	35.1%	51.4%	89.9%	133	24	56	53	397	3365	62	72	44.7%	0.172	82	20	40
MOT17-04-SDP	61.9%	74.5%	52.9%	69.0%	97.1%	83	32	39	12	981	14763	119	367	66.6%	0.151	82	46	13
MOT17-09-FRCNN	43.5%	61.1%	33.7%	53.3%	96.6%	26	6	16	4	99	2485	35	38	50.8%	0.096	25	15	5
MOT17-02-SDP	34.7%	47.2%	27.4%	44.7%	77.2%	62	11	32	19	2458	10276	168	341	30.6%	0.199	110	69	14
MOT17-11-DPM	50.5%	72.2%	38.8%	50.0%	93.1%	75	8	27	40	351	4720	50	55	45.7%	0.219	33	27	11
MOT17-02-FRCNN	31.5%	52.7%	22.5%	34.6%	81.1%	62	6	26	30	1494	12152	91	124	26.1%	0.126	53	49	11
MOT17-11-SDP	53.6%	63.3%	46.5%	66.6%	90.6%	75	21	37	17	648	3154	54	100	59.1%	0.149	39	27	14
MOT17-09-DPM	40.5%	50.4%	33.8%	53.4%	79.7%	26	2	19	5	726	2479	70	164	38.5%	0.273	49	23	6
MOT17-13-FRCNN	50.9%	61.7%	43.4%	55.8%	79.3%	110	32	48	30	1698	5150	192	257	39.5%	0.168	131	84	32
MOT17-10-SDP	49.0%	56.9%	42.9%	67.4%	89.5%	57	23	30	4	1018	4180	166	292	58.2%	0.205	111	61	10
MOT17-13-DPM	24.9%	69.3%	15.2%	19.2%	87.6%	110	8	25	77	317	9412	40	106	16.1%	0.272	30	26	16
MOT17-11-FRCNN	52.4%	70.3%	41.7%	55.2%	93.0%	75	15	32	28	393	4232	42	48	50.5%	0.095	31	23	12
MOT17-10-DPM	29.5%	50.2%	20.9%	36.2%	87.2%	57	7	18	32	682	8190	80	142	30.3%	0.252	38	48	7
MOT17-04-FRCNN	49.7%	68.0%	39.1%	53.2%	92.4%	83	17	43	23	2072	22239	91	88	48.7%	0.108	41	55	5
MOT17-13-SDP	59.8%	77.2%	48.8%	54.8%	86.6%	110	44	23	43	987	5266	74	204	45.7%	0.213	72	25	27
MOT17-10-FRCNN	39.5%	46.3%	34.4%	57.8%	77.8%	57	15	35	7	2117	5422	223	306	39.5%	0.162	143	88	16
MOT17-04-DPM	33.4%	53.9%	24.2%	38.8%	86.3%	83	4	45	34	2934	29093	235	389	32.2%	0.217	104	139	10
MOT17-05-SDP	43.3%	54.1%	36.1%	58.0%	86.8%	133	32	63	38	611	2907	88	125	47.9%	0.165	113	22	47
MOT17-02-DPM	21.9%	61.0%	13.4%	18.9%	86.6%	62	4	14	44	546	15061	48	102	15.7%	0.246	23	30	5
MOT17-05-DPM	36.3%	62.1%	25.6%	35.9%	87.0%	133	11	52	70	370	4433	65	118	29.6%	0.248	71	22	29
OVERALL	44.5%	62.4%	34.6%	49.2%	88.7%	1638	329	697	612	21110	171253	2038	3516	42.3%	0.170	1415	917	337

Рис. 2.20. Результати тестування метода збіжності K-D дерево для різних моделей ЗНМ з тестовою еталонною вибіркою MOT17.

	IDF1	IDP	IDR	Rcll	Prcn	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	IDt	IDa	IDm
MOT17-09-SDP	42.7%	53.1%	35.7%	64.2%	95.5%	26	9	16	1	161	1905	72	130	59.8%	0.158	69	11	11
MOT17-05-FRCNN	48.9%	66.4%	38.7%	52.9%	90.7%	133	24	58	51	375	3258	58	79	46.6%	0.171	78	23	43
MOT17-04-SDP	66.2%	77.2%	58.0%	73.2%	97.5%	83	38	33	12	910	12728	121	338	71.1%	0.153	93	28	14
MOT17-09-FRCNN	50.2%	68.2%	39.8%	57.4%	98.6%	26	5	19	2	43	2266	35	49	56.0%	0.109	29	13	7
MOT17-02-SDP	28.7%	37.4%	23.3%	47.7%	76.7%	62	12	36	14	2692	9725	273	406	31.7%	0.200	202	72	21
MOT17-11-DPM	39.2%	54.2%	30.7%	51.9%	91.7%	75	9	26	40	444	4538	59	69	46.6%	0.219	49	25	16
MOT17-02-FRCNN	33.1%	54.8%	23.7%	35.0%	80.8%	62	6	27	29	1546	12085	96	138	26.1%	0.125	68	42	14
MOT17-11-SDP	50.9%	57.9%	45.4%	70.8%	90.4%	75	26	33	16	709	2755	82	109	62.4%	0.151	64	26	20
MOT17-09-DPM	38.3%	45.7%	33.0%	56.9%	78.8%	26	2	19	5	815	2296	81	165	40.1%	0.270	68	18	7
MOT17-13-FRCNN	47.0%	53.7%	41.8%	59.3%	76.1%	110	37	49	24	2163	4737	278	326	38.3%	0.178	199	90	39
MOT17-10-SDP	49.5%	55.3%	44.7%	72.1%	89.2%	57	29	24	4	1124	3582	186	305	61.9%	0.206	126	49	11
MOT17-13-DPM	26.5%	71.9%	16.2%	19.8%	87.7%	110	9	26	75	323	9337	29	92	16.8%	0.271	18	24	13
MOT17-11-FRCNN	47.4%	63.1%	37.9%	55.6%	92.4%	75	16	32	27	430	4192	48	50	50.5%	0.095	38	22	14
MOT17-10-DPM	34.9%	58.1%	24.9%	37.4%	87.2%	57	7	18	32	707	8042	57	137	31.4%	0.251	23	41	7
MOT17-04-FRCNN	52.0%	70.5%	41.2%	54.2%	92.9%	83	18	42	23	1969	21769	86	95	49.9%	0.108	48	42	4
MOT17-13-SDP	60.0%	75.5%	49.8%	57.2%	86.7%	110	47	25	38	1017	4987	83	224	47.7%	0.214	85	19	26
MOT17-10-FRCNN	39.8%	45.7%	35.3%	59.8%	77.4%	57	14	38	5	2243	5163	235	329	40.5%	0.165	150	84	18
MOT17-04-DPM	35.9%	54.9%	26.7%	41.9%	86.4%	83	7	43	33	3144	27627	273	428	34.7%	0.218	173	98	12
MOT17-05-SDP	46.9%	56.1%	40.2%	63.4%	88.4%	133	32	70	31	578	2533	96	131	53.6%	0.165	132	22	59
MOT17-02-DPM	20.3%	55.2%	12.4%	19.5%	86.4%	62	4	14	44	568	14965	58	113	16.1%	0.248	36	25	3
MOT17-05-DPM	34.5%	56.3%	24.9%	37.4%	84.4%	133	10	55	68	477	4332	68	123	29.5%	0.247	78	22	34
OVERALL	45.2%	61.4%	35.8%	51.7%	88.6%	1638	361	703	574	22438	162822	2374	3836	44.3%	0.171	1826	796	393

Рис. 2.21. Результати тестування метода збіжності Угорського алгоритму для різних моделей ЗНМ з тестовою еталонною вибіркою MOT17.

де:

— IDF1: Співвідношення правильно ідентифікованих виявлень за

середньою кількістю істинних об’єктів та обчислених виявлень;

— IDP: глобальна точність мінімальних витрат;

— IDR: Глобальне якість мінімальних витрат;

— GT: Кількість істинних об'єктів;

— RCLL: Відношення правильних виявлень до загальної кількості реігонів розпізнавання GT;

— PRCN: Співвідношення ІП / ІП + ХП;

— MT: кількість переважно відстежуваних траєкторій. Тобто ціль має однакову ID принаймні 80% часу відстеження;

— PT: частково відстежуванні траєкторії істинних об’єктів;

— ML: кількість переважно втрачених траєкторій. тобто ціль не відстежується принаймні 20% її тривалості часу відстеження;

— FP: кількість помилкових виявлень (ХП);

— FN: кількість пропущених виявлень (ХН);

— IDs: кількість разів, коли ідентифікатор перемикається на інший раніше відстежуваний об'єкт;

— Frag: кількість фрагментацій, коли шлях відстеження переривається виявленням промаху;

— FM: кількість фрагментацій шляху відстеження. Підраховує, скільки разів переривається траєкторія істинних об’єктів;

— MOTA (формула 2.9): Точність відстеження багато-об'єктного відео потоку. Підсумовує три джерела помилок (ХН, ХП, IDs) з єдиним показником продуктивності (GT);

— MOTP (формула 2.8): Оцінка влучності позиціювання розпізнаних об’єктів. Середня несхожість між усіма ІП та відповідними їм істинними об’єктами. Для перефедення у відсотки потрібно скористатись формулою: $(1 - MOTP) \times 100$ (%);

— iDt, iDa, iDm: істинно-позитивні (ІП) ids.

Оскільки запропонований алгоритм збіжності (Угорський алгоритм)

знаходить оптимальний шлях для розпізнаних об’єктів, в результаті отримано більшу кількість ХП (FP) значень, але у той час меншу кількість ХН (FN) значень. Додатково, значення МТ є більшим у порівняні з алгоритмом K-D дерево (361 проти 329).

В цілому, запропонований алгоритм працює гірше, коли модель розпізнавання має багато помилкових спрацьовувань і краще, коли до алгоритму не надходить великої кількості помилкових спрацьовувань. У той же час в контексті вибору моделі розпізнавання Yolov4, при малій кількості хибних спрацювань такої моделі, запропонований Угорський алгоритм як метод збіжності буде показувати більшу точність.

Варто відзначити, що значення мінімізаційного фільтру IOU має бути максимально збільшено, для зменшення кількості залишкових регіонів розпізнавання.

2.5.4. Методи мемоїзації відстежених об’єктів.

При розробці методів відстеження довільної кількості об’єктів важливо визначити граничні властивості таких алгоритмів.

Мемоїзація – спосіб збереження інформації відстеження з попередніх інформаційних кадрів, для оптимізації шляху відстеження у наступних кадрах.

В ході дослідження методів відстеження об’єктів розроблено наступні методи мемоїзації:

— **FPL (граничний ліміт кадрів/frame pending limit)** [5]. Даний фільтр обраховує кількість кадрів для збереження напрямку траєкторії, при умові що наступний кадр їй не відповідає. Варто зауважити, що при високих значеннях цього фільтри виникатиме менше перемикань ідентифікаторів ID_n, при цьому збільшується ймовірність виникнення ХП (FP) значень з ID_n присвоєним до іншого об’єкту.

— **BAT (граничний кут нахилу/ boundary angle threshold)** [5]. Метод обраховує кількість об’єктів, які пересікли граничну лінію, при умові що кут між їхньою траєкторією і граничною лінією є більшим за задане значення граничного кута λ у градусах. При цьому, при 90 градусів будуть обраховуватись лише чисто

перпендикулярні до граничної лінії об’єкти. При 0 градусів будуть включені усі об’єкти. Графічна репрезентація розроблених методів показана на рис 2.22.

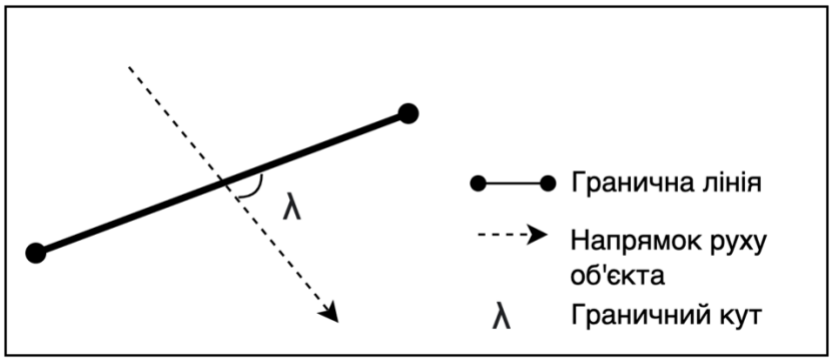


Рис. 2.22. Структурна схема метода граничного кута нахилу, для фільтрації вихідних відстежених об’єктів.

Така методика дозволяє відфільтрувати відстеженні об’єкти, які перетинають граничну лінію під великим кутом.

— **PFTP (траєкторія передбачених кадрів/past frames trajectory prediction)** [5]**.** Така методика дозволяє обрахувати траєкторію руху відстеженого об’єкта, враховуючи попередні відстеження, які пересікли граничну лінію. У більшості випадків, траєкторія центру отриманого регіону розпізнавання від yolov4 змінюється кожного кадру, тому доречно використати ЗФС для верифікації моменту пересічення об’єктом граничної лінії (рис 2.23).

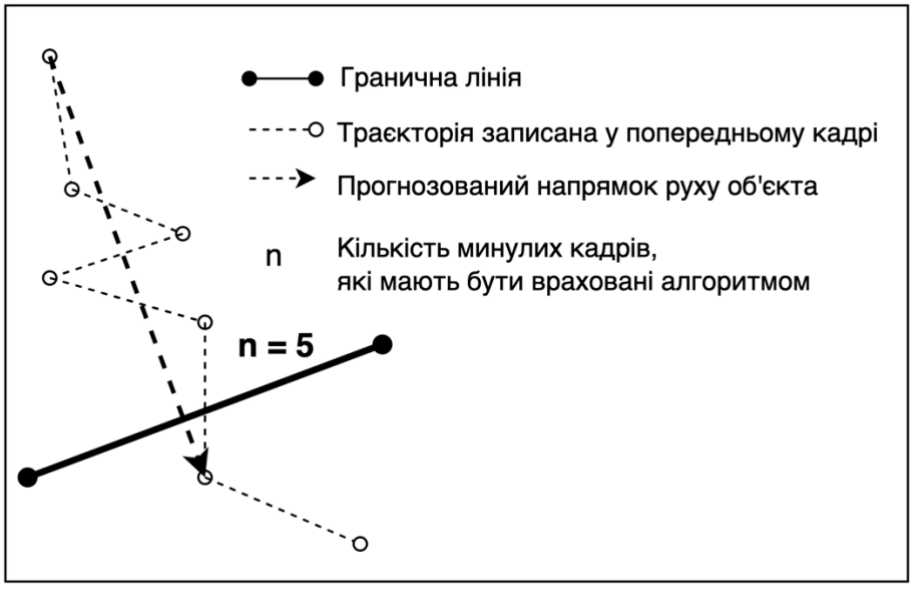


Рис. 2.23. Структурна схема метода траєкторії передбачених кадрів, для нормалізації траєкторії до граничної лінії.

Для прикладу, якщо відстежений об’єкт ID був втрачений на протязі процедури відтеження, алгоритм вибере останнє розташування об’єкта з таким ID з пам’ятті, нормалізуючи траєкторію.

2.6. Метод квантизації ваг моделі нейронної мережі для мобільної платформи.

Квантизація – процес зменшення кількості бітів які відповідають за колір зображення при обрахунку вагового коєфіцієнта.

Зменшення розміру вагових коефіцієнтів є важливим для задач РО на МП, оскільки така платформа має обмежені апаратні можливості.

Афінні перетворення або афінна квантизація або лінійне перетворення – це процес квантизації який співвідносить вхідні числа з рухомою комою в квантизований проміжок, наприклад який слкадається з 8-бітних цілих чисел [0,255] з лінійною інтерполяцією. Таку задачу можна вивести до формули [1]:

$$w_{float} \in [0,255], \qquad w_{float} = (w_q - bias_q) \times scale$$

Де, w_{float} – вхідні вагові коєфіцієнти моделі в форматі з рухомою комою; $scale$ – коєфіцієнт мінімального значення у форматі з рухомою комою; w_q – цілочисельне значення квантизованого вагового коєфіцієнта; $bias_q$ – цілочисельне зміщення до максимального значення з рухомою комою.

Така методика застосовується для зменшення вагових каєфіцієнтів з рухомою комою у скомпресований ваговий коєфіцієнт з N – біт розмірністю.

Варто зауважити, при збільшенні коефіцієнтів $scale$ та $bias_q$ зменшується розмір вагових коефіцієнтів, але і пропорційно зменшується якість розпізнавання. Для визначення оптимальних параметрів квантизації, необхідно відтестувати вихідні моделі на тестовій вибірці. Ступінь квантизації q доречно використовувати при аналізі метрик продуктивності ЗНМ на МП.

Висновки до другого розділу.

Проаналізовано метрики оцінювання результатів розпізнавання та відстеження та використано матрицю невідповідностей або матрицю помилок для визначення продуктивності алгоритму розпізнавання. Введено

мінімізаційний фільтр IOU, який отримав широке використання у пропонованих алгоритмах розпізнавання та відстеження. Визначено метрику середньої влучності mAP для визначення якості роботи ЗНМ. Для задач відстеження визначено метрики MOTA та MOTP.

Сформовано та описано загальну структуру ЗНМ Yolov4, описано модифікації її хребту (backbone), шиї (neck) та голови (head).

Використано модифікований алгоритм навчання без учителя к-середніх++ замість алгоритму к-середніх для генерації якорів розпізнавання ЗНМ на етапі тренування. З результатів дослідження видно, що якість кластеризації вище в алгоритма к-середніх++ при однакових значеннях початкових центроїдів. При більших обчислювальних затратах у порівнянні з алгоритмом к-середніх, помилки при збіжності значно мінімізуються. Для задач РО при формуванні якорів розпізнавання, алгоритм к-середніх++ збільшує точність вибору якорів, що може значно покращити визначення помилки при класифікації на зображенні.

Розроблено методи фільтрації результатів розпізнавання, у тому числі фільтри ЗФС, ФРК та задання порогових значень IOU та object confidence в залежності від потреб поточної задачі. Такі фільтри дозволяють точніше відображати результати розпізнавання в багатокласовій системі, зменшуючи надлишковість регіонів розпізнавання.

Розроблено 3 метода відстеження об’єктів у реальному часі. Перший метод є алгоритмічним представленням відстеження об’єктів в залежності від їх відхилення від центру інформаційного кадру з можливістю наведення на об’єкт за допомогою ВКВ КФС. Другий метод полягає у модифікації алгоритму відстеження, шляхом додавання агенту динамічного середовища DDPG, який був навчений методом навчання з підкріпленням. Для зменшення апаратної залежності, був розроблений третій модифікований алгоритм оперативного відстеження на базі IOU, який полягає у використанні Угорського алгоритму збіжності для порівняння об’єктів відстеження у реальному часі. Ефективність роботи алгоритму була підтверджена у ході тестів на базі еталонної вибірки MOT17.

Запропоновано методи мемоїзації результатів відстеження об’єктів PFTP, BAT та FPL, які полягають у формуванні порогових значень та граничних ліній для збереження та фільтрування результатів відстеження у разі втрати ID об’єкта. Також такі методи дозволяють зменшити частоту перемикань ідентифікаторів ID_n.

Запропоновано метод використання квантизації вагових коефіцієнтів ЗНМ, для зменшення вихідної моделі, при збережені достатньою якості розпізнавання що важливо для виконання задач РО на МП.

РОЗДІЛ 3
ЗАСОБИ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ МОДЕЛІ ЗНМ YOLO НА МОБІЛЬНІЙ ПЛАТФОРМІ

У розділі розроблено алгоритми тренування моделі ЗНМ, автоматичного анотування вхідних зображення та конвертування моделі у CoreML формат для мобільної платформи. Розроблено вбудований модуль для відстеження рухомих об'єктів.

Результати розділу опубліковано в працях автора [2 – 7].

3.1. Визначення гіперпараметрів та функції втрат на етапі тренування моделі.

Функція втрат – функція яка відображає результат визначення події або величини на певне значення дійсного числа, що відображає ймовірнісні витрати, пов'язані з подією. При задачі оптимізації йде спрямоване намагання мінімізувати це значення.

Функція втрат для задач класифікації, або LOSS – це така функція втрат, яка предсталяє ціну, яка була сплачена за помилкові припущення у задачах РО.

Гіперпараметри – це такі параметри, значення яких використовується для керування процесом навчання моделі ЗНМ. Час навчання моделі може залежати від вхідного набору гіперпараметрів. Також гіперпараметри можуть мінімізувати визначену функцію LOSS.

Задачу тренування розробленої ЗНМ доречно розділити на два етапи.

На першому етапі виконується тренування та генерація базових вагових коефіцієнтів моделі з використанням наявних апаратних засобів. При цьому застосовуються методика кластериації k-means++ для обрахунку якорів розпізнавання та задаються вхідні гіперпараметри, такі як кількість ітерацій тренування, визначення розмірів блоків та під блоків ЗНМ, а також задання додаткових параметрів в залежності від обраного алгоритма оптимізації. Також важливою частиною такого тренування є визначення значення LOSS.

На другому етапі виконується задача поліпшення (fine-tuning) тренованої

моделі ЗНМ шляхом перевірки чи поточний ваговий коефіцієнт з найліпшим значенням mAP досяг найвижчого значення.

У даному підрозділі розглядається визначення гіперпараметрів та функції втрат на першому етапі тренування.

У задачах тренування ЗНМ Yolov4, для визначення значення LOSS, на першому етапі використовується комбінація з трьох функцій втрат: функція впевненості ($\boldsymbol{L_{confidence}}$), функція класифікації ($\boldsymbol{L_{classification}}$), та функція локалізації ($\boldsymbol{L_{localization}}$).

Функції $\boldsymbol{L_{confidence}}$ та $\boldsymbol{L_{classification}}$ були використані з архітектурної моделі Yolov3. Вони набувають наступного вигляду [41]:

$$\boldsymbol{L_{confidence}} = \sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}\left[\hat{C}\log\left(C_i\right) + \left(1-\hat{C}\right)\log\left(1-c_i\right)\right]$$

$$-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}\left[\hat{C}_{\mathrm{i}}\log\left(C_i\right) + \left(1-\hat{C}_i\right)\log\left(1-c_i\right)\right]$$

$$\boldsymbol{L_{classification}} = \sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in classes}\left[\hat{p}_i(c)\log(p_i(c)) + (1-\hat{p}_i(c))\log(1-p_i(c))\right]$$

Де, S^2 визначає сітку інформаційнго кадру з $S \times S \in \mathbb{R}$ клітинами. Кожна клітина сітки визначає пропонований регіон розпізнавання B. У той же час кожен пропонований регіон розпізнавання отримує відповідний регіон розпізнавання за допомогою тренованої ЗНМ. На фінальній стадії створюється регіон розпізнавання $S \times S \times B \in \mathbb{R}$.

Значення $\boldsymbol{L_{confidence}}$ для кожного фінального регіону розпізнавання опрацьовується у випадку якщо відсутні розпізнаванні об’єкти у регіоні розпізнавання ($noobj$). При цьому використвоється визначення помилки перехресної ентропії, тобто визначення значення помилки розділяється на дві частини: для сценарію існування об’єктів у регіоні розпізнавання (obj) та відстуності об’єкта ($noobj$). Функція втрат LOSS для $noobj$ збільшує ваговий коефіцієнт λ, що у свою чергу зменшує вагу внеску у поточній ітерації $noobj$.

Значення $\boldsymbol{L_{classification}}$ використовує за аналогією значення помилки перехресної ентропії. Це відбувається при умові, коли якір розпізнавання з індеком j клітини сітки інформаційного кадру i буде відповідати певним вихідним істинним значенням.

У той же час, для поліпшення вихідного значення LOSS у випадку локалізації ($\boldsymbol{L_{localization}}$), був використаний CIOU [Complete IOU] на базі мінімізаційного фільтру IOU (формула 2.1). Його формула набуває наступного вигляду [120]:

$$\boldsymbol{L_{localization}} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v$$

Де:

b, b^{gt} – центральні точки регіону розпізнавання B та B^{gt} відповідно;

c – довжина діагоналі найменшого регіону розпізнавання, яка покриває два суміжні регіони розпізнавання;

$\rho^2(b, b^{gt})$ – евклідова відстань d;

α – позитивний компромісний параметр. Обчислюється за формулою:

$$\boldsymbol{\alpha} = \frac{\mathrm{v}}{1 - \mathrm{IOU} - v};$$

v – значення для виміру коефіцієнта постійності співвідношення сторін. Обчислюється за формулою:

$$\boldsymbol{v} = \frac{4}{\pi^2}(arctan\frac{\mathrm{w}^{gt}}{\mathrm{h}^{gt}} - arctan\frac{\mathrm{w}}{\mathrm{h}})^2;$$

w, h − задані значення розмірів якорів розпізнавання перед початком тренування;

IOU-мінімізаційний фільтр.

Графічна репрезентація обрахування значення $\boldsymbol{L_{localization}}$ з використанням відстані d та довжини діагоналі c наведена на Рис. 3.1.

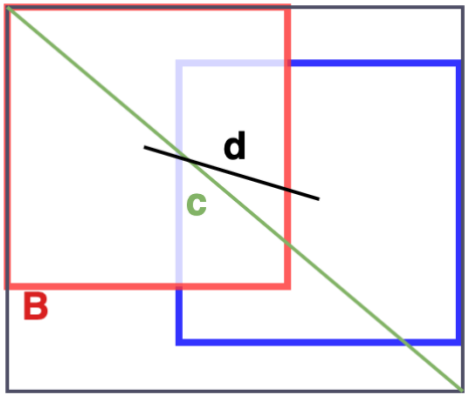


Рис. 3.1. Графічна репрезентація обрахунку відстані d $\boldsymbol{L_{localization}}$. Виконується лінійне зменшення нормалізованої відстані між центральними точками регіонів розпізнавання B.

У випадку використання CIOU значно зменшується нормалізована відстань між двома аналізованими об’єктами.

Отже, коефіцієнт втрат для розроблюваної моделі yolov4 можна вивести за формулою, як різниця втрат функції локалізації, функції впевпненості та функції класифікації:

$$\boldsymbol{LOSS = L_{localization} - L_{confidence} - L_{classification}}$$

Було визначено гіперпараметри для тренування ЗНМ. Основні з них наведені у таблиці 3.1.

Таблиця 3.1. Опис основних гіперпараметрів розроблених моделей ЗНМ Yolov4.

Модель Yolov4	**Batch**	**Subdivisions**	**height /weight**	**Learning Rate (LR)**	**Decay**	**Momentum**
Yolov4_getr	64	16	416/416	0.002	0.0005	0.95
Yolov4_getr_tiny	64	8	416/416	0.001	0.0005	0.9

Де:

batch – загальна кількість вхідних даних (зображень) які можуть бути опрацьовані у одній ітерації;

subdivisions – кількість частинок загальною групи batch, розмір якої

визначається ділення значення batch на subdivisions. Необхідно для коректної роботи GPU для обрахунку таких під груп;

height/weights – вхідні розміри (висота та ширина відповідно) вхідної ЗНМ на етапі тренування та висновування;

Learing Rate (LR) – кількістна характеристика частоти оптимізації вагових кеофіцієнтів;

Momentum – методика оптимізації градієнту при обрахунку попередніх і поточних вагових коефіцієнтів. Використовує акумуляцію руху зміни вагових коефіцієнтів, тобто наскільки історичні зміни градієнту впливають на поточний стан мережі;

Decay – методика оптимізації градієнту при обрахунку попередніх і поточних вагових коефіцієнтів. Слабкіше оновлення ваг для типових ознак. Зменшує розмір великих вагових коефіцієнтів та дає перевагу рідкісним ознакам об'єктів. Зарахунок цього усувається дизбаланс у базі знань.

Для оптимізації вхідних гіперпараметрів, був обраний алгоритм градієнтного спуску Nesterov Accelerated Gradient (NAG) [121], з використанням згаданих вище оптимізацій градієнту Momentum та Decay.

Momentum використовується для зменшення коливань розміру вагових кеофіцієнтів між ітераціями.

$$\boldsymbol{\Delta w_i(it+1)} = -LR\frac{\partial Err}{\partial w_i} + \left(m \times \Delta w_i(it)\right)$$

У той же час Decay використовується для зменшення розміру великих вагових коефціцієнтів за інтерацію:

$$\boldsymbol{\Delta w_i(it+1)} = -LR\frac{\partial Err}{\partial w_i} - \left((d \times \mathrm{LR}(it)\right)$$

Таким чином, для виконання задачі обчислення градієнта методом зворотнього розповсюдження помилки, доречно об'єднати наведені вижче формули для отримання ефективнішого проміжного показника вагового коефціцієнту:

$$\boldsymbol{\Delta w_i(it+1)} = -LR\frac{\partial Err}{\partial w_i} + \left(m \times \Delta w_i(it)\right) - \left((d \times \mathrm{LR}(it)\right)$$

Де:

Δw_i – середнє значення вагового коефіцієнта при ітерації i;

it – поточна ітерація тренування;

LR – Learning Rate;

$Err(w)$ – функція помилки;

m/d – Momentum/Decay.

Значення LR вираховується як комбінація кроків при початковому старті обробки вагових коефіцієнтів, а саме лінійному розігріванні (linear wamup) до певної точки, та покроковій функції Decay (step decay).

Функція лінійного розігрівання (linear warmup) – функція LR для лінійного збільшення значеня кроку до порогового значення. За допомогою цього зменшується нестабільність на ранніх етапах навчання;

Покрокова функція Decay (step decay) – функція LR для ітеративного зменшення значення кроку кожної ітерації на прикладі Decay.

Таким чином, можна вивести формулу для багатокрокового LR у контексті тренування розроблених ЗНМ:

$$\boldsymbol{LR_{it+1}} = \begin{cases} \text{якщо } it \text{ в діапазоні } [St], & dc \in |0 < \mathbb{R} < 1|, & dc \times LR_{it} \\ \text{інакше,} & & LR_{it} \end{cases}$$

Де St – вхідний граничний поріг початку застосування фільтру Decay dc. LR_{it} визначає попередню ітерацію. Значення St буде найбілш оптимальне при визначені його як 80% від максимальної кількості груп (batches), яка у свою чергу напряму залежить від кількості вхідних класів ЗНМ.

3.2. Засоби контейнеризації та масштабування елементів системи.

3.2.1. Узагальнена схема контейнеризації та масштабування системи.

Контейнеризація – процес ізоляції окремих модулів системи засобами віртуалізації, такими як Docker. Кожен окремий контейнер може працювати в окремому ізольованому середовищі, не залежачи від інших контейнерів. Тому

доречно систему задачі РО на МП розділити на окремі контейнери/сервіси для генерації моделі ЗНМ.

Процес створення розробленої ЗНМ доречно розділити на 3 основних етапи: сервіс автономного завантаження анотованих даних, сервіс масштабованого тренування моделі ЗНМ та сервіс конвертування моделі у CoreML формат для МП. На Рис. 3.2 зображена пропонована структурна схема такої системи.

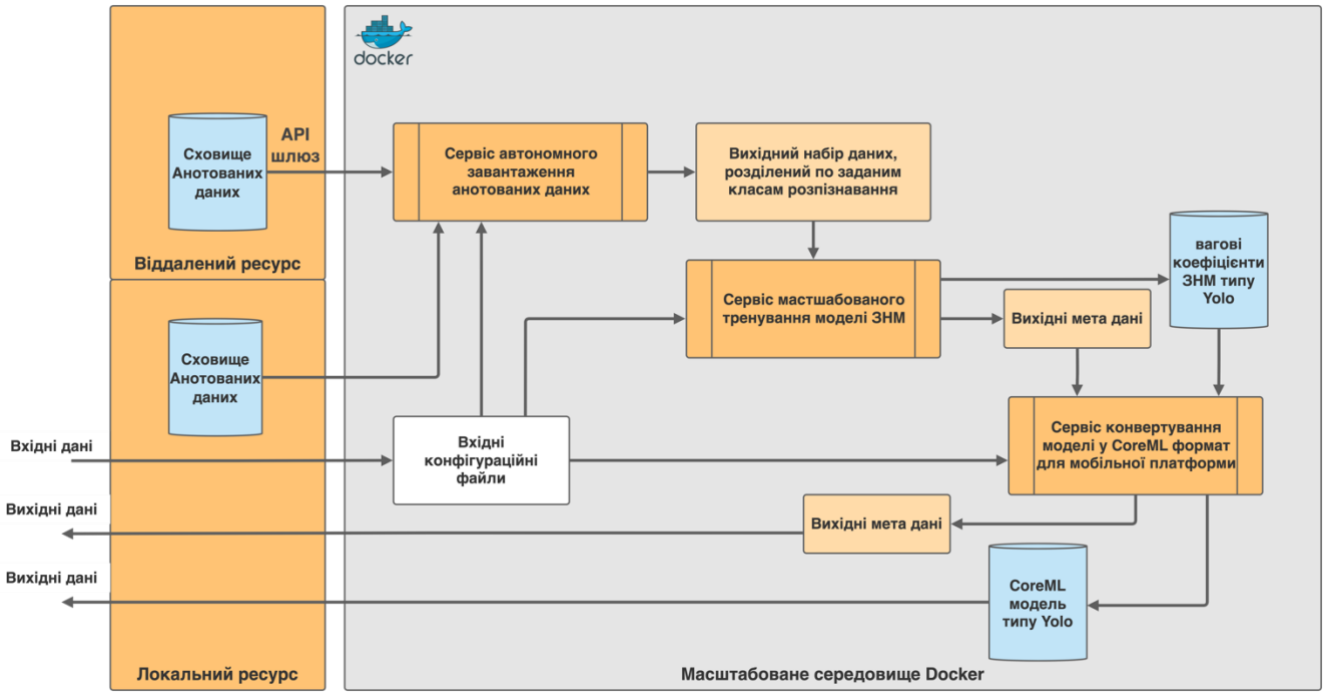


Рис. 3.2. Структурна схема масштабованої системи контейнеризації засобів завантаження анотованих даних, тренування та конвертування моделі ЗНМ у формат для МП.

Де:

- **Сервіс автономного заванатаження анотованих даних**. Визначає вхідні набори анотованих зображень, відповідно до типів вхідних класів ЗНМ. Автоматизовано завантажує вхідні набори тренованої та тестованої вибірки з публічних баз даних, таких як OpenImage 7.0. На виході формує анотовані класи для кожного зображення у прийнятному форматі. У цьому дослідженні використовується формат моделі ЗНМ Yolov4. Також за потреби є можливість додати додаткові анотовані зображення за допмогою утиліти LabelImg;
- **Сервіс масштабованого тренування моделі ЗНМ**. Отримує вхідні анотовані дані для подальшого масштабованого тренування як з використанням CPU, так

і GPU в залежності від наявних апаратних засобів. Підтримує усі наявні операційні системи на базі Unix (Ubuntu 20.04, MacOs Monterey та інші) а також ОС Windows 7, 10, 11. На виході після тренування на анотованих даних з заданими гіперпараметрами, функцією втрат та методом кластеризації, отримуються фіналізовані вагові коефіцієнти та аналітичні значення аналізу ефективності тренування, такі як mAP;

- **Сервіс конвертування моделі ЗНМ у CoreML формат для МП.** Отримує вихідні вагові коефіцієнти у форматі моделі ЗНМ Yolov4. Виконує квантизацію методом афінних перетворень для оптимізації розміру вагового коецієнта. Записує згенеровані за допомогою кластеризації якорі розпізнавання та інші додаткові характеристики ЗНМ у метадані для подальшого опрацювання аплікацією. Засобами CoreML, відбувається конвертація моделі ЗНМ у MLPackage формат для завантаження на МП iOS за допомогою засобів мови програмування Swift 5 та вбудованих засобів системи розробки Xcode.

У наступних підрозділах буде детально розглянуто кожен з згаданих вище контейнерів/сервісів.

3.2.2. Засіб автоматизованого завантаження набору анотованих даних для класу об’єктів.

Процес анотації зображень для вхідного набору класу при тренуванні моделі ЗНМ зазвичай є досить часозатратним. При навчанні з учителем необхідно анотувати кожне тестове та вибіркове зображення з вхідної вибірки.

Для того щоб зменшити час анотування та зменшити ймовірність помилки при виділянні об’єктів, доречно завантажувати вже анотовані зображення з публічних сховищ даних по назві класу як ключу. У такому випадку формується загальна вибірка вже анотованих даних, яка в може бути доповнена вручну анотованими даними.

Як приклад таких сховищ даних, можна використано публічну бібліотеку анотованих зображень OpenImage 7.0 [122]. Набір зображень можна отримувати по назві класу через вбудоване API (Рис 3.3).

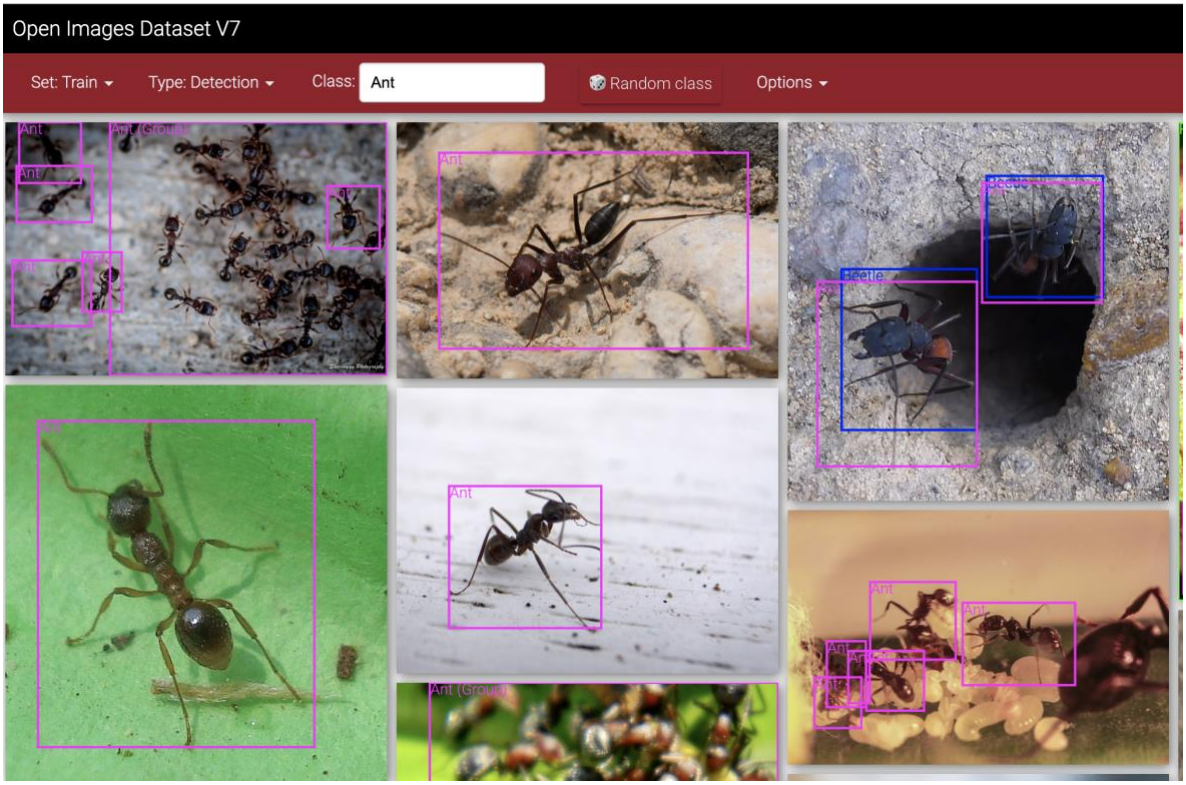


Рис. 3.3. Приклад завантаження даних з публічної бібліотеки OpenImage 7.0 для класу об’єктів «мурахи».

Ендпоінт – це шлюз, який об’єднує серверні процеси додатку з зовнішнім інтерфейсом. За домогою такого щлюзу можна отримати дані з зовіншнього середовища за певними вхідними даними.

Для завантаження анотованих даних для певного класу, необхідно отримати запит до віддаленого хмарного сховища AWS S3 під доменом open-images-dataset з вказанням шляху завантаження та типу бажаних об’єктів (Лістинг 3.1).

Лістинг 3.1. Структура ендпоінту для завантаження анотованого списку зображень по класу.

curl –location –request GET
‘aws s3 cp s3://open-image-dataset/:imageDir/:image/:datasetDir/:ClassesName’
Де:
imageDir – тека для зберігання отриманих даних у теках для тренування та валідації’
image – назва зображення для завантаження, отриманого з CSV списку зображень даного класу;
datasetDir – шлях до віддлаеного набору даних анотованих зображень;
ClassesName – клас або класи зображень, які необхідно завантажити.

Як результат, для кожного вхідного класу, завантажуються зображення та відповідний файл з анотаціями, який конвертується у потрібний формат. Зображення по змовчуванню розподіляються по теках тренування та валідації у співвідношенні 70:30. Це співвідношення за потреби може бути скориговане. У випадку, якщо завантажуються одразу декілька класів, для кожного з них генерується відповідна тека з назвою відповідного класу.

При умові, якщо набір вихідних анотованих даних не є достатнім, або зображення на ньому не відповідають вимогам, доречно доповнити завантажений набір власними вручну анотованими зображення. Це дозволить максималізувати результати тренування ЗНМ. При чому дані можуть бути доповнені у напівавтоматичному режимі при використанні початково навченої ЗНМ для анотації таких же класів об’єктів.

Для тестового прикладу, для задачі тренування, було обрано об’єкти малого розміру – мурахи Messor Structor розміром від 8 до 14 міліметрів при роздільній здатності зображень 512 пікселів.

Був створений окремий набір неанотованих зображень (data) для такого класу об’єктів [123].

Набір анотованих даних був створений з відеопотоку засобом Fast Forward MPEG (FFmpeg) на базі платформи ОС Linux, методом отримання вхідного потоку зображень. Використано два типи відео: для відображення класу мурах з близької так і далекої відстані. Кожен з інформаційних кадрів має роздільну здатність 2160x3840 пікселів.

Фіналізований набір даних складається з 500 зображень, які відображають рух мурах у закритому середовищі. Важливо відзначити, що для анотування даних необхідно максимально чітке зображення на видимій поверхні для збільшення ефективності моделі ЗНМ.

Для процесу анотації зображень, був використаний засіб анотування LabelImg [124]. На Рис. 3.4 зображено процес анотації вхідного класу об’єктів. Як засіб спостереження, використано камеру HD Pro C920.

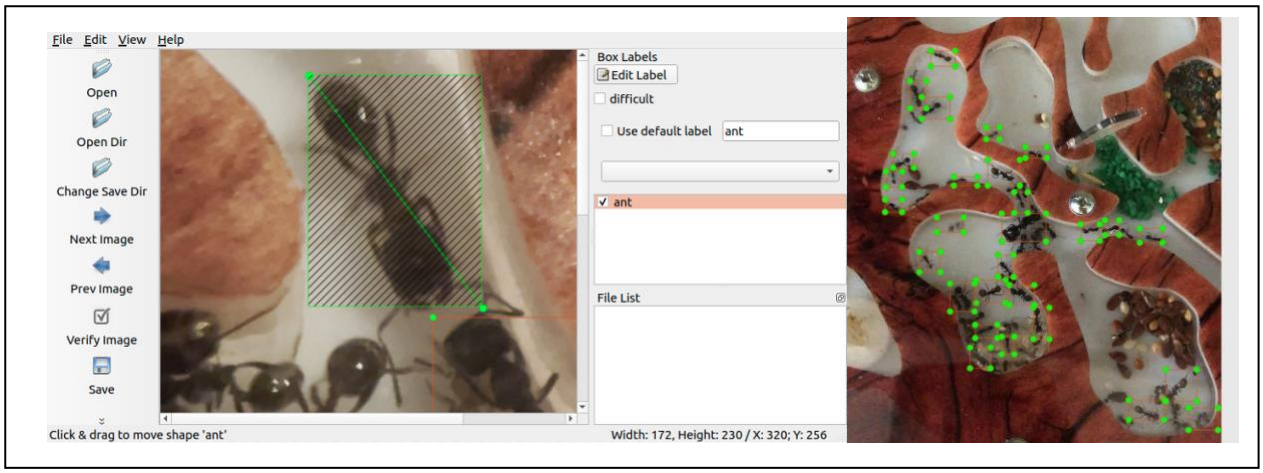


Рис 3.4. Процес анотації тестових зображень засобом Labeling для класу об’єктів Messor Structor різного розміру. Граничний ліміт розміру для класу був заданий як 14 міліметрів.

У Лістингу 3.2 та 3.3 зазначено Docker Image та DockerFile для пропонованого засобу. При чому, вхідні анотовані дані можуть надходити як з віддаленого середовища, так і з локального.

Лістинг 3.2. DockerFile пропонованого сервісу завантаження анатованих зображень.

```
FROM python:3.7

COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN apt-get update

RUN pip install -r requirements.txt
COPY . /app

ENTRYPOINT python ./main.py downloader --classes './classes.txt' -y --type_csv validation --
multiclasses 1 --limit 300 \
   && python ./main.py downloader --classes './classes.txt' -y --type_csv train --multiclasses 1 --limit
1500 \
   && python ./convert_annotations.py \
   && tail -f /dev/null
```

Де аргумент classes визначає перелічені через кому вхідні класи; multicalsses визначає необхідність завантаження одразу декількох класів у різні теки в різних програмних потоках; type_csv визначає тип списку зображень для завантаження, з типом CSV; limit визначає максимальну кількість завантажених зображень.

Лістинг 3.3. Docker Image пропонованого сервісу завантаження анатованих зображень.

```
images-download-service:
  build: ./services/images-download
  image: getr-network:images-download-service
  environment:
    PYTHONUNBUFFERED: 1,
    PYTHONIOENCODING: UTF-8
  volumes: [ './services/images-download:/app' ]
  expose:
    - '6000'
  networks:
    - getr-network
```

На виході, як результат автоматизованого завантаження анотованих даних генеруються файли з координатами об’єктів на зображені для тестувальної та еталонної вибірки, які в подальшому передаються до наступного модуля – сервіса масштабованного тренування. Такі дані необхідні для тренування ЗНМ на базі навчання з учителем.

3.2.3. Засіб масштабованого тренування моделі нейронної мережі Yolov4.

Як зазначалось у пункті 3.1, процес тренування був розділений на два основних етапи: етап генерації базових вагових коефіцієнтів та етап поліпшення (fine-tuning) вагових коефіцієнтів для визначення найбільш оптимальних ознак при різних розмірах розпізнаних об’єктів [7].

Процес fine-tuning полягає у порівнянні значення mAP вагових коефіцієнтів отриманих на першому етапі з заданим пороговим максимальним значенням. Алгоритм fine-tuning розробленої моделі ЗНМ наведений нижче (алгоритм 3.1).

Алгоритм 3.1. Алгоритм fine-tuning на етапі тренування для розробленої моделі ЗНМ.

1. Визначити значення ітерації $\boldsymbol{i}$, яка збільшується на 1 кожну треновану вибірку;
2. Визначити $\boldsymbol{mAP_i}$, $\boldsymbol{LR_i}$ для кожного $\boldsymbol{i}$-го + 1 етапу тренування;
3. Визначити значення вагового коефіцієнта $\boldsymbol{w_i}$ для кожного $\boldsymbol{i}$-го + 1 етапу тренування;
4. Визначити порогове значення ***LRP*** (Learning Rate Policy) рівним 1500;
5. Визначити значення ***dc*** як decay*;*
6. Визначити ***degrade*** як початковий етап деградації ***LR***;

7. Визначити ***best*** як найбільше значення ***mAP*** для кожного ***i***-го етапу тренування*;*
8. *Я*кщо значення $\boldsymbol{best_0}$ більше ***degrade***;
9. Виконати тренування як $\boldsymbol{LR_1 = dc^2 \times LR_0}$ для порогу $\boldsymbol{best_1 + LRP}$;
10. *Я*кщо значення $\boldsymbol{mAP_0}$ менше $\boldsymbol{mAP_1}$ вибрати $\boldsymbol{w_1}$ інакше $\boldsymbol{w_0}$;
11. Інакше;
12. Виконати тренування як $\boldsymbol{LR_1 = dc \times LR_0}$ для порогу $\boldsymbol{best_0 + LRP}$;
13. *Я*кщо значення $\boldsymbol{mAP_0}$ менше $\boldsymbol{mAP_1}$;
14. Виконати тренування як $\boldsymbol{LR_1 = dc^2 \times LR_0}$ для порогу $\boldsymbol{best_1}$ + **LRP**;
15. *Я*кщо значення $\boldsymbol{mAP_1}$ менше $\boldsymbol{mAP_2}$ вибрати $\boldsymbol{w_2}$ інакше $\boldsymbol{w_1}$;
16. Інакше;
17. Вибрати $\boldsymbol{w_0}$;
18. Отримати фіналізований fine-tuned ваговий коефіцієнт ***w***.

Таким чином, фіналізоване значення $\boldsymbol{w}$ обирається шляхом послідовного порівняння найбільших значень в залежності від значень ***LR*** та ***dc*** визначених на першому етапі тренування. Загальна кількість ітерацій для вибору найліпшого вагового коефцііента залежить від порогового значення ***LRP***, яке зазначається вручну перед початком тренування.

Процес тренування моделі на базі ЗНМ Yolov4 виконується у ізльованому контейнері, який доступається до анотованих даних у локальній директорії поточної ОС.

Для прикладу, для паралелізації ресурсозатратних процесів навчання, доречно застосувати апаратні можливості на базі GPU з фреймворком паралельних обчислень CUDA 11.

Для навчання вхідної вибірки для ЗНМ розмірністю 512x512 пікселів з трьома вихідними шарами ЗНМ, використано відеокарту RTX 2070. На Рис. 3.5 зображено процес тренування з результуючим значенням **mAP** ваговоого коефіцієнта.

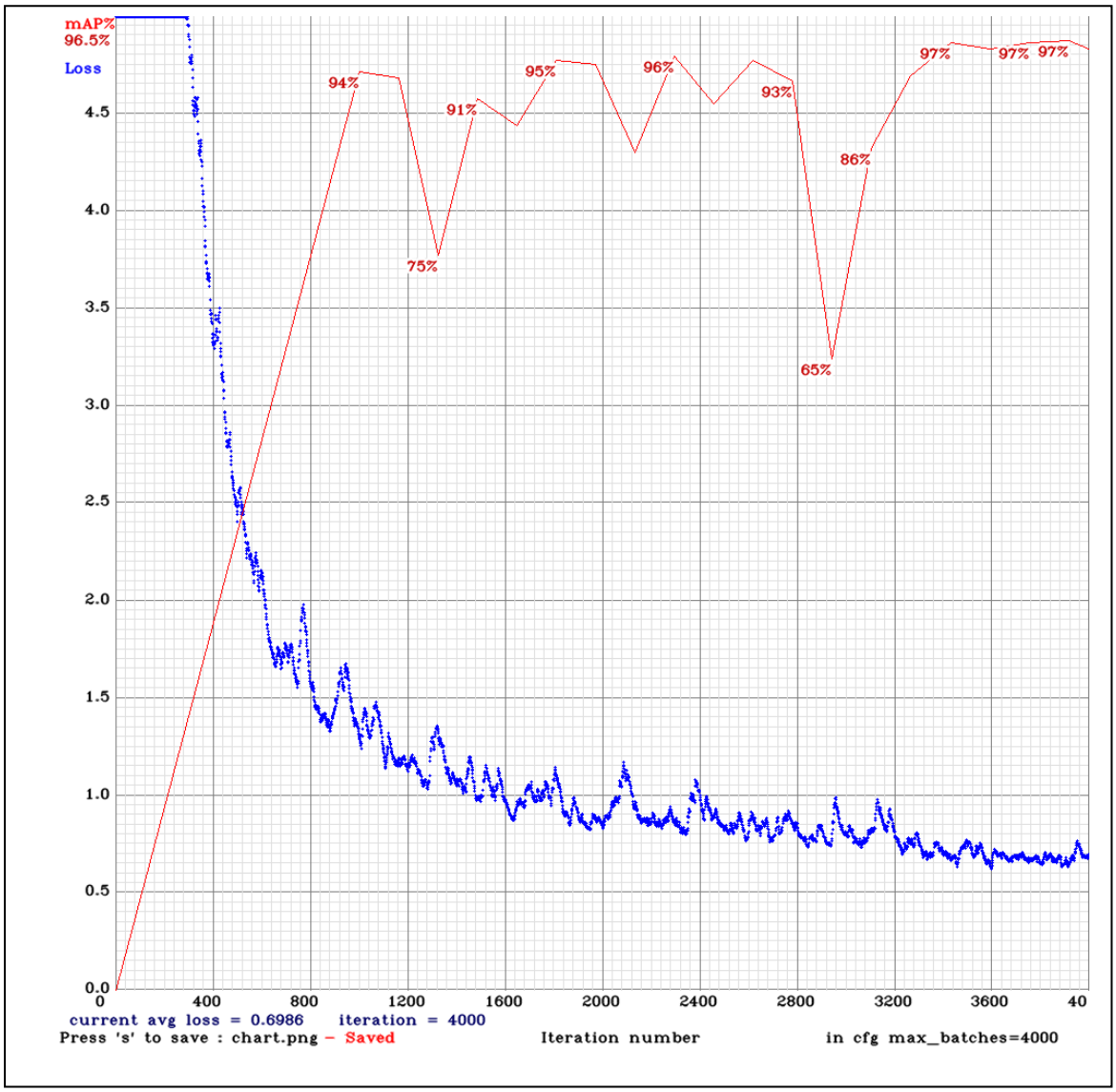


Рис 3.5. Процес тренування пропоонованої моделі ЗНМ га базі Yolov4. Кількість ітерацій – 4000; максимальне значення mAP рівне 97%; рівень LOSS не опускається нижче 0.84; по осі x – кількість ітерацій тренування; по осі y – значення LOSS за час проходження ітерацій.

На рисунку 3.3 синя крива визначає рівень LOSS, тобто наскільки багато помилок виникає при навчанні і яи тренована модел деградує (крива йде угору) чи навчається у правильному напрямку (крива прямує донизу). За задане порогове значення 4000, найліпше значення LOSS визначено як 0.84. У той же час, червона крива визначає рівень середньої вибрікової влучності mAP, рівень якого залежить від кількості проведених ітерацій навчання (epochs).

Результатом кожної ітерації, в залежонсті від значення LRP, є ваговий коефіцієнт, який передається до наступного ізольованого контейнера разом з

необхідними метаданим.

Тренування проводилось на базі ОС Linux, де фреймворк для тренування darknet та засоби паралелізапції процесів CUDA були контейнеризовані у Docker контейнер.

У Лістингу 3.4 та 3.5 зазначено Docker Image та DockerFile для пропонованого засобу.

Варто зауважити, що локальні директорії, отримані з попереднього кроку, мають бути прокинути до докер контейнера через volumes. Для збереження приватності та безпеки від зовнішнього втручання до Docker композиції, були застосовані лише приватні порти для комунікаації між докер контейнерами. У даному випадку застосований приватний порт 7000 приватної мережі getr-network (Лістинг 3.4).

Лістинг 3.4. Docker Image пропонованого сервісу завантаження анатованих зображень.

```
train-service:
  build: ./services/train
  image: getr-network:train-service
  volumes:
    - ./services/train/dataset/data/obj:/darknet/data/obj
    - ./services/train/dataset/data/test:/darknet/data/test
    - ./services/train/dataset/data/test.txt:/darknet/data/val.txt
    - ./services/train/dataset/data/obj.txt:/darknet/data/train.txt
    - ./services/train/dataset/data/yolo-obj.names:/darknet/data/yolo-obj.names
    - ./services/train/dataset/obj.data:/darknet/obj.data
    - ./services/train/dataset/yolo-obj.cfg:/darknet/yolo-obj.cfg
    - ./services/train/dataset/backup:/darknet/backup
  expose:
    - '7000'
  networks:
    - getr-network
```

Лістинг 3.5. DockerFile пропонованого сервісу завантаження анатованих зображень.

```
FROM nvidia/cuda:10.2-cudnn8-devel-ubuntu18.04
ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update
RUN apt-get -y install nano sed wget git

# Встановлення залежностей OpenCV
RUN apt-get install -y build-essential cmake git pkg-config libgtk-3-dev \
    libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
    libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev

# Завантаження OpenCV
RUN mkdir /opencv_build
WORKDIR  /opencv_build
RUN git clone https://github.com/opencv/opencv.git

# Компліяція OpenCV
RUN mkdir /opencv_build/opencv/build
WORKDIR  /opencv_build/opencv/build
RUN cmake -D CMAKE

RUN make -j$(cat /proc/cpuinfo | grep processor | wc -l)
RUN make install

# Отримання посилання на віддалений репозитаррій YOLO
WORKDIR  /
RUN git clone https://github.com/dmytro-kushnir/darknet.git
WORKDIR  /darknet

# Генерація Makefile
RUN sed -i '1 s/GPU=0/GPU=1/' Makefile
RUN sed -i '2 s/CUDNN=0/CUDNN=1/' Makefile
RUN sed -i '3 s/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
RUN sed -i '4 s/OPENCV=0/OPENCV=1/' Makefile

# Компіляція фреймворку Darknet
RUN make -j$(cat /proc/cpuinfo | grep processor | wc -l)

# Завантаження вагових коефіцієнтів у робочу директорію
WORKDIR  /darknet
RUN wget https://github.com/dmytro-kushnir /darknet/releases/download/yolov4/yolov4-tiny.conv.29
RUN wget https://github.com/dmytro-kushnir /darknet/releases/download/yolov4/yolov4-tiny.weights

CMD [ "sleep", "infinity" ]
```

Для розпаралелення процесів, на вхідному пристрої повина бути встановлена система масштабування CUDA з бібліотекою cudnn для роботи з GPU. Далі доступ до CUDA отримується з Docker контейнера. При відстуності CUDA, тренування може бути запущене на CPU.

3.2.4. Засіб конвертування моделі ЗНМ Yolo у CoreML формат для мобільних пристроїв iOS.

Засоби конвертування моделі ЗНМ на МП визначають способи запуску таких моделей на специфічних МОС, таких як iOS. Як було визначено у аналітичному розділі, на основі аналізу літературних джерел, для конвертації на таку платформу доречно використати фреймворк CoreML.

Запропонована структурна схема засобу конвертування моделі ЗНМ у CoreML формат зображено на Рис. 3.6.

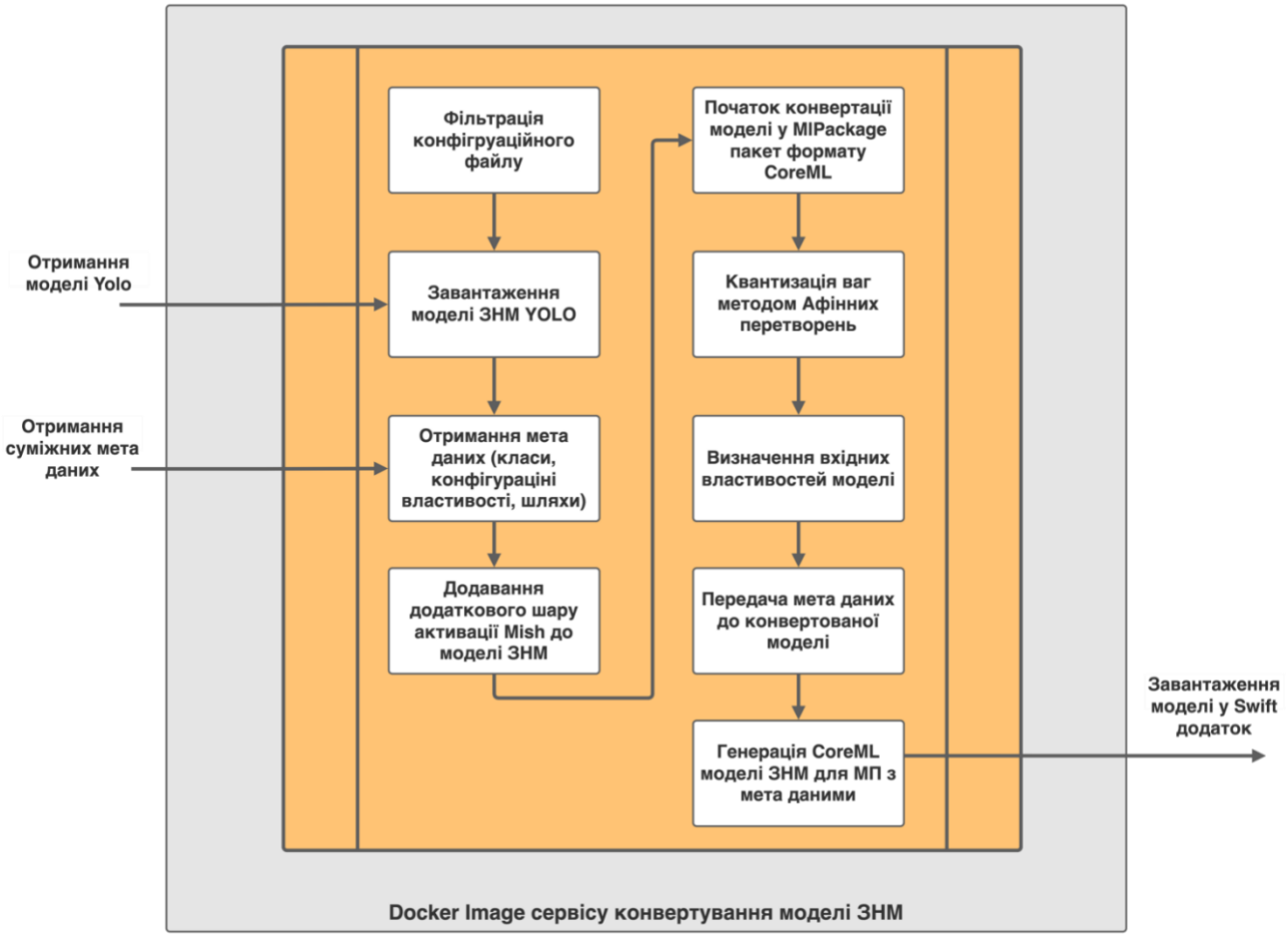


Рис. 3.6. Структурна схема засобу конвертування моделі ЗНМ у CoreML формат для МП.

Система масштабованого конвертування моделі використовує CoreML фреймворк з формуванням файлів MlPackage, які можуть бути зчитані МОС iOS.

Для поліпшення продуктивності моделі ЗНМ на МП з обмеженими апаратними можливостям, пропонується додатковий шар активації з функцією

Mish до моделі Yolo, а також провести квантизацію ваг методом афінних перетворень. Така комбінація методів та засобів дозволяє забезпечити швидкодію системи у реальному часі з довільною кількістю вхідних класів [4].

Згідно аналізу літературних джерел, обрано функцію активації Mish, яку можна додати як додатковий шар ЗНМ під час конвертування моделі. При цьому, проаналізовано ефективність таких функцій активації як Relu та Sigmoid (Рис. 3.7).

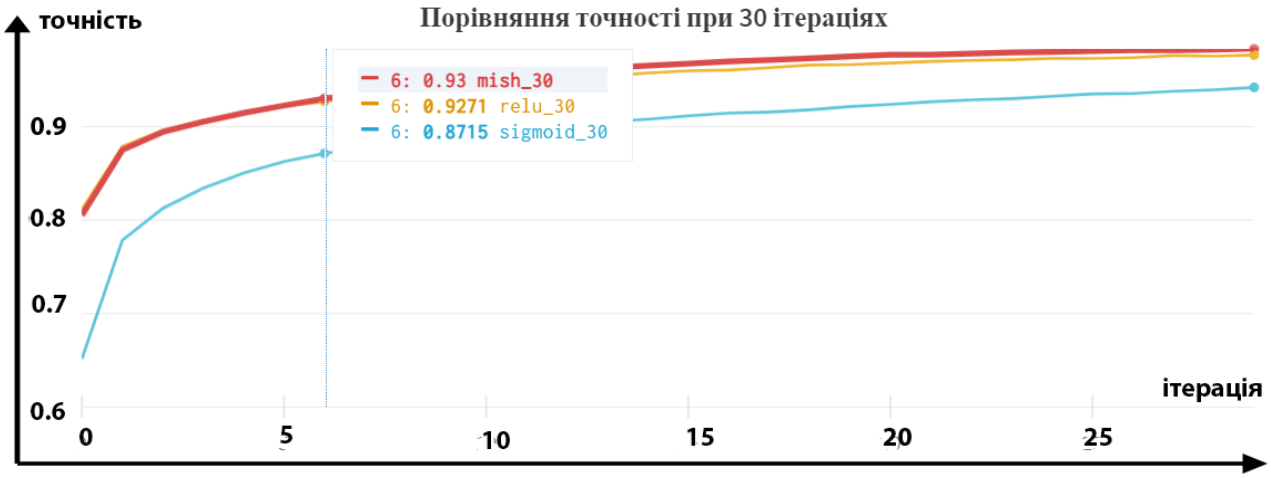


Рис. 3.7. Порівняння функції активації Mish з функціями Relu та Sigmoid. По осі x – кількість ітерації, по осі y – mAP (точність розпізнавання).

За результатами тестування, точність функції активації Mish на протязі 30 ітерацій становить 0.982, що є ефективнішим показником точності, у порівнянні з функціями активаціх Relu та Sigmoid [4].

Графічно, функція активації Mish представлена на Рис. 3.8.

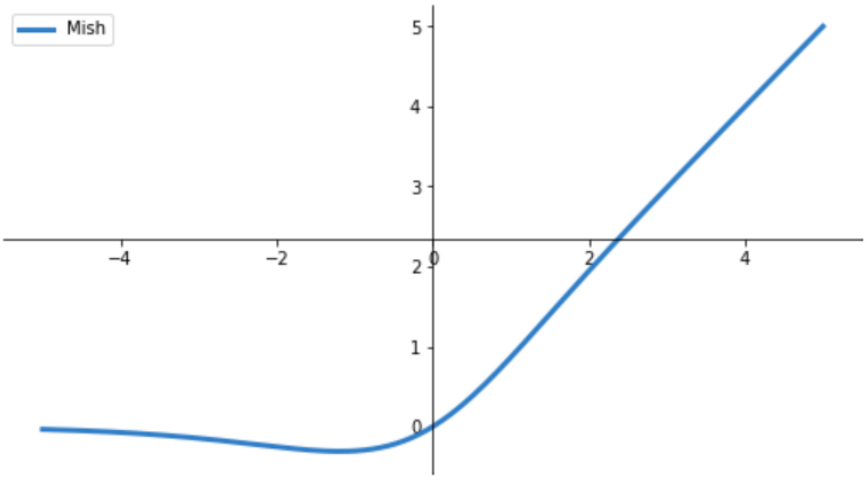


Рис. 3.8. Функція активації Mish для ЗНМ.

Функція, при використанні у прихованому шарі ЗНМ не є монотонною, при чому і похідна від неї також не є монотонною. Як результат, на виході отримується більш згладжений простір у порівнянні з ReLU. Це досягається також лінійною властивістю $f(x) = x$ при наближенні до нуля.

Для зменшення вагових коефіцієнтів моделі та збільшення її швидкодії доречно застосувати визначені методи квантизації. Пропонований метод квантизації за допомогою Афінних перетворень дозволяє зменшити точність до 8 біт. У той же час, метод квантизації з використанням таблиці пошуку, яка формується методом кластеризації к-середніх, може зменшити точність до 4 біт (Лістинг 3.6).

Лістинг 3.6. Лістинг засобу конвертування моделі та квантизації ваг методом афінних перетворень.

```
# Конвертування моделі ЗНМ  Yolo у CoreML формат. Точність розпізнавання – 32/16 чисел з рухомою комою
    model = ct.convert(yolo.model,
      inputs=[ct.ImageType(name='first_layer', scale=1/255)],
      minimum_deployment_target=ct.target.iOS15,
      compute_precision=ct.precision.FLOAT32 if args.precision_float32 else ct.precision.FLOAT16,
      compute_units=ct.ComputeUnit.ALL)

    if args.affine_quantize :
       # Компресія моделі до точності 16 біт для чисел з рухомою комою
       model = ct.compression_utils.affine_quantize_weights(model, mode="linear")
    elif args.k_means_quantize :
       # Компресія моделі до точності 16 біт для чисел з рухомою комою
       model = ct.compression_utils.palettize_weights(model, nbits=4, mode="kmeans")
```

У результаті, утворюються ваги з точністю у 8 біт/4 біти для чисел з рухомою комою, що зменшить розмір моделі у 2/4 рази. При цьому якість розпізнавання також знизиться лінійно, тому для малих моделей, з менше ніж трьома вихідними шарами ЗНМ, квантизація не є необхідним засобом при адаптації моделі до МП. У такому разі цей крок може бути пропущений.

При цьому, точність розроблюваної моделі може бути збільшена вдвічі: з числа половинної точності у 16 біт до числа одинарної точності 32 біти. Проте у цьому випадку значно збільшуються апаратні затрати при виконані задач РО. У розділі IV буде проаналізовано вплив зменшення/збільшення точності при процесі квантизації.

У Лістингу 3.7 та 3.8 зазначено Docker Image та DockerFile для пропонованого засобу конвертування.

Лістинг 3.7. Docker Image пропонованого сервісу конвертування моделі ЗНМ у CoreML формат для МП [3].

Для масташбування процесу конвертації, було задано вхідні параметри

```
conversion-service:
  build: ./services/conversion
  image: getr-network:conversion-service
  environment:
    PYTHONUNBUFFERED: 1,
    PYTHONIOENCODING: UTF-8
    MODEL_NAMES: './input_models/${MODEL_FILE_NAME}/${MODEL_FILE_NAME}.names'
    MODEL_CFG_PATH: './input_models/${MODEL_FILE_NAME}/${MODEL_FILE_NAME}.cfg'
    MODEL_CFG_TEMP_PATH:
'./input_models/${MODEL_FILE_NAME}/${MODEL_FILE_NAME}_tmp.cfg'
    MODEL_WEIGHTS_PATH:
'./input_models/${MODEL_FILE_NAME}/${MODEL_FILE_NAME}.weights'
    MODEL_COREML_PATH:
'./output_models/${MODEL_FILE_NAME}/${MODEL_FILE_NAME}.mlpackage'
  volumes: ['./services/conversion:/app']
  expose:
    - '5000'
  networks:
    - getr-network
```

еталонної моделі, які конфігуруються користувачем. Таким чином, шлях до конфігруаційного файлу, назва згенерованих вагових коефіцієнтів Yolo з набором класів та шляхи до директорій можна задавати, в залежності необхідної у даний момент моделі ЗНМ.

Як результат, на виході отримується модель CoreMl з заданими вихідними шарами ЗНМ та метаданими, в які входять: якорі розпізнавання для кожного класу, назви класів, розміри зображень моделі ЗНМ, тип моделі.

Лістинг 3.8. DockerFile пропонованого сервісу конвертування моделі ЗНМ у CoreML формат для МП [3].

```
FROM python:3.7

COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN apt-get update
RUN apt install -y libgl1-mesa-glx
RUN pip install -U pip
RUN pip install -r requirements.txt

COPY . /app

ENTRYPOINT  sh ./prepare_config.sh ${MODEL_CFG_PATH} ${MODEL_CFG_TEMP_PATH} & \
        python ./convert.py -n ${MODEL_NAMES} -c ${MODEL_CFG_TEMP_PATH} -w
${MODEL_WEIGHTS_PATH} -m ${MODEL_COREML_PATH}
```

У подальшому, модель завантажується на МП платформу iOS за допмогою засобу розробки XCode.

3.3. Засіб інтеграції модуля відстеження на мобільну платформу iOS.

Вбудований модуль (ВМ) – це такий модуль, який дозволяє інтегрувати додатковий функціональний фрагмент у існуючу систему, без зміни самого вбудованого модуля.

Таким чином, при інтеграції додатового модуля у систему пошуку, розпізнавання та відстеження об’єктів, виконується декілька принципів дизайну програмного забезпечення SOLID [125].

Серед них варто виділити наступні принципи:

- Принцип інверсії залежностей (Dependency inversion). Полягає у абстрагуванні від імплементації конкретного ВМ при будуванні системи. Таким чином, досить легко замінити існуючий модуль без значної зміни архітектури на вищому рівні;
- Принцип єдиного обов’язку (Single responsibility). Полягає у інкапсуляції певних обов’язків системи. В результаті, така задача як відстеження рузомих об’єктів виконується в ізольованому ВМ.

На основні результатів досліджень у II розділі, розроблено 3 алгоритми відстеження об’єктів у реальному часі. Для забезпечення швидкості інтеграції та тестування таких алгоритмів, доречно інтегрувати ізольовані ВМ для задачі відстеження у систему РО на МП iOS. При цьому сам модуль може бути написаний на довільній мові прогамування.

Для виконання задач відстеження, була обрана мова програмування JavaScript. Ця мова програмування дозволить використовувати модуль не лише як ВМ, але і як окремий сервіс у випадку використання системи як веб-застосунку у браузері.

Для інтеграції модуля відстеження у розроблювану систему була використана бібліотека JavascriptCore на мові Swift 5. Перевагою такої бібліотеки є використання ізольованого віртуального середовища Javascript. Варто зазначити, що Javascript працює лише в одному програмному потоці, який називається event loop, що значно обмежує апаратну продуктивність системи. Низька продуктивність є прийнятно для веб-браузера, проте для системи на МП iOS критично важливим є забезпечити високу швидкодію при великому навантаженні на CPU та GPU.

Як рішення цієї проблеми, пропонується використання багатопоточної системи мови програмування Swift 5. Для кожного потоку виділяється спільний доступ для сутності (instance) JavascriptCore, яка називається JsContext, та окрему чергу асинхронних повідомлень. Таким чином, роботу відстеження можна розділити між декількома наявними програмними потоками. Також, для зменшення впливу процесу відстеження об’єктів на продуктивність пристрою, доречно використати засоби кешування ідентичних об’єктів.

У алгоритмі 3.2 зображено процес ініціалізації сутності JsContext та використання її як окремого ВМ для відстеження об’єктів.

Алгоритм 3.2. Алгоритм інтеграції ВМ для відстеження рухомих об’єктів на МП засобами мови програмування Swift 5 та бібліотеки JavascriptCore.

---

1. Визначити **BatchN** як граничне значення опрацювання запитів для ВМ для одного асинхроного потоку;
2. Ініціалізація класу **JsRunner**;
3. Створення єдиного спільного глобального контексту **JsCore**;
4. Виконання ініціалізації;
5. Визначити допоміжних функції у глобальному об’єкті;
6. Визначити шляхи до ВМ для відстеження об’єктів та допоміжних скриптів;
7. Підключити ВМ до відповідних класів системи. Дати доступ до глобального контексту **JsCore** через клас **JsRunner**;
8. **Допоки** працює головний потік програми;
9. Якщо зарежстрований виклик метода відстеження рухомих об’єктів **formTrackedBoundingBoxes**;
10. Тоді виділити окремий аснихроний потік під обраний алгоритм відстеження для кожної **BatchN** вхідних запитів: **DispatchQueue.main.async { [self] in**;
11. Передача вхідних даних до ВМ для відстеження об’єктів;
12. Обробка даних модулем, та повернення результату;
13. Закриття завершеногоо потоку;
14. Завершення роботи головного потоку.

Як результат, асинхроно отримується набір вихідних даних з ВМ, який може бути опрацьований системою на МП iOS.

При створенні ВМ, доречно мінімізувати розмір вихідного модуля, оскільки розмір інтегрованих модулів може суттєво впливає на МП з обмеженими апаратними можливостями. Для виконання цієї задачі доречно використати засіб Rollup. Мова Javascript підтримує декілька патернів проєктування модулів, серед них варто виділити: CJS (commonJS), який переважно використовується у backend аплікаціях; ESM (EcmaScript Modules), для фикористанні у фронтенд апплікаціях; UMD (Universal Module Definiion), який використовує анонімні функції для обгортки та ізоляції окремих модулів. Працює на усіх апплікаціях.

Для досліджень був обраний патерн створення модулів UMD, оскільки він

працює на будь якій платформі. В результаті, кожен згенерований UMD модуль працює в ізольованому середовищі, та може бути успішно інтегрований на МП iOS.

Висновки до розділу третього розділу.

Визначено вхідні гіперпараметри та функцію втрат на етапі тренування. На першому етапі тренування було обрано функцію втрат для локалізації $\boldsymbol{L_{localization}}$ на базі CIOU. У цьому випадку значно зменшується нормалізована відстань між двома аналізованими об'єктами.

Для оптимізації вхідних гіперпараметрів, був обраний алгоритм градієнтного спуску NAG, з використанням оптимізацій градієнту Momentum та Decay.

На другому етапі тренування визначено алгоритм поліпшення (fine-tuning) вихідних вагових коефіцієнтів нейронної мережі. Таким чином, фіналізоване значення вагового коефіцієнта w обирається шляхом послідовного порівняння найбільших значень в залежності від значень *LR* та *dc* визначених на першому етапі тренування

Згідно з обраними засобів масштабування та контейнеризації Docker, побудована структура системи автономного анотування, тренування та конвертації моделі на МП iOS. З даної структури можна виділити Docker контейнери для кожного модуля/сервісу, які використовують масштабовані апаратні можливості операційної системи. Описано взаємозалежності між кожним елементом такої системи.

Засіб автоматизованого завантаження анотованих даних визначає вхідні набори анотованих зображень, відповідно до типів вхідних класів ЗНМ. Автоматизовано завантажує вхідні набори тренованої та тестованої вибірки з публічних баз даних, таких як OpenImage 7.0. На виході формує анотовані класи для кожного зображення у прийнятному форматі. У цьому дослідженні використовується формат моделі ЗНМ Yolov4. Також за потреби є можливість додати додаткові анотовані зображення за допмогою утиліти LabelImg.

Засіб масштабованого тренування моделі ЗНМ отримує вхідні анотовані дані

для подальшого масштабованого тренування як з використанням CPU, так і GPU в залежності від наявних апаратних засобів. Підтримує усі наявні операційні системи на базі Unix (Ubuntu 20.04, MacOs Monterey та інші) а також ОС Windows 7, 10, 11. На виході після тренування на анотованих даних з заданими гіперпараметрами, функцією втрат та методом кластеризації, отримуються фіналізовані вагові коефіцієнти та аналітичні значення аналізу ефективності тренування, такі як mAP.

Засіб конвертування моделі ЗНМ Yolo у CoreML формат для мобільних пристроїв iOS отримує вихідні вагові коефіцієнти у форматі моделі ЗНМ Yolov4. Виконує квантизацію методом афінних перетворень для оптимізації розміру вагового коецієнта. Записує згенеровані за допомогою кластеризації якорі розпізнавання та інші додаткові характеристики ЗНМ у метадані для подальшого опрацювання аплікацією. Засобами CoreML, відбувається конвертація моделі ЗНМ у MLPackage формат для завантаження на МП iOS за допомогою засобів мови програмування Swift 5 та вбудованих засобів системи розробки Xcode.

Запропоновано засіб інтеграції ВМ для відстеження рухомих об’єктів на МП iOS. Інтеграція полягає у використанні бібліотеки JavaScriptCore для передачі даних між системою та модулем. Також інтегровано багатопоточну систему обміну інформацією для зменшення навантаження на ВМ. Для створення ВМ відстеження рухомих об’єктів використано патерн створення модулів UMD.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ СИСТЕМИ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА МОБІЛЬНІЙ ПЛАТФОРМІ

У розділі представлено розроблену архітектуру систем на МОС iOS та ОС Ubuntu та обґрунтовано вибір компонент таких систем. Представлено результати аналізу та апробації системи.

Результати розділу опубліковано в працях автора [□ – □].

4.1. Архітектура системи.

4.1.1. Обґрунтування вибору підсистем та їх процесів.

Перед системою пошуку та розпізнавання об'єктів у відеозображеннях у реальному часі на мобільній платформі постають декілька важливих обмежень. У першу чергу це обмеженість апаратних та програмних ресурсів таких систем. Тому такі операції як обробка вагових коефіцієнтів розроблених ЗНМ, пошук, розпізнавання та відстеження довільного класу об'єктів мають бути максимально ефективними, при використанні мінімальних ресурсів CPU та GPU.

На Рис. 4.1 зображено узагальнена схема пропонованих систем.

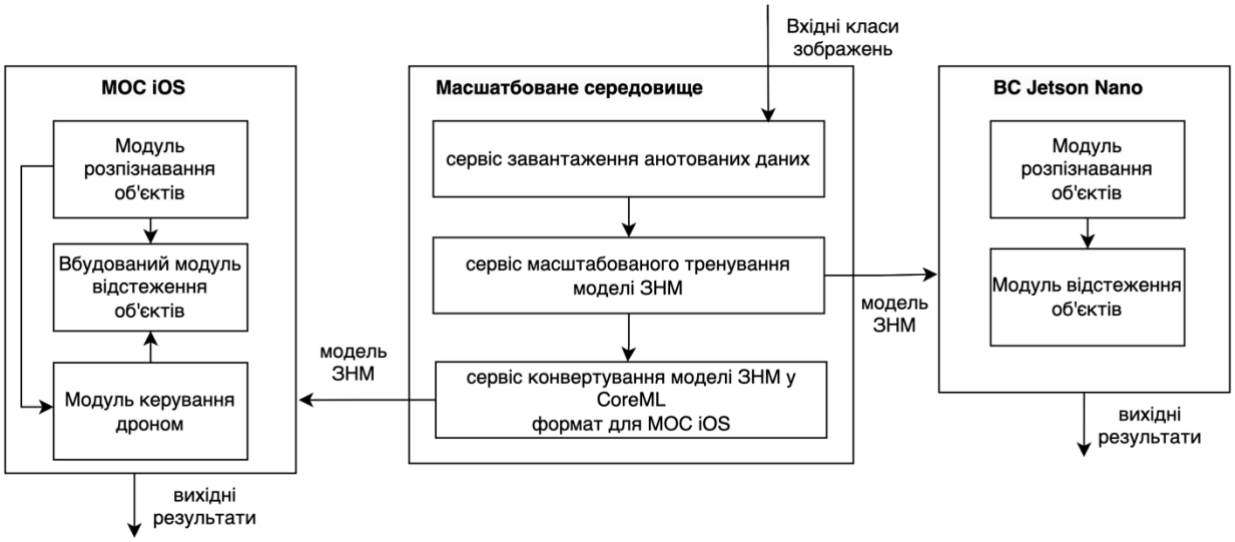


Рис. 4.1. Узагальнена структурна схема пропонованих систем.

Пропонується розробка 2 незалежних систем для пошуку, розпізнавання та відстеження об'єктів на мобільній платформі: на МОС iOS та ВС Jetson Nano на ОС

Ubuntu. Для МОС iOS додатково пропонується інтегрувати модуль зв’язку з дроном типу Tello для виконання задач розпізнавання та пошуку заданих об’єктів.

Для вирішення таких задач, на МОС iOS, був використаний фреймворк CoreML для обробки вагових коефіцієнтів ЗНМ для задач пошуку та розпізнавання об’єктів (пункт 4.1.2).

Для вбудованих систем на МП була використана система акселерації GPU CUDA на чипі Jetson Nano з процесором ARM Nvidia Carmel на базі ОС Linux (пункт 4.1.3). Це дозволить виконувати задачі пошуку та розпізнавання довільного класу об’єктів у межах реального часу, з доступом через веб браузер.

Для вирішення задач анотування вхідних зображень, тренування вагових коефіцієнтів ЗНМ, кластеризації зображень та конвертування моделі у CoreML формат для МП використано середовище масштабування Docker де кожен модуль/сервіс був контейнеризований як окреме середовище виконання процесів.

Для задачі відстеження рухомих об’єктів, для підтримування принципів SOLID, вирішено інтегрувати такі методи та засоби як окремий налаштовуваний Javascript модуль, який працює як для мобільних систем на платформі iOS, так і на вбудованій МП на базі ОС Linux.

Таким чином, розроблювана мобільна платформа – це середовище взаємодії з користувачем системи, де клієнт буде здатний:

- завантажувати вхідні фотографії та відеозображення, які потрібно опрацювати на наявність об’єктів для розпізнавання та відстеження;
- отримання та обробку вхідного відеопотоку у реальному часі з отриманням та аналізом кожного інформаційного кадру;
- налаштуванням роботи системи пошуку, розпізнавання та відстеження об’єктів, з можливістю вибору вхідних класів об’єктів розпізнавання та типу та виду вхідної ЗНМ; налаштування значень мінімізаційних та згладжуючих фільтрів;
- отримання та збереження вихідної статистики розпізнавання та відстеження об’єктів.

У той же для підготовки моделей ЗНМ використано PAAS платформу на базі

ОС Linux та середовища масштабування Docker. Для виконання визначених задач досліджень цілком достатньо використання локальної PAAS платформи без використання хмарних обчислень по типу SAAS платформ, таких як AWS та Google Collab. Основною перевагою використання локальної платформи є відсутність грошових витрат на дослідження в залежності від використаних апаратних ресурсів, а також відсутність потреби у постійному інтернет з'єднанні, що критично важливо, якщо розроблювана МП знаходиться поза межами доступу інтернет мережі.

4.1.2. Компоненти та підсистеми на базі мобільної платформи iOS.

Компоненти та потоки даних системи пошуку, розпізнавання та відстеження довільного класу об'єктів на МП iOS продемонстрована на Рис. 4.2.

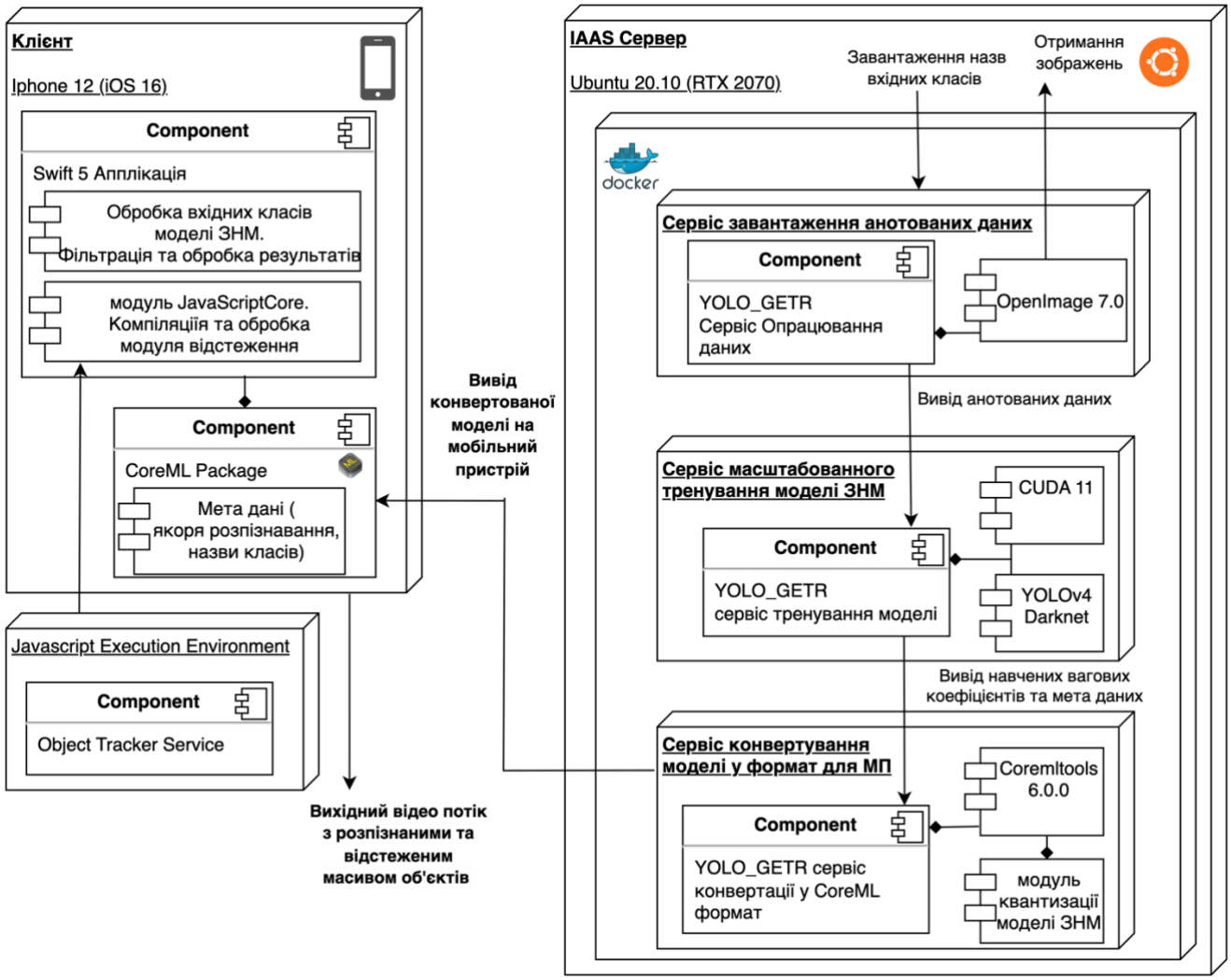


Рис. 4.2. Функціональні компоненти та потоки даних системи розпізнавання та відстеження на мобільній платформі iOS.

Систему варто розділити на дві основні частини: клієнт та сервер.

Сервер – це PAAS платформа на базі ОС Ubuntu 20.10, на якій запускається масштабоване середовище Docker у якому знаходяться контейнери для кожного сервісу. Задача сервера системи – автоматизоване завантаження анотованих даних з віддаленої бази даних OpenImage 7.0 або з локально сховища; тренування моделі ЗНМ за допомогою фреймворку Darknet та бібліотеки CUDA 11; конвертації отриманих вагових коефіцієнтів у MLPackage формат засобами coremltools 6.0.0. Модулі були реалізовані на мовах програмування Python, bash scripts та C. Усі модулі об’єднані у Docker композицію, для передачі даних між кожним сервісом. Як результат, до клієнту надсилається вихідні вагові коефіцієнти у форматі CoreML, а також необхідні метадані, такі як: назви класів поточної моделі ЗНМ, якорі розпізнавання після кластеризації, розмір ЗНМ.

Клієнт – це мобільний додаток на МОС iOS під назвою RETR (Recognizer and Tracker) для взаємодії з користувачем та виконанні задач пошуку, розпізнавання та відстеження довільного класу об’єктів у реальному часі. Клієнтський застосунок написаний мовою програмування Swift 5, з можливістю імпортування CoreML моделей у додаток. Для реалізації задачі відстеження, був імпортований JavaScript модуль відстеження засобами бібліотеки JavascriptCore. Зображення чи відеопотоки можуть бути отримані з галереї пристрою, або ж з вбудованої камери.

Розробка інтерфейсу користувача МОС RETR

Для розробки інтерфейсу користувача програми та визначення можливих кроків та дій користувача при використанні системи, використано підхід прототипування.

Цей підхід полягає у визначенні загальних сторінок (фреймів) аплікації та потоків даних між ними. Основна задача прототипування полягає у наступному:

- Описі основних фреймів системи та взаємодій між ними;
- Структурі подання інформації користувачу;
- Мінімальній можливій кількості кроків та фреймів для досягнення задач аплікації (розпізнавання та відстеження заданого класу об’єктів);
- Візуалізація те тестування системи.

На Рис. 4.3 зображено загальну структуру аплікації GETR побудовану за допомогою прототипування. Всього представлено 4 основних фрейми: фрейм роботи з дроном (UAV); фрейм роботи з галереєю та збереженими медіа, такими як фото та відео; фрейм отримання відеопотоку у реальному часі; фрейм налаштувань додатку.

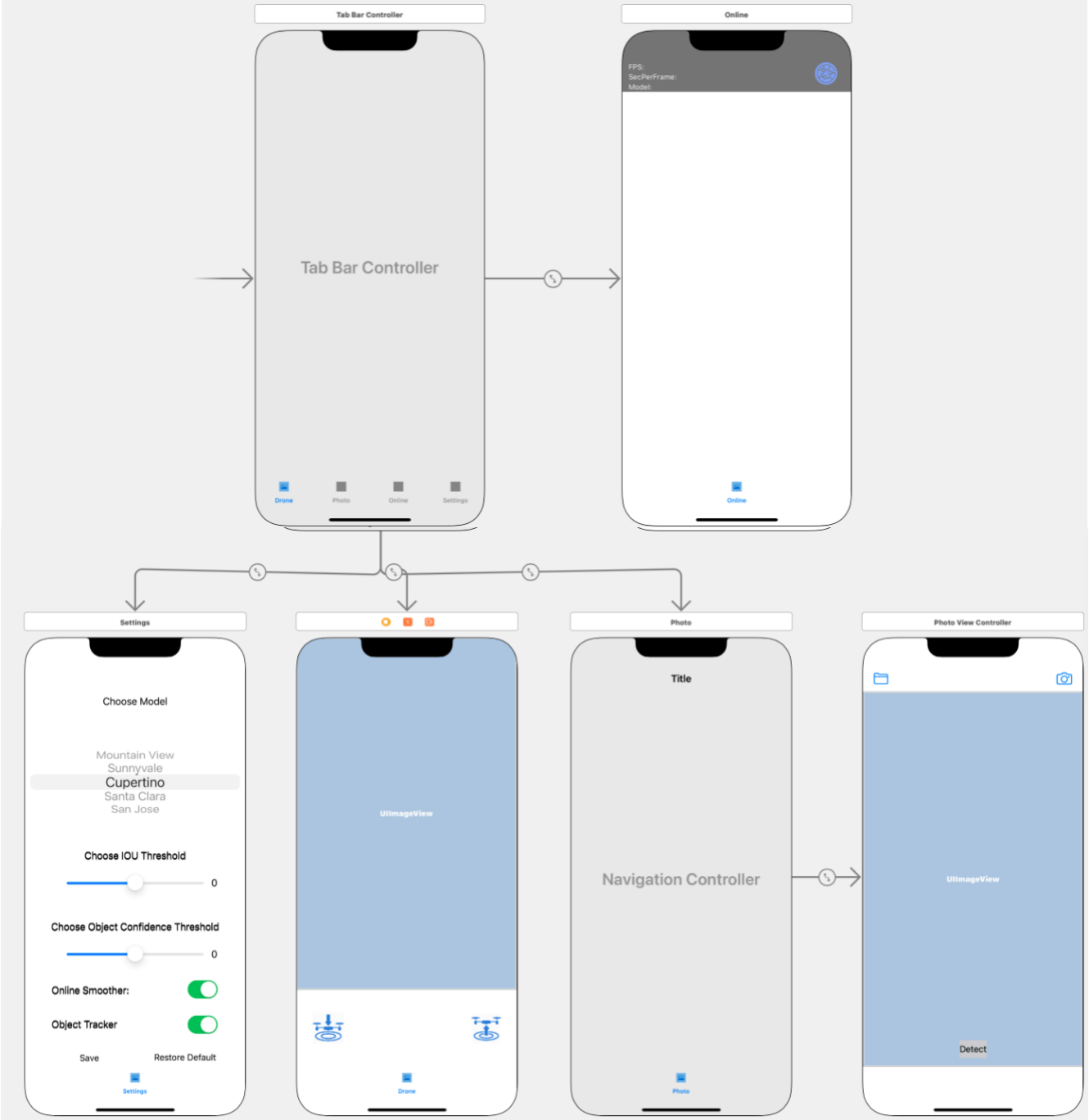


Рис. 4.3. Прототипування інтерфейсу користувача для розроблюваної МОС RETR на ОС iOS.

На початковому екрані отримується можливість отримати горизонтальний список (tab-bar) з усіх можливих фреймів з довільною послідовністю вибору

кожного з них. Першим фреймом показується фрейм дронів, який дозволяє запускати та керувати дроном, та отримати вхідний відеопотік з камери дрона. Другий фрейм це сторінка для обробки медіа даних, таких як фото та відео конкретного користувача. Третій фрейм це сторінка обробки відеопотоку з камери у реальному часі. До кожного функціонального фрейму у верхній половині екрану додається вікно з метаданими, такими як: FPS, назва поточної моделі ЗНМ, кількість кадрів опрацьованих за секунду, кількість розпізнаних об’єктів тощо.

Четвертий фрейм – це фрейм налаштувань, на якому можна: обрати вхідну модель ЗНМ з вже завантажених у додаток; кількісні значення мінімізаційних та згладжуючих фільтрів; запуск алгоритмів відстеження та функції затримки малювання кадрів.

Таким чином, сформовано інтуїтивно зрозумілий інтерфейс користувача з мінімальною кількістю кроків для виконання задач розпізнавання та відстеження об’єктів на МП.

Діаграма класів MOC RETR

Програмний додаток на мові Swift складається з багатьох функціональних сутностей, таких як класи, протоколи, інтерфейси та модулі (бібліотеки). На Рис. 4.4 зображена UML діаграма класів MOC RETR.

При початковому запуску процесу розпізнавання необхідно завантажити модель ЗНМ. При чому модель буде закешована у додатку для подальшого використання. Визначемо кожен з модулів системи.

Клас **ModelProvider** слугує хабом для завантаження CoreML моделі ЗНМ, та реалізації функцій передбачення результатів розпізнавання та відображення результатів розпізнавання. Побудований за принципами патерна проєктування Обсервер (Observer), тобто інші модулі системи підписуються на зміну подій, під час виконання процесу розпізнавання, та запускають відповідні методи.

Протокол **ModelProviderObserver** визначає підписану подію show для усіх функціональних фреймів. Таким чином, при отриманні результатів розпізнавання буде запущено відповідний метод show у поточному функціональному фреймі.

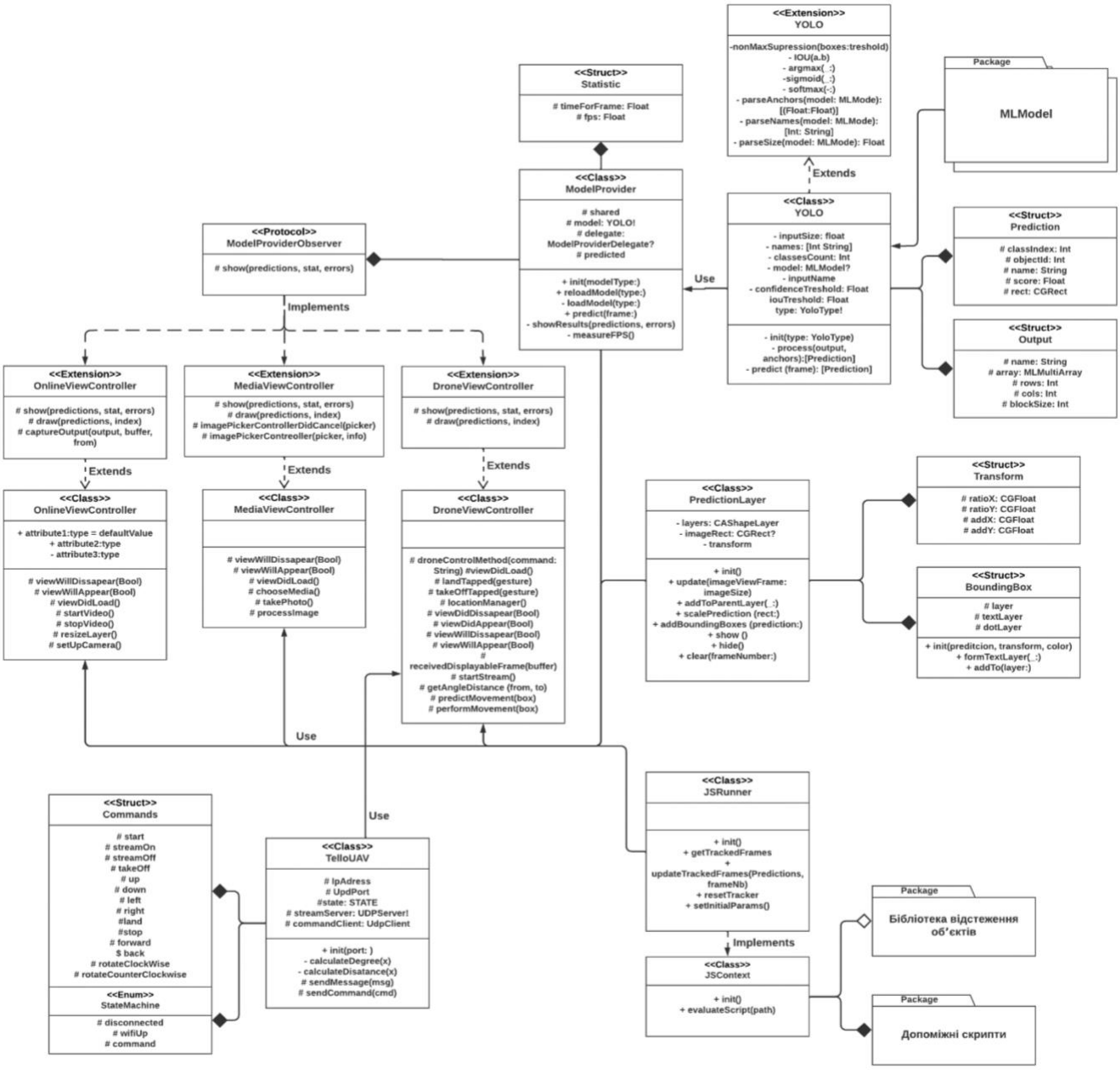


Рис. 4.4. UML діаграма класів МОС RETR на ОС iOS.

Клас **YOLO** визначає основні методи завантаження вагових коефіцієнтів та мета даних ЗНМ, обробку кожного шару ЗНМ на вході та виході та отримання результатів розпізнавання у форматі масиву розпізнаних об'єктів. Також визначаються вхідні та вихідні структури даних: Output та Prediction для передачі у **ModelProvider**. Функції мінімізації, а також фільтри зберігаються у розширеному наборі методів класу **YOLO** та застосовуються під час обробки кожного розпізнанного об'єкту. Приклад результуючої відповіді, яка записується в структуру **Prediction** продемонстровано на Рис. 4.5.

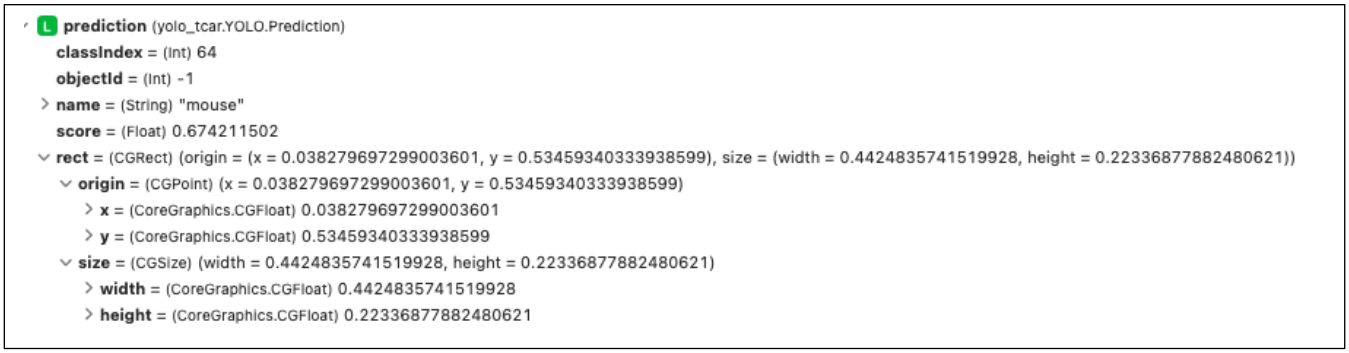


Рис. 4.5. Результуючі дані про розпізнаний об’єкта типу Mouse з метаданими про розташування та вірогідність розпізнавання об’єкта серед масиву вхідних класів.

Набір пакетів **MLPackage** визначає вхідні завантаженні ЗНМ для розпізнавання об’єктів. Приклад пропонованої моделі з двома вихідними шарами та збереженими метаданими продемонстрований на Рис. 4.6.

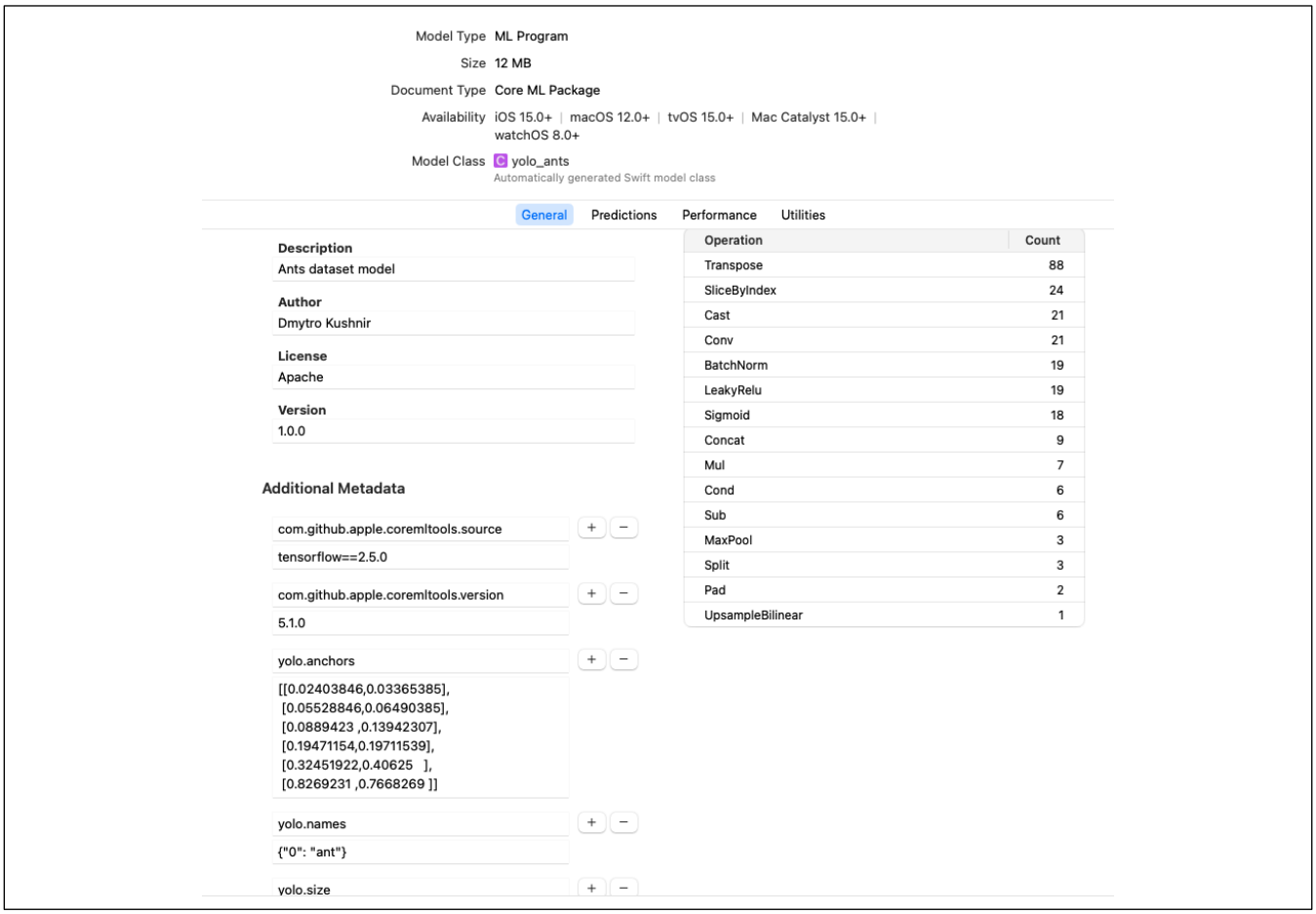


Рис. 4.6. Технічні характеристики CoreML моделі ЗНМ.

У даній інформаційній панелі середовища розробки Xcode визначаються розміри моделі, вхідні та вихідні шари, додаткові метадані, та точність опрацювання чисел з рухомою комою у випадку використання методу квантизації.

Перевагою пакетів CoreML (CoreML Package) над звичайною нейронною мережею є можливість додавання мета даних до моделі, що дозволяє значно розширити можливості додатку. Наприклад передача розмірів чи якорів розпізнавання конкретної моделі ЗНМ, дозволяє змінювати ці значення у мобільному додатку на льоту. Варто відзначити, що до аплікації можна додати довільну кількість моделей ЗНМ з довільною кількістю класів, що значно збільшує масштабованість застосунку. При цьому кожну з моделей можна обрати на фреймі налаштувань.

Клас **PredictionLayer** визначає яким чином мають бути відображені регіони розпізнавання на екрані, враховуючи коефіцієнт розміру екрану мобільного пристрою iOS. Також цей клас слугує для відмальовування унікальним кольором об’єкта розпізнавання, та відображення шляху руху цього об’єкта. При цьому до кожного об’єкта присвоюється його унікальний ідентифікатор, який відрізняє його від інших об’єктів, зображених на поточному інформаційному кадрі.

Клас **JSRunner** виконує функцію запуску контексту JavascriptCore – **JsContext**. У цьому контексті викликаються та запускаються відповідні методи імпортованої бібліотеки відстеження об’єктів.

Клас **TelloUAV** відповідає за контроль, асинхронну оброку відеопотоку, розпізнавання, наведення та відстеження на об’єкт. Цей клас включає структуру **Commands** в якій описано усі можливі команди, які відправляються по протоколу UDP до дрона (Рис. 4.7).

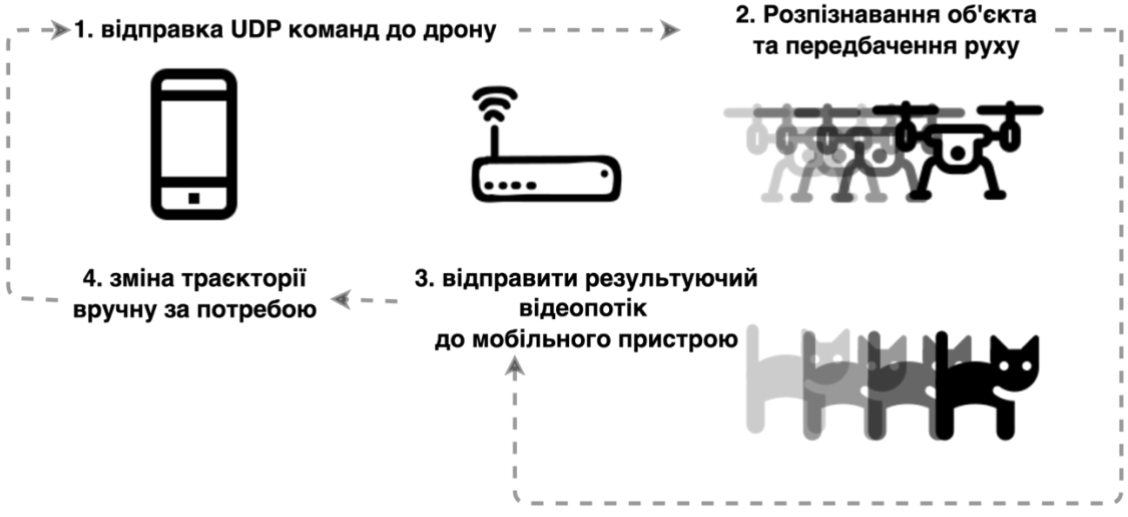


Рис. 4.7. Схематичне представлення роботи дрона за допмогою MOC RETR.

Управління дроном може вестить з аплікації за допомогою віртального джойстика. Серед команд можна виділити: запуск дрона, рух вперед, назад, проти годинникової стрілки, угору, початок відео трансляції на МП.

Класи **DroneViewController**, **MediaViewController**, **OnlineViewController** відповідають за відображення результатів розпізнавання та відстеження. Ці класи слугують стартовими точками для аплікації та взаємодії з користувачем. Також ці класи мають додаткові функції для з'єднання з модулями розпізнавання (**ModelProviderObserver**), відстеження (**JSRunner**) та додатковими утилітарними функціями для формування відеопотоку.

Дослідження проводились на наступних пристроях з ОС iOS: Iphone X та Iphone 12, які автоматично налаштовують CPU та GPU для задач розпізнавання об'єктів.

4.1.3. Компоненти та підсистеми на базі платформи Linux.

Специфіка операційної системи iOS та закритість її API шлюзів до втручання ззовні вимагає розробки більш уніфікованого рішення для операційних систем з відкритим кодом.

Для реалізації системи розпізнавання та відстеження об'єктів у відеозображеннях у реальному часі на мобільних платформах, відмінних від iOS, пропонується застосувати ті ж методи масштабування Docker для усіх модулів системи та ОС Linux для зберігання та виконання роботи сервісів системи (Рис. 4.8).

Система складається з вже реалізованих сервісів анотування зображень та тренування моделі ЗНМ, а також вбудованих модулів OpenDataCam для обробки та відображення результатів розпізнавання.

Система є повністю докеризованою, тобто вона може бути інтегрована у такі кібер-фічизичні системи, які базуються на чипах Jetson Nano з ОС Ubuntu 20.04 (Linux). У той же час вона може бути побудована та запущена на стаціонарному комп'ютері.

Дослідження проводились на вбудованому пристрої Nvidia Jetson Nano WaveShare та стаціонарному комп'ютері з графічною картою Ray Tracing Texel

eXtreme (RTX) 2070. Обидва пристрою працюють на ОС Ubuntu та мають підтримку бібліотеки CUDA. Також можливим є запуск системи на платформі Raspbery з ОС Raspbian.

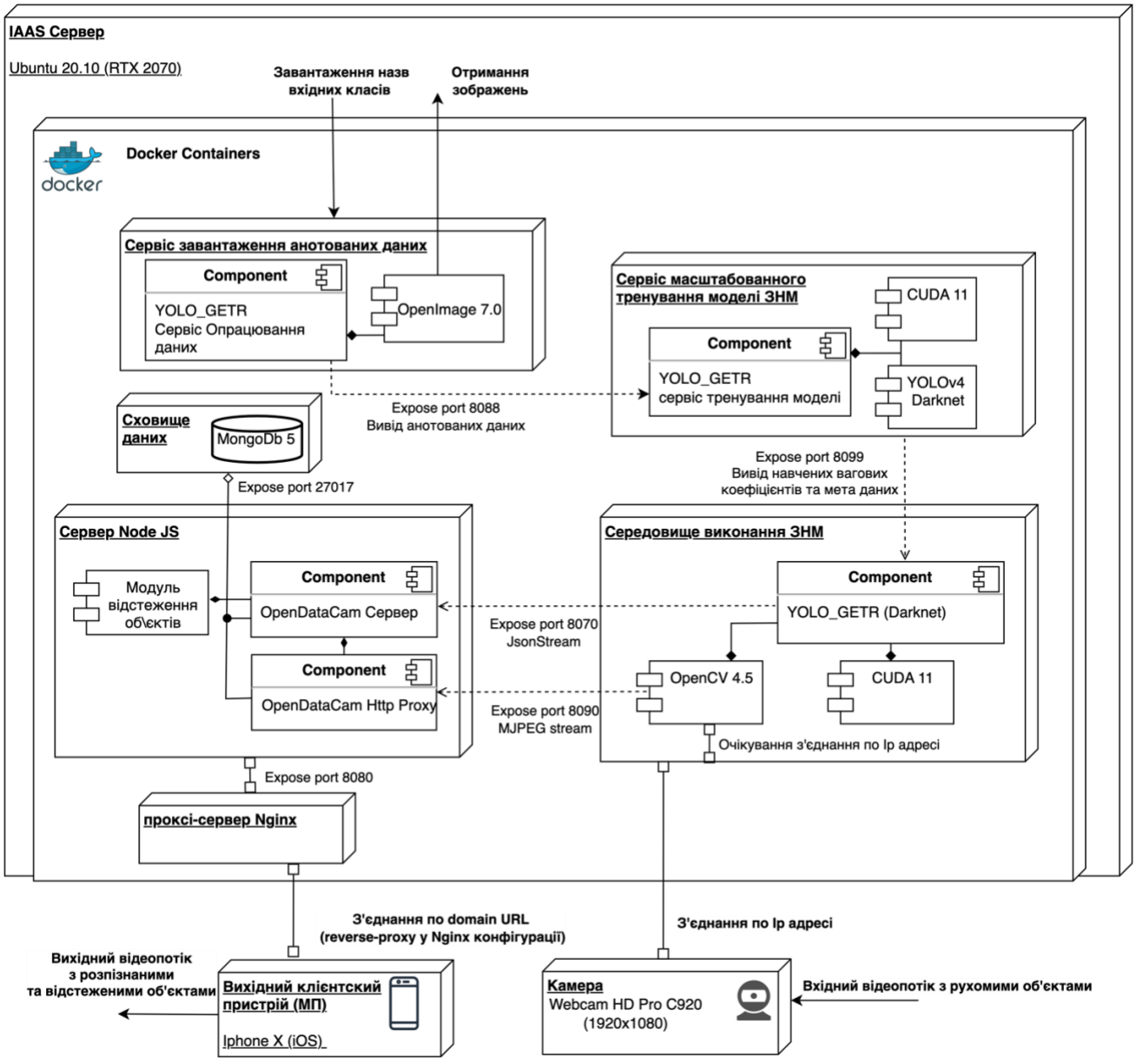


Рис. 4.8. Функціональні компоненти та потоки даних системи розпізнавання та відстеження на ОС Linux.

Система являє собою Docker композицію та складається з наступних компонент:

- **Сервіс завантаження анотованих даних**. Завантажує анотовані дані з віддлаеного сховища OpenImage або з локального сховища. Формує дані для сервіса тренування ЗНМ;

- **Сервіс масштабованого тренування моделі ЗНМ**. Сервіс тренування вхідної ЗНМ з отриманням вихідних вагових коефіцієнтів та мета даних;
- **Середовище виконання ЗНМ**. Докер контейнер для зберігати фіналізованих вагових коефцієнтів моделі ЗНМ на основі Yolov4 та фреймворку Darknet. У той же час бібліотека Compute Unified Device Architecture (CUDA) опрацьовує паралельні операції розпізнавання. Додатково, контейнер слухає зміни у вхідному відеопотоці з камери та перенеаправляє дані для розпізнавання за допомогою бібліотеки Open Source Computer Vision Library (OpenCV) до серверного модуля Node Js;
- **Сервер Node JS**. Серверна частина аплікації написана на фреймворку NodeJS, яка складається з декількох частин: OpenDataCam Сервера, Hypertext Transfer Protocol (HTTP) Proxy та модуля відстеження рухомих об’єктів;
- **OpenDataCam Сервер**. Сервіс, який аснихроно слухає повідомлення про надходження даних з **Середовища виконання ЗНМ** з відеопотоку JsonStrem через відкритий порт 8070. Ці дані обробляються та збергіаються у внутрішній нереляційній базі даних MongoDB через порт 27017. Наступним кроком, отримані інформаційні кадри х розпізнаними об’єктами з відеопотоку відправляються до модуля відстеження та аналізу розпізнаних об’єктів. Ці дані будуть збережені у Docker data volume бази даних, з можливістю збереження інформації при виключені пристрою. Після опрацювання кожного інформаційного кадру відеопотоку, він надсилається до клієнтської програми через порт 8080;
- **OpenDataCam HTTP Proxy**. Сервіс, який слугує стартовою точкою при отримані конфігурацій моделі ЗНМ та ініціалізації процесу запуску **Середовища виконання ЗНМ** Docker контейнера. Додатково, цей модуль запускає графічний веб інтерфейс, та передає вхідні дані за допомогою відеопотоку Motion Joint Photographic Experts Group (MJPEG) через порт 8090 до клієнтської аплікації;
- **Модуль відстеження об’єктів**. Імпортований модуль, який виконує алгоритм відстеження рухомих об’єктів. Оскільки це зовнішній модуль,

плгоритм відстеження може бути підлаштований чи змінений під конкретні задачі користувача.

- **Сховище даних**. Сховище для нереляційної бази даних MongoDb, у якому збережені основні дані відстеження, розпізнавання та необхідної аналітики робот системи;
- **Проксі сервер Engine–X (Nginx)**. Docker контейненр для переадресування відеопотоку з додатку до клієнтської аплікації. Використовує методику reverse-proxy для приховання сервверних портів та надсилання запитів на необхідний веб домен;
- **Вихідний клієнсткий пристрій**. Пристрій користувача з веб браузером для отримання інформації з сервера та роботи з системою. Оскільки веб браузер є практично на кожному клієнтському пристрою, такі системи можуть бути використані для задача РО на МП;
- **Камера**. Вхідний Сенсор КФС для запису відеопотоку для задач розпізнавання та відстеження. У дослідженні використано камеру Webcam HD Pro C920 на стаціонарному комп'ютері та Waveshare IMX477-160 на системі Jetson Nano.

Основна задача масштабованої системи – можливість розгоротання сервісів на різних видах МОС, для виконання задачі розпізнавання та відстеження об'єктів з довільною кількістю вхідних класів, тобто різними видми вхідних моделей ЗНМ. Такі системи в подальшому можуть бути модифіковані для роботи у хмарному середовищі. Також системи аналітики розпізнавання та відстеження об'єктів може бути розширена в залежності від потреб користувача.

4.2. Апробація та аналіз результатів.

Дослідження та аналіз результатів роботи імплементованих систем розпізнавання та відстеження об'єктів у відеозображеннях у реальному часі на мобільних платформах варто розпочинати з визначення ключових проблем, які постають перед такими системами. Додатково, для аналізу ефективності та продуктивності розроблених методів та засобів, необхідно структурувати

попередньо визначені у дисертаційному досліджені метрики оцінки.

Визначено, що основними проблемами, які постають перед такими системами є: визначення ефективності моделі ЗНМ для задач розпізнавання об’єктів, визначення продуктивності розробленої моделі ЗНМ на мобільній платформі враховуючи наявні апаратні та програмні засоби та визначення ефективності розроблених методів відстеження об’єктів. Для досліджень були обрані визначені формули 2.1 – 2.9.

4.2.1. Ефективність виконання розробленої ЗНМ у задачах РО.

Основні метрики при виконанні дослідження: ***R*** (recall/чутливість), ***P*** (precision/влучність), **TP** (істинно позитивне значення), **FN** (хибнопозитивне значення), **F1** (F-міра), розмір вагових коефіцієнтів **w**, **mAP** (середня вибіркова влучність).

Також визначено основні характеристики згладжуючих та мінімізаційних фільтрів **ЗФС (**Згладжуючий фільтр стиснення) та **IOU**, методів кластеризації **K-means/K-means++, ФРК** (фільтр розміру кадру), розмір тренувальної вибірки, вхідна роздільна здатність зображень ЗНМ.

Для цілей дослідження навчено та реалізовано 4 основні ЗНМ типи Yolov4. Їх можна розділити по:

- Кількості вихідних шарів: 2 (tiny model) та 3 (звичайна модель);
- Максимальній роздільній здатності вхідних зображень: 512 та 416 пікселів;
- Використання методу кластеризації: поліпшений метод k-means++ чи k-means;
- Кількість класів визначена як 40 для стандартної моделі yolov4 та 6 для моделі getr;
- Застосованих фільтрах для виконання задачі висновування з такими значеннями для усіх моделей **ЗНМ**: **ФРК** був визначений як 3024 x 1964 пікселів для екрана тестованого пристрою MacBook Pro 14;

  Розмір тренувальної вибірки був сталий і становив 424 зображення для тренування та 184 для тестування. **ЗФС** був визначений як 0.9 та **IOU** був визначений як 0.2 для мінімізації малоймовірних результатів.

Результати тестування ефективності роботи розроблених моделей ЗНМ для задач РО наведені у Таблиці 4.1.

Таблиця 4.1. Метрики ефективності роботи розроблених моделей ЗНМ для задач РО в залежності від обраного методу кластеризації, роздільної здатності вхідних зображень ЗНМ та кількості вихідних шарів.

Метрика / Модель ЗНМ	R (%)	P (%)	FN	TP	F1 (%)	w (МБ)	mAP (%)
Yolov4_getr_416 (k-means++)	**91.9**	97.4	**27**	**272**	**96.77**	256	**97.2**
Yolov4_getr_512 (k-means++)	**93.7**	98.2	**19**	**284**	95.32	256	**96.2**
Yolov4 _416 (k-means)	91.3	**99.2**	32	267	91.87	246	94.2
Yolov4 _512 (k-means)	92.2	**98.1**	18	277	**93.787**	246	93.6
Yolov4_getr_tiny_416 (k-means++)	**86.1**	90.4	41	**268**	**91.5**	24.2	**86.92**
Yolov4_getr_tiny_ 512 (k-means++)	**87.4**	92.5	**38**	261	**91.8**	25.2	82.99
Yolov4_tiny_416 (k-means)	81.5	**95.4**	42	256	88.6	24.3	81.21
Yolov4_tiny_512 (k-means)	82.2	**97.2**	**39**	**262**	87.2	25.1	**83.11**

У тесті використовувалось 4 імплементовані моделі ЗНМ та 4 стандартні моделі ЗНМ такого ж типу для порівняння. У випадку використанні алгоритму кластеризації k-means++, на етапі формування якорів розпізнавання, згенерована модель показує ліпші результати **mAP** на 5%. При чому можна спостерігати що значення **P (**влучність**)** та **R** (чутливість/точність) взаємно обмежують один одного: чим вище значення **R**, тим меншим буде значення **P** у лінійній прогресії. Таким чином, значення **R** при використанні методу кластеризації k-means++ збільшується в середньому на 3-4%, а значення **P** зменшується на 2-3%. Метрика **F1** призначена для балансування значень **R** та **P.** Отже, чим вищий показник **F1,** тим вища фінальна точність моделі. В загальному, значення показнику **F1** покращилось на 5-6% для більшості вхідних моделей.

Кількісні значення помилок **FN** та **TP** є лінійними та залежать від ефективності роботи ЗНМ при тестуванні. У той же час для моделей з 2 вихідними шарами (tiny) кількість помилок **FN** є дещо більшою у порівнянні з аналогами з 3 вихідними шарами (Рис. 4.9).

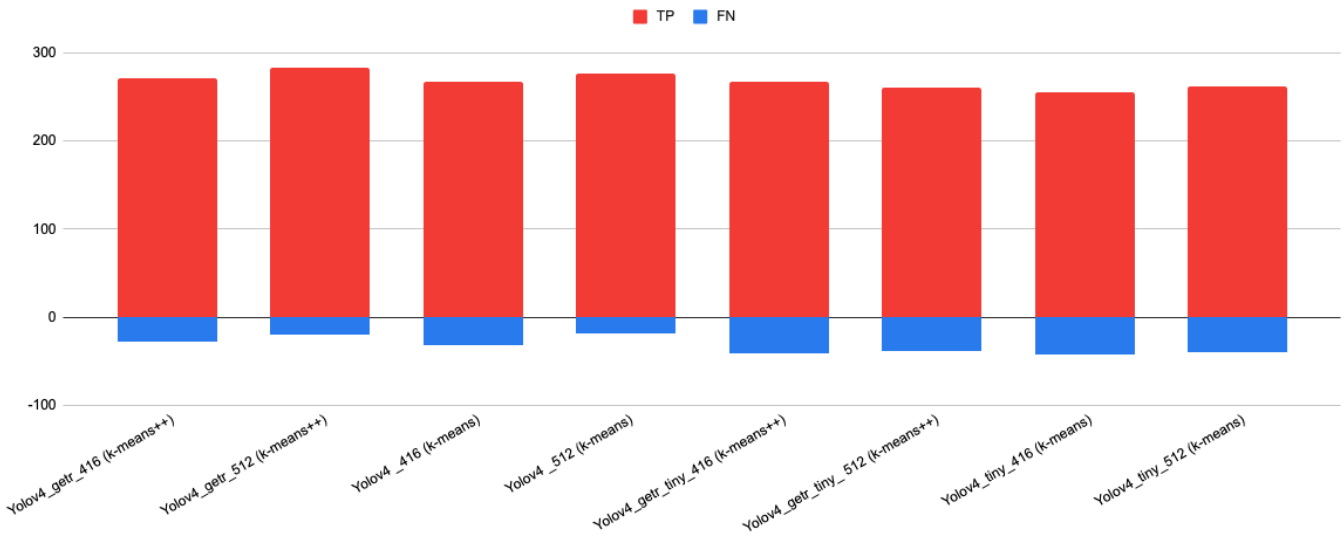


Рис. 4.9. Порівняння значень **FN** (сині) та **TP** (червоні) для тестованих моделей ЗНМ. Значення метрик зберігаються у Таблиці 4.1.

Проте розбіжність у кількості **FN** є прийнятною, оскільки для таких малих моделей точність розпізнавання все ще лишається високою, в діапазоні 85 **mAP.**

Значення роздільної здатності вхідних зображень, напряму впливає на результати ефективності розпізнавання моделі ЗНМ. При роздільній здатності в 512 пікселів, збільшується можлива площа обробки вхідних зображень, таким чином, усі характеристики ефективності моделі збільшуються в діапазоні 8-10%. Проте чим більший розмір вхідних зображень, тим більше апаратних ресурсів потрібно для виконання задачі розпізнавання. Тому на МП доречно обрати модель з оптимальною роздільною здатністю вхідних зображень, що становить 416x416 пікселів.

4.2.2. Продуктивність розроблюваної моделі ЗНМ на МП у реальному часі.

Основні метрики при виконанні дослідження: **FPS** (кількість кадрів за секунду)**, BFLOPs** (кількість мільярдів операцій з рухомою комою на секунду), час відображення результатів на екрані після початку опрацювання $\boldsymbol{t_{frame}}$, медіана часу для виконання операції передбачення об'єкта $\boldsymbol{t_{predict}}$, медіана часу для завантаження моделі у тестований пристрій $\boldsymbol{t_{load}}$, розмір вагових коефіцієнтів **w**, ступінь квантизації моделі ЗНМ **q**, апаратні засоби, типи процесорів, програмні засоби. Також для оцінки ефективності роботи ЗНМ на різних МП необхідно використати оцінку **mAP** (середня вибіркова влучність) та **ФРК** (фільтр розміру

кадру).

Для оцінки продуктивності розроблюваних моделей ЗНМ на МП у відеозображеннях у реальному часі доречно порівняти наявні програмні та апаратні засоби відповідно до визначених параметрів та метрик оцінки.

Для тестування, були обрані:

- Моделі ЗНМ з кількістю вихідних шарів: 2 (tiny model) та 3 (звичайна модель). Частина цих моделей зконвертована у CoreML формат для МОС iOS;
- Роздільна здатність вхідних зображень рівна 416x416 пікселів, тобто є сталою для всіх моделей ЗНМ та пристроїв на яких вона тестується;
- Апаратні засоби: для МОС iOS використовується електронна схема CPU, чипи акселерації NNA та GPU (в залежності від обраної точності обчислень з рухомою комою). Для вбудованої системи Jetson Nano використано CPU та чип акселерації GPU.

  Загальне Значення метрик $\boldsymbol{t_{predict}}$, $\boldsymbol{t_{load}}$ визначено з урахуванням використання усіх можливих засобів акселерації (CPU/GPU/NNA);
- Програмні засоби: фреймворк CoreML для МОС iOS та бібліотека OpenCV для вбудованої системи Jetson Nano на ОС Ubuntu;
- Ступінь квантизації вагових коефіцієнтів **q** моделей ЗНМ на МОС iOS:

  16 біт. Оптимальне значення для задач РО з використанням CPU та NNA;

  8 біт методом Афінних перетворень;

  4 біти методом створення таблиці пошуку (Lookup Table) за допомогою метода кластеризації к-середніх;

  32 біти. Подвійна точність обчислень, для збільшення продуктивності моделі при використанні більших апаратних ресурсів. Використовуються при цьому CPU та GPU.

  Для вбудованих систем застосовується стандартна точність у **16 біт**;
- **ФРК**, який був визначений як 1170 x 2532 пікселів для екрану тестованого мобільного пристрою Iphone 12, та 1024 x 760 пікселів для тестування відеопотоку на відео екрані МОС Ipad 2, який надходить з вбудованого пристрою Jetson Nano під ОС Ubuntu;

Результати тестування продуктивності розроблених ЗНМ наведені у Таблиці 4.2 та Таблиці 4.3.

Таблиця 4.2. Метрики продуктивності роботи двошарових ЗНМ в залежності від ступеню квантизації та типу моделі.

	Метрика / **Модель ЗНМ**	**FPS** **(кадрів/с)**	**BFLOP's** **(міл. оп.)**	**w** **(МБ)**	t_{frame} (с)	$t_{predict}$ (с)	t_{load} (с)	**mAP** **(%)**
q = 32	**Yolov4_getr_tiny_coreml**	8.4	6.454	**23.2**	0.02	**0.111**	**0.358**	86.9
	Yolov4 _tiny_coreml	**9**	**8.76**	24.5	0.02	0.134	0.42	87.1
q = 16	**Yolov4_getr_tiny_coreml**	**30.2**	7.34	**12**	0.03	**0.042**	**0.183**	82.1
	Yolov4_getr_tiny_nano	19.1	7.84	24	0.02	0.123	0.67	86.92
	Yolov4 _tiny_coreml	30.1	8.21	12.3	0.03	0.43	0.212	86.2
	Yolov4 _tiny_nano	18.0	**8.44**	24	0.02	0.234	0.69	87.21
q = 8	**Yolov4_getr_tiny_coreml**	**33.3**	4.22	**6.1**	0.02	**0.067**	0.434	82.1
	Yolov4 _tiny_coreml	33	**4.94**	7.9	0.02	0.074	**0.383**	81.7
q = 4	**Yolov4_getr_tiny_coreml**	32	**2.44**	**3.2**	0.03	0.08	0.46	68.1
	Yolov4 _tiny_coreml	32	2.56	3.3	0.04	0.08	0.41	61.2

Таблиця 4.3. Метрики продуктивності роботи тришарових ЗНМ в залежності від ступеню квантизації та типу моделі.

	Метрика / **Модель ЗНМ**	**FPS** **(кадрів/с)**	**BFLOP's** **(міл. оп.)**	**w** **(МБ)**	t_{frame} (с)	$t_{predict}$ (с)	t_{load} (с)	**mAP** **(%)**
q = 32	Yolov4_getr_coreml	5.1	49.2	257	4.6	0.427	2.6	98.1
	Yolov4 _coreml	5.2	**52.8**	258.5	4.2	0.428	2.5	97.9
q = 16	**Yolov4_getr_coreml**	**6.3**	**45.1**	**129**	3.7	**0.32**	3.55	97.3
	Yolov4_getr_nano	3.2	42.1	256	**3.2**	0.39	**2.44**	97.2
	Yolov4 _coreml	**6.4**	**46.2**	**129.7**	3.6	**0.34**	3.63	94.8
	Yolov4 _nano	3.3	42.3	257	**3.3**	0.4	**2.37**	94.2
q = 8	**Yolov4_getr_coreml**	8.1	**29.1**	64.2	2.1	**0.101**	**3.21**	82.1
	Yolov4 _coreml	8.0	28.2	65.2	2.32	0.14	3.39	85.5
q = 4	**Yolov4_getr_coreml**	**13.7**	18.3	33.3	1.35	0.081	**2.12**	73.1
	Yolov4 _coreml	13.4	18.1	34.1	1.43	0.083	2.43	54.2

Результати тестування варто аналізувати по значеннях метрик. Як видно з таблиць 4.2 та 4.3, чим більше значення **q**, тим збільшується значення **FPS**. При цьому пропорційно зменшується значення ефективності розпізнавання об'єктів по

метриками **mAP,** $t_{predict}$. При чому при квантизації на рівні 4 бітів, якість розпізнавання різко падає.

Значення **BFLOPS, w** та t_{load} зменшуються лінійно в залежності від збільшення ступеня квантизації та зміни кількості вихідних шарів ЗНМ. У більшості тестів пропонована модель ЗНМ GETR показує поліпшені результати у порівняні з прямими аналогами, та становить в середньому 5-10% по більшості метрик.

При порівнянні продуктивності роботи моделі ЗНМ на МОС iOS, використовуючи апаратний засіб Iphone 12 та вбудований пристрій Jetson Nano (при рівні квантизації 16 біт) спостерігається значна перевага МОС iOS. Перевага досягається за допомогою вдалого поєднання роботи процесорів системи (NNA та GPU) при вирішенні задач РО, у той час як приблизно аналогічний по характеристиках процесор Jeston ARM Nvidia не може забезпечити достатню кількість **BFLOPS**.

Для перевірки цієї гіпотези про використання апаратних засобів, був проведений тест роботи моделей ЗНМ на МОС iOS, з використанням обмежених апаратних ресурсів для метрик $t_{predict}$ та t_{load} (Рис. 4.9 – 4.12).

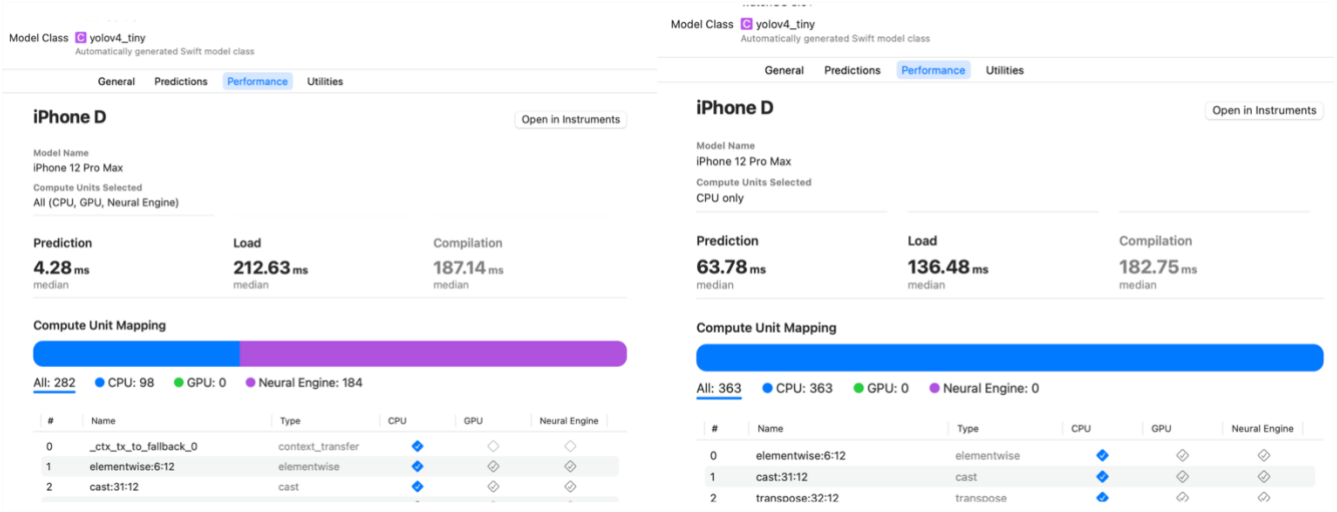


Рис. 4.10. Використання CPU та NNA (зліва) та лише CPU (справа) при тестування моделі ЗНМ yolov4_tiny. Де Prediction – $t_{predict}$, Load – t_{load}.

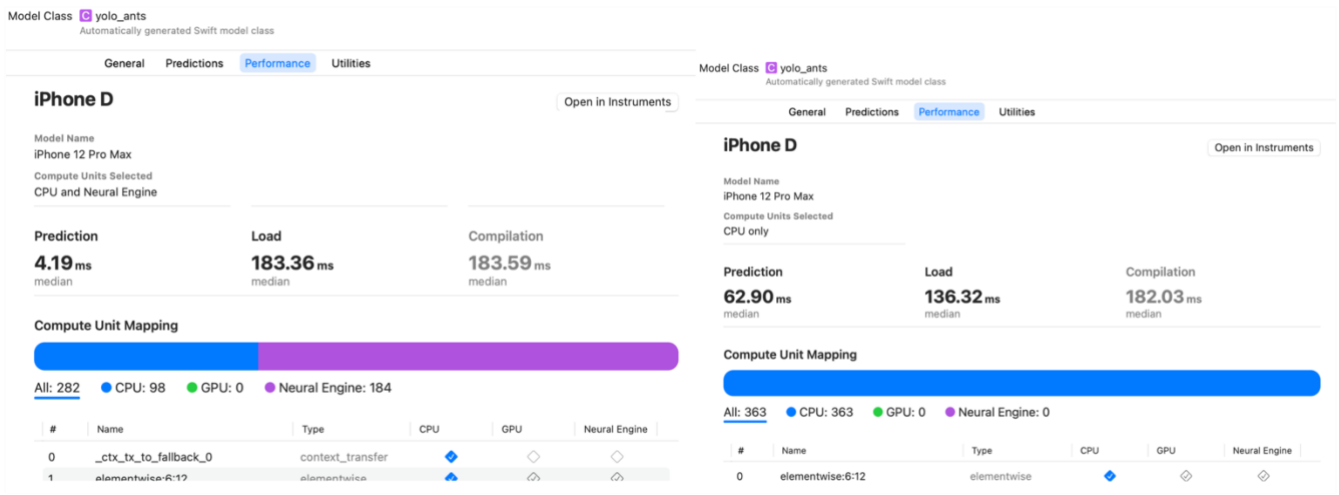


Рис. 4.11. Використання CPU та NNA (зліва) та лише CPU (справа) при тестування моделі ЗНМ yolov4_tiny_getr Де Prediction – $t_{predict}$, Load – t_{load}.

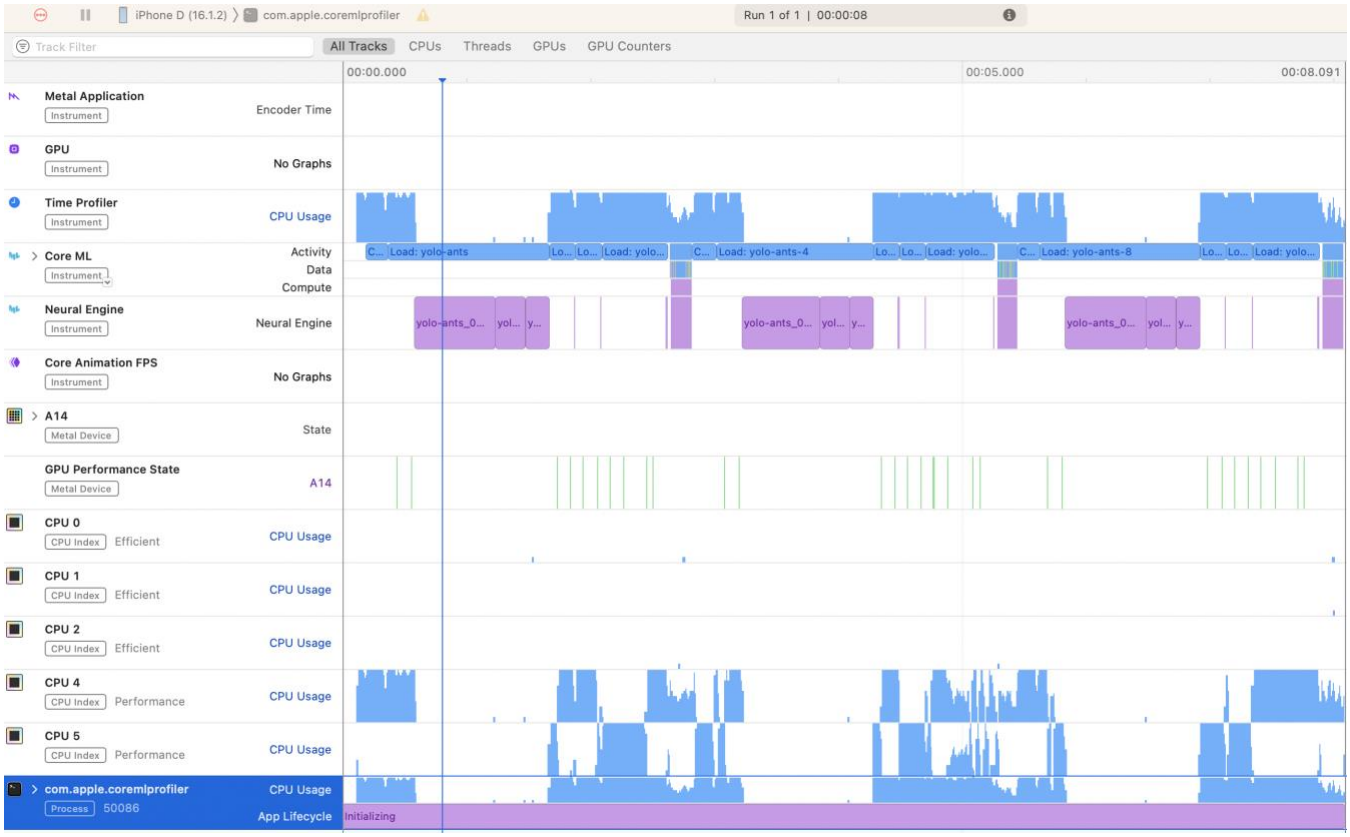


Рис. 4.12. Візуальна репрезентація розпаралелення процесі між CPU та NNA при тестуванні моделі ЗНМ yolov4_tiny_getr.

За результатами перевірки продемонстровано, що при використанні лише CPU при виконанні задачі РО на МП продуктивність передбачення (розпізнавання) та оброки інформації зменшується у 12 разів. При цьому час завантаження моделі збільшуються на величину 64%, що не впливає на ефективність розробки.

У той же час, при використанні квантизації у **32** біти виконується застосування чипа акселерації GPU замість NNA, що зменшує продуктивність моделі на 70%.

Таким чином, продуктивність моделі ЗНМ на МОС iOS при використанні CPU та GPU приблизно рівні метрикам вбудованої ОС Jetson Nano (при ступені квантизації у **16 біт** значення $t_{predict}$ рівне 0.111 для МОС iOS та 0.123 для вбудованої системи Jetson Nano).

FPS для більшості tiny моделей є вищим за граничне значення у 24 кадри на секунду, тому розроблені моделі підходять для вирішення поставлених задач у реальному часі.

4.2.3. Ефективність алгоритмів відстеження довільного класу рухомих об'єктів у реальному часі.

Основні метрики при виконанні дослідження: тип та методика алгоритму, **F1** (F-міра). Були використані попередньо визначені метрики з еталонної вибірки **MOT17**: **MOTA** (Багатоцільова точність), **MOTP (**Багатоцільова влучність**), MT** (кількість відстежуваних траєкторій), **ML** (кількість втрачених траєкторій), **ID** (кількість перемикань ідентифікатора), **FM** (кількість фрагментацій шляху відстеження).

Додатково були застосовані методи **FPL** (граничний ліміт кадрів) **BAT** (граничний кут нахилу), **PFTP** (траєкторія передбачених кадрів) для виконання задач відстеження довільної кількості об'єктів.

Значення мінімізаційного фільтру **IOU** визначено як 0.2, тобто мінімально необхідне для відстеження об'єктів, регіони розпізнавання яких перекриваються.

Для оцінки ефективності розроблених методів та засобів відстеження, були обрані імплементовані три алгоритми пошуку та відстеження об'єктів та 1 сторонній алгоритм DeepSort для порівняння характеристик метрик.

У Таблиці 4.4 визначено основні типи пропонованих до аналізу алгоритмів та метрики, на основі еталонної вибірки **MOT17**.

Таблиця 4.4. Визначення ефективності алгоритмів відстеження використовуючи метрики еталонної вибірки MOT17. Стрілочка вгору вказує, що найліпші результати є найбільшими, стрілочка униз вказує що найліпші результати є найменшими.

№	Метрика / Алгоритм	Тип	MOTP ↑	MOTA ↑	F1 (%) ↑	MT(%) ↑	ML (%) ↓	ID ↓	FM ↓
A1	**Алгоритмічний пошук та наведення**	Відстеження шляхом пошуку	44.2	34.2	68.23	10.34	54.2	2852	3256
A2	**Автоматизований пошук та наведення**	Навчання з підкріпленням	74.4	32.8	72.1	14.1	32.1	**610**	**1318**
A3	**Алгоритмічний пошук при перетині регіонів**	Відстеження шляхом пошуку	**82.9**	44.3	**82.4**	**36.2**	27.2	2374	3836
A4	**DeepSort [101]**	Сіамські ЗНМ	61.4	**79.1**	73.82	32.8	**18.2**	781	2008

У досліджені порівнюються 3 типи алгоритмів: відстеження шляхом пошуку, навчання з підкріпленням та Сіамські ЗНМ.

Згідно результатів, найвищий показник **F1,** оцінки влучності позиціювання розпізнаних об’єктів (**MOTP**) та **MT** має алгоритм **A3**. При чому як алгоритм збіжності для цього метода був обраний Угорський алгоритм, що значно збільшив показники ефективності у порівнянні з існуючим рішенням з алгоритмом k-d tree (наприклад значно зменшилась кількість перемикань **ID** та збільшився показник **MT,** що продемонстровано у попередніх дослідженнях). Загалом метрики **MOTP** та **MT** для алгоритму **A3** показали на 12% ліпші результати у порівняні з другим по ефективності алгоритмом **A4.** Ліпші результати на 10% показала метрика **F1.** Недоліком алгоритму **A3** є все ще високий рівень перемикань **ID** та втрат траєкторії **FM** у порівнянні з прямими аналогами.

На Рис. 4.13 зображено візуальне порівняння основних характеристик, отриманих у дослідженні.

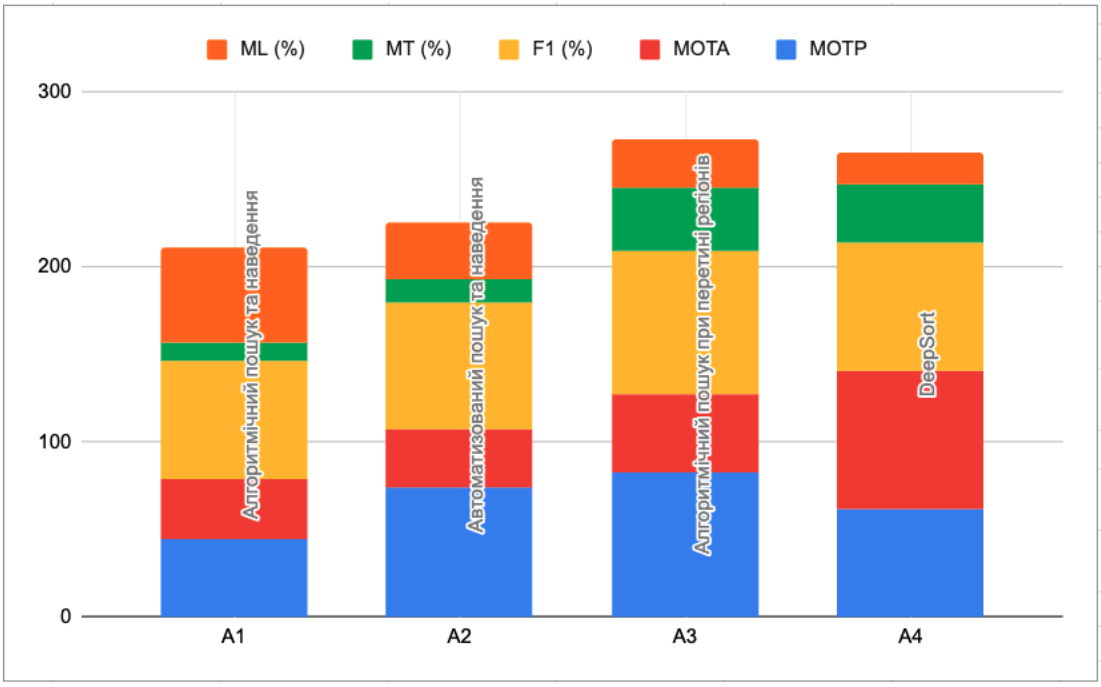


Рис. 4.13. Метрики ML, MT, F1, MOTA, MOTP ефективності відстеження об’єктів для алгоритмів A1-A4.

Алгоритм DeepSort (**A4**) показав ефективність при генерації метрики **MOTA** та **ML**. При чому значення перемикань **ID** було незначним.

Алгоритм **A2** з використанням навчання з підкріпленням показав середні результати по більшості метрикам, проте мав мінімальні значення перемикань **ID** та кількості втрачених траєкторій за принаймі 20% тривалості часу відстеження (**ML**). З цього можна зробити висновок, що алгоритм **A2** цілком підходить для виконання задач у реальному часі, при великій кількості вхідних об’єктів (Рис. 4.14).

При цьому значення **ID** та **FM** дуже залежать від вхідного значення мінімізаційного фільтру **IOU**, тобто наскільки максимально часто можуть перетинатися ймовірні об’єкти. Чим це значення більше, тим менше буде значення **ID** та **FM**, проте й менша кількість об’єктів пройде фільтрацію.

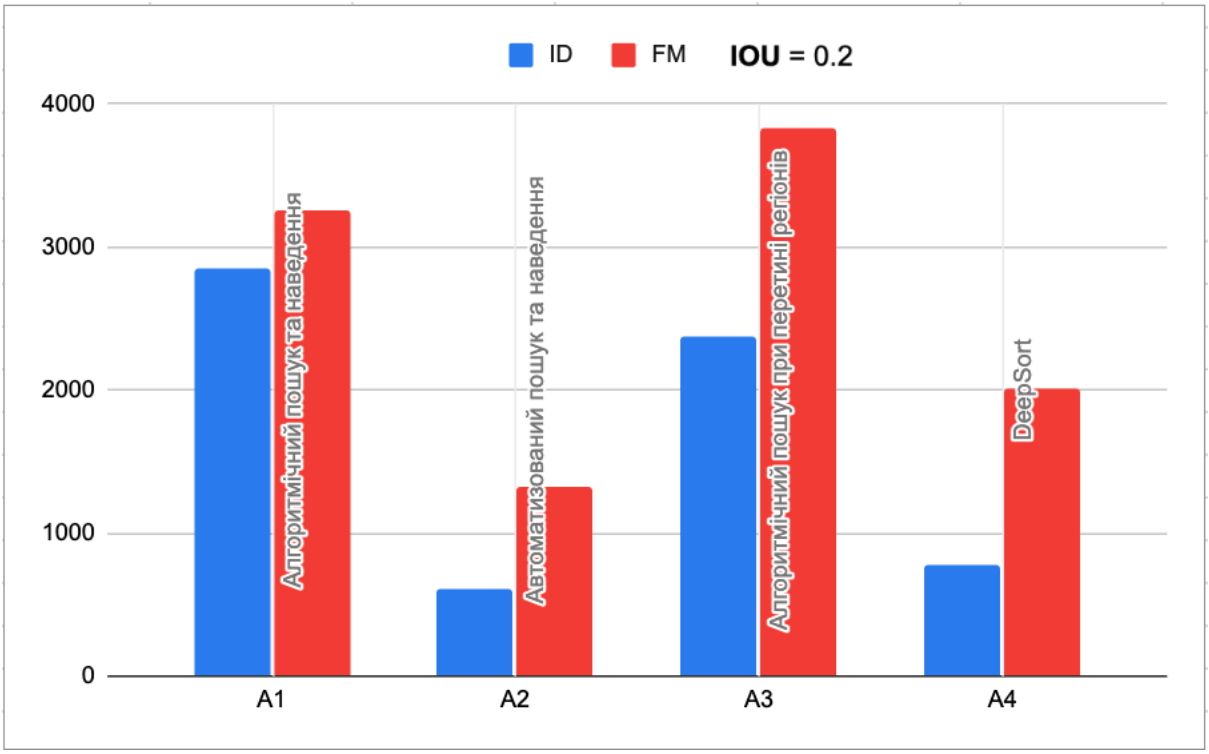


Рис. 4.14. Метрики ID, FM ефективності відстеження об’єктів для алгоритмів A1-A4. Чим менше перемикань ідентифікаторів об’єктів ID та відхилень траєкторії FM, тим ефективніший алгоритм відстеження при обробці багатьох рухомих об’єктів.

Найгірші результати показав алгоритм **A1,** через жадібний (greedy) підхід до перебору об’єктів при знаходженні правильної траєкторії. При цьому значення **F1** для цього підходу ще є у прийнятних межах, та становить **68.23%**.

Таким чином, для апробації результатів досліджень був обраний алгоритм A3, оскільки значення **F1**, **MOTP** та **MT** серед досліджуваних аналогів, що дозволить розпізнавати та відстежувати рухомі об’єкти.

Апробація результатів.

Був виконаний набір тестів для перевірки розроблених методів та засобів на МП. До задач тестів віднесено: пошук, розпізнавання та відстеження класу об’єктів “мурахи”; присвоєння унікального ідентифікатора кожному розпізнаному об’єкту; відтворення шляху просування об’єктів класу мурахи; рахування кількості об’єктів, які перетнули задані граничні лінії. Результати дослідження наведені на Рис. 4.15 – 4.19.

Як метрики досліджень використано: пропоновані алгоритми збіжності; методи **FPL** (граничний ліміт кадрів) **BAT** (граничний кут нахилу), **PFTP** (траєкторія передбачених кадрів).

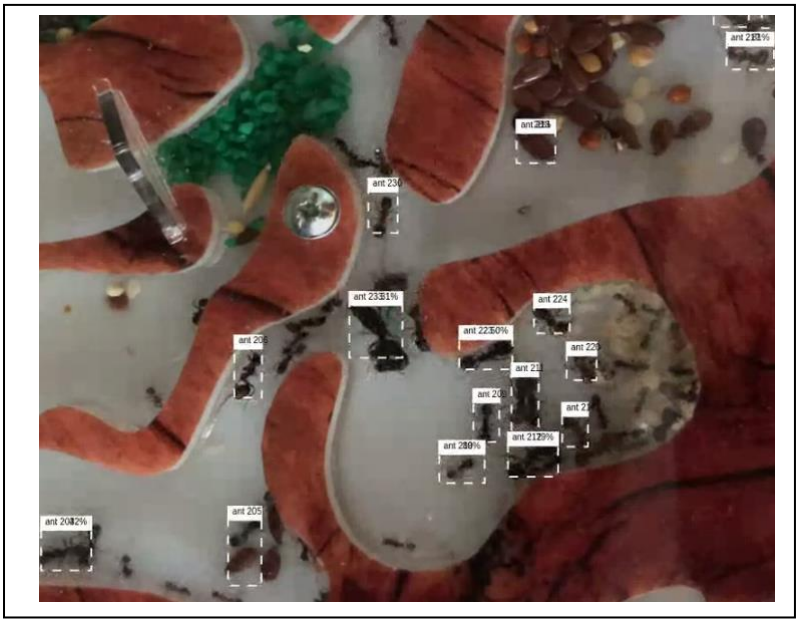


Рис. 4.15. Результати розпізнавання та відстеження класу об’єктів “мурахи”, з присвоєнням унікальних ідентифікаторів.

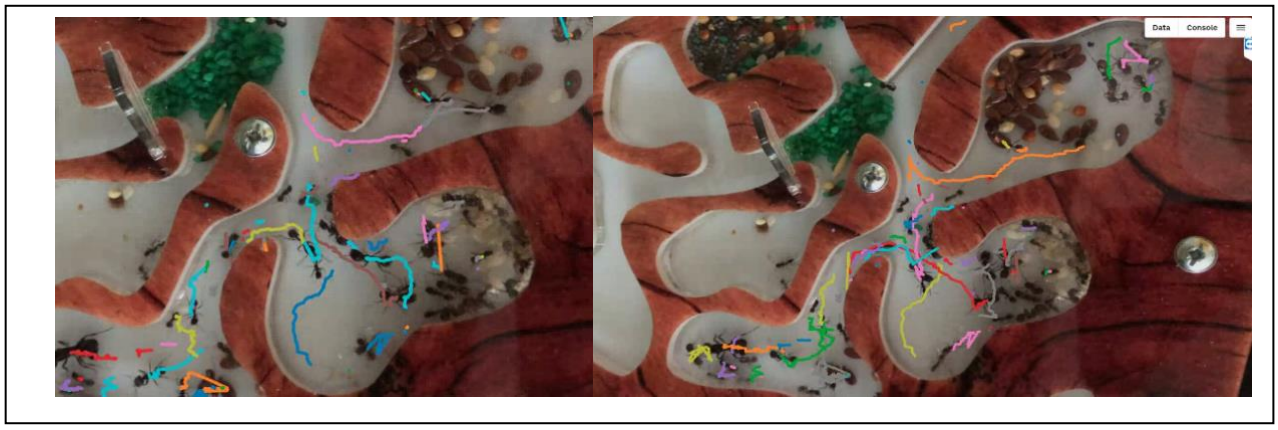


Рис. 4.16. Розпізнавання, відстеження, та збереження шляху руху кожної мурах. Зліва – з використанням Угорського алгоритму збіжності. Справа – з використанням алгоритму збіжності Kd-tree.

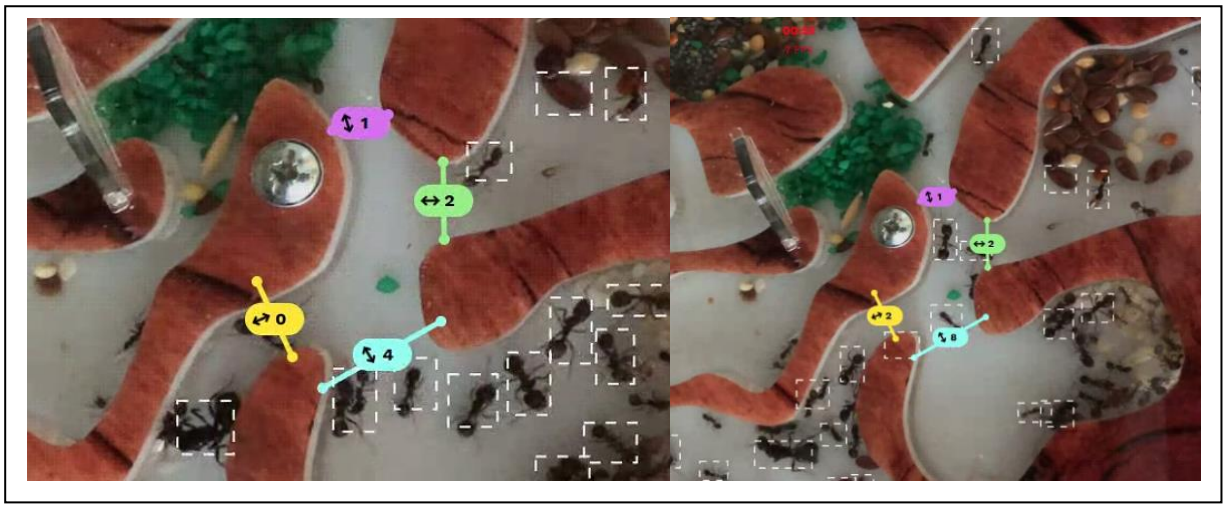


Рис. 4.17. Розпізнавання та рахування руху класу об’єктів у різні регіони формікарія. Зліва – з використанням Угорського алгоритму збіжності. Справа – з використанням алгоритму збіжності Kd-tree. Параметр BAT був визначений як **45** для обидвох випадків.

Рис. 4.18. Розпізнавання та рахування руху класу об’єктів у різні регіони формікарія. Зліва – з використанням Угорського алгоритму збіжності. Справа – з використанням алгоритму збіжності Kd-tree. Параметр **BAT** був визначений як **0** для обидвох випадків.

Рис. 4.19. Результати тестів, збережені у сховище для кожної заданої граничної лінії.

Наступні вхідні параметри були визначені в ході тестів:

- *Модель ЗНМ*: модель розпізнавання об’єктів Yolov4_getr_416_tiny (k-means ++). Ступінь квантизації **q** - 16 біт;
- *Час виконання тестів*: 1 хвилина. Для кожного тесту був використаний однаковий відео фрагмент;
- *Граничне значення **FPS**:* методом **FPL** був визначений граничний ліміт кадрів у 30, для збереження великої кількості істинно позитивних значень (**TP**) при відстеженні об’єктів;
- *Розмір вхідних об’єктів класу мурах:* 8-12 мм;
- *Мінімізаційний фільтр **IOU**: 0.2;*
- ***ФРК***: який був визначений як 1170 x 2532 пікселів для екрану тестованого мобільного пристрою Iphone 12, та 1024 x 760 пікселів для тестування відеопотоку на відео екрані МОС Ipad 2, який надходить з вбудованого пристрою Jetson Nano під ОС Ubuntu;
- ***ЗФС***: 90.

Результати проведених тестів висвітлені у Таблиці 4.5.

Таблиця 4.5. Ітерації тестів запуску методів та засобів для розпізнавання та відстеження рухомих об’єктів в залежності від алгоритму збіжності.

Метрика **№ ітерації тесту**	**Алгоритм збіжності**	**BAT (°)**	**PFTP (°)**	**Об’єкти, які Перетнули граничну лінію**	**Розпізнані об’єкти**
Запуск–1	Hungarian	90	90	22	32
Запуск–2	k-d tree	90	90	11	24
Запуск–3	k-d tree	45	45	0	27
Запуск–4	Hungarian	45	45	4	23
Запуск–5	k-d tree	20	60	5	28
Запуск–6	Hungarian	20	60	9	29
Запуск–7	k-d tree	0	0	0	21
Запуск–8	Hungarian	0	0	7	24

Тести були запущені для записаного відеопотоку для симуляції середовища з реальним часом, на якому відображено рух мурах по формікарію. Для даної задачі була натренована ЗНМ типу yolov4 з автоматично завантажених анотованих зображень класу «мурахи». Для автоматизованого створення якорів розпізнавання, був використаний метод k-means++.

Кількість розпізнаних об’єктів це сумарна кількість унікальних об’єктів, які були розпізнані, та до яких був присвоєний унікальний ідентифікатор. Деякі ідентифікатори ID були перевизначені протягом тесту, при переміщенні мурахи у іншій площині формікарію з коричневими стінками. Результати також показують більшу втрату траєкторії при використанні алгоритму збіжності k-means (це видно на Рис. 4.14). Також тести показують що кількість об’єктів класу «мурахи», які перетнули граничну лінію в цілому при використанні k-means є меншою, у порівнянні з Угорським (Hungarian) алгоритмом. Як видно з Таблиці 4.5, точність відстеження при використанні граничних фільтрів та Угорського алгоритму збільшилась на 50%. При чому якщо BAT та PFTP є відключеними, то точність розпізнавання та відстеження зростає на 40%, при високому FPS. Згідно з отриманими даними, значення перемикань ID зменшується лінійно в залежності від значень BAT та PFTP, та знаходиться у межах 5-6%. У той же час негативні результати можуть бути скоректовані фільтром FPL.

Результати розпізнавання мурах є дещо гіршими при знаходженні мурахи на вертикальній стіні чи на коричневій поверхні, проте це очікувано, оскільки набір даних був сформований на основі мурах, які живуть в лабораторних умовах. При використанні змішаного набору даних (мурахи в лабораторних та природних умовах), результати розпізнавання мали б бути вищими. Клас «мурах» є прикладом можливостей роботи системи на мобільній платформі. Клас об’єктів може бути розширений в залежності від задач системи.

Для МОС iOS були розроблені суміжні методи та засоби для розпізнавання та відстеження рухомих об’єктів. Де шлях визначається відображенням центроїдів кожного унікального об’єкта зі специфічним кольором (Рис. 4.20).

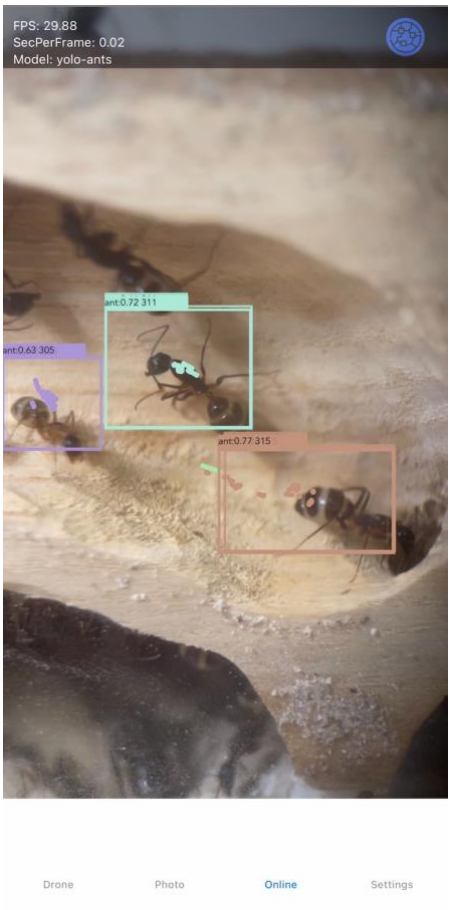


Рис. 4.20. Результати апробації розроблених методів та засобів на МОС iOS.

В цілому, розроблені методи та засобу пошуку, розпізнавання та відстеження об’єктів на мобільній платформі показали хороші результати при виконанні тестів, там можуть бути застосовані у обробці відеозображень в реальному часі.

Висновки до четвертого розділу.

У розділі визначено засоби імплементації елементів системи. Виконано аналіз та апробацію результатів роботи такої системи.

Обґрунтовано вибір компонентів системи для виконання задач РО на МП. Обрано PAAS платформу на базі системи масштабування Docker для виконання завдань анотування, тренування та конвертування моделі ЗНМ, як платформу яка є не залежною від інтернет-з'єднання, що є критично в умовах відсутності доступу до мережі інтернет.

Для розв'язання задач дисертаційного дослідження на МОС iOS, був використаний фреймворк CoreML. Розроблено схему функціональних компонентів та потоків даних такої системи, а також діаграму класів системи. Проведено прототипування інтерфейсу користувача системи.

Для вбудованих систем на МП була використана система акселерації GPU CUDA на чипі Jetson Nano з процесором ARM Nvidia Carmel на базі ОС Linux. Розроблено схему функціональних компонентів та потоків такої даних системи.

Для задачі відстеження рухомих об'єктів, для підтримування принципів SOLID, вирішено інтегрувати такі методи та засоби як окремий налаштовуваний Javascript модуль, який працює як для мобільних систем на платформі iOS, так і на вбудованій МП на базі ОС Linux.

Проведений аналіз та апробація результатів дослідження. Висвітлено, що основними проблемами, які постають перед такими системами є: визначення ефективності моделі ЗНМ для задач розпізнавання об'єктів, визначення продуктивності розробленої моделі ЗНМ на мобільній платформі враховуючи наявні апаратні та програмні засоби та визначення ефективності розроблених методів відстеження об'єктів.

Результати аналізу показали, що за допомогою метода кластеризації k-means++ вдалось досягти ліпших результатів для mAP на 5%, F1 на 5-6%, R на 3-4% при оцінці ефективності розпізнавання об'єктів. При цьому визначено, що для оптимальної роботи моделі ЗНМ на МП, враховуючи наявні апаратні можливості,

доречно використовувати модель з 2 вихідними шарами, та роздільну здатність ЗНМ рівну 416х416 пікселів.

Аналіз метрик продуктивності роботи розробленої ЗНМ показав що оптимальне значення ступеня квантизації моделі ЗНМ є 16 біт для моделі з 2 вихідними шарами, та 8 біт для моделі з 3 вихідними шарами. При цьому FPS для більшості моделей є у межах 24 FPS що є достатнім для виконання задач у реальному часі.

Найбільшу продуктивність показали моделі на МОС iOS, зконвертовані у CoreML формат, оскільки така МОС вдало поєднує роботу процесорів системи (NNA та GPU) при вирішенні задач РО, у той час як приблизно аналогічний по характеристикам процесор Jeston ARM Nvidia не може забезпечити достатню кількість BFLOPS. Дослідження показали, що використання чипів NNA та CPU збільшують показники передбачення (розпізнавання) та оброки інформації $\boldsymbol{t_{predict}}$ у 12 разів у порівнянні з використанням CPU на МОС iOS.

У більшості тестів на продуктивність пропонована модель ЗНМ GETR показує поліпшені результати у порівняні з прямими аналогами, та становить в середньому 5-10% по більшості метрик.

Для аналізу метрик ефективності алгоритмів відстеження об’єктів, обрано 3 реалізованих алгоритми, та алгоритм пошуку типу RE-ID DeepSort.

За результатами тестів визначено, що пропонований алгоритм пошуку при перетині регіонів (А3), показав найвищі результати по метриках F1, MOTP та MT, з використанням Угорського алгоритму як алгоритму збіжності. Загалом метрики MOTP та MT для алгоритму А3 показали на 12% ліпші результати у порівняні з другим по ефективності алгоритмом DeepSort**.** Ліпші результати на 10% показала метрика F1**.** Недоліком алгоритму А3 є все ще високий рівень перемикань ID та втрат траєкторії FM у порівнянні з прямими аналогами.

Проведено апробацію результатів дослідження на МП у реальному середовищі на базі класу об’єктів «мурахи».

При апробації був порівняний алгоритм збіжності kd-tree з пропонованим Угорським алгоритмом для виконання задачі відстеження рухомих об’єктів.

За результатами дослідження, виявлено що точність відстеження при використанні граничних фільтрів та Угорського алгоритму збільшилась на 50%. При чому якщо BAT та PFTP є відключеними, то точність розпізнавання та відстеження зростає на 40%, при високому FPS. Згідно отриманих даних, значення перемикань ID зменшується лінійно в залежності від значень BAT та PFTP, та знаходиться у межах 5-6%. У той же час негативні результати можуть бути скоректовані фільтром FPL.

Проаналізувавши отримані результати, можна дійти до висновку, що поставлені задачі дослідження та імплементації методів та засобів для пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі у реальному часі успішно виконано.

ВИСНОВКИ

У дисертаційній роботі проведено теоретичне обґрунтування та розв’язання актуальної наукової задачі розробки та дослідження методів та засобів пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі у реальному часі. При цьому отримано наступні наукові та практичні результати:

1. Проведено аналіз сучасних методів та засобів розпізнавання та відстеження об’єктів на МП. Визначено, що для виконання задачі розпізнавання об’єктів на мобільній платформі, доречно використати сімейство моделей ЗНМ Yolo. Як платформу масштабування, обрано PAAS платформу Docker. Як метод запуску моделей ЗНМ на МП iOS, був обраний фреймворк CoreML. З урахуванням проаналізованих матеріалів посталено вимоги до створення мобільної платформи для пошуку та розпізнавання об’єктів у реальному часі.
2. Впроваджено поліпшений метод кластеризації k-means++ при генерації якорів розпізнавання на етапі тренування моделі ЗНМ. Результати аналізу показали, що за допомогою метода кластеризації k-means++ вдалось досягти ліпших результатів для mAP на 5%, F1 на 5-6%, R на 3-4% при оцінці ефективності розпізнавання об’єктів, у порівнянні з існуючим методом кластеризації k-means.
3. Впроваджено метод квантизації вагових коефіцієнтів моделі методом афінних перетворень для МП. Аналіз метрик продуктивності роботи розробленої ЗНМ показав що оптимальне значення ступеню квантизації моделі ЗНМ є 16 біт для моделі з 2 вихідними шарами, та 8 біт для моделі з 3 вихідними шарами. При цьому FPS для більшості моделей є у межах 24 FPS що є достатнім для виконання задач у реальному часі. У більшості тестів на продуктивність пропонована модель ЗНМ GETR показує поліпшені результати у порівняні з прямими аналогами, та становить в середньому 5-10% по більшості метрик.
4. Розроблено методи фільтрації результатів розпізнавання: фільтри ЗФС,

ФРК та задання порогових значень IOU та Object Confidence в залежності від потреб поточної задачі. Такі фільтри дозволяють точніше відображати результати розпізнавання в багатокласовій системі, зменшуючи надлишковість регіонів розпізнавання.

5. Запропоновано методи мемоїзації результатів відстеження об’єктів PFTP, BAT та FPL , які полягають у формуванні порогових значень та граничних ліній для збереження та фільтрування результатів відстеження у разі втрати ID об’єкта. Також такі методи дозволяють зменшити частоту перемикань ідентифікаторів ID_n.
6. Розроблено засоби масштабованого розгортання систем автономного анотування вхідних зображень, тренування моделі ЗНМ та конвертування отриманої моделі у CoreML формат для МП на базі PAAS платформи Docker;
7. Розроблено 3 методи відстеження об’єктів у реальному часі. Перший метод (A1) є алгоритмічним представленням відстеження об’єктів в залежності від їх відхилення від центру інформаційного кадру з можливістю наведення на об’єкт за допомогою ВКВ КФС. Другий метод (A2) полягає у модифікації алгоритму відстеження, шляхом додавання агенту динамічного середовища DDPG, який був навчений методом навчання з підкріпленням. Для зменшення апаратної залежності, був розроблений третій модифікований алгоритм (A3) оперативного відстеження на базі IOU, який полягає у використанні Угорського алгоритму збіжності для порівняння об’єктів відстеження у реальному часі. Ефективність роботи алгоритму була підтверджена у ході тестів на базі еталонної вибірки MOT17. За результатами тестів визначено, що пропонований алгоритм пошуку при перетині регіонів (A3), показав найвищі результати по метриках F1, MOTP та MT, з використанням Угорського алгоритму як алгоритму збіжності. Загалом метрики MOTP та MT для алгоритму A3 показали на 12% ліпші результати у порівняні з другим по ефективності алгоритмом DeepSort. Ліпші результати на 10% показала метрика F1.

Недоліком алгоритму А3 є все ще високий рівень перемикань ID та втрат траєкторії FM у порівнянні з прямими аналогами.

8. Запропоновано засіб інтеграції ВМ для відстеження рухомих об’єктів на МП iOS. Інтеграція полягає у використанні бібліотеки JavaScriptCore для передачі даних між системою та модулем. Також інтегровано багатопоточну систему обміну інформацією для зменшення навантаження на ВМ.
9. Розроблена система була апробована на реальному середовищі, для виконання задачі розпізнавання, відстеження та рахування рухомих об’єктів класу «мурахи». При апробації був порівняний алгоритм збіжності kd-tree з пропонованим Угорським алгоритмом для виконання задачі відстеження рухомих об’єктів за допомогою алгоритму А3. За результатами дослідження, виявлено що точність відстеження при використанні граничних фільтрів та Угорського алгоритму збільшилась на 50%. При чому якщо BAT та PFTP є відключеними, то точність розпізнавання та відстеження зростає на 40%, при високому FPS. Згідно отриманих даних, значення перемикань ID зменшується лінійно в залежності від значень BAT та PFTP, та знаходиться у межах 5-6%. У той же час негативні результати можуть бути скоректовані фільтром FPL.
10. Розроблений мобільний додаток на базі МОС iOS для реалізації усіх задач дослідження. Також для порівняння та аналізу розроблених методів та засобів, була розгорнута ВС Jetson Nano на базі ОС Ubuntu на якій були реалізовані поставлені задачі пошуку, розпізнавання та відстеження. Практичне застосування розроблених методів та засобів полягає у наступному: інтеграції системи аналізу рухомих об’єктів за допомогою дрону Tello на МОС iOS; інтеграції системи аналізу малих об’єктів на ВС Jetson Nano. Також частина результатів дисертаційної роботи використана в навчальному процесі Національного університету «Львівська політехніка» при викладанні дисципліни «Цифрова обробка сигналів» на кафедрі електронних обчислювальних машин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кушнір, Д., & Парамуд, Я. (2019). Методи пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі ios в реальному часі. Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 1(1), 24-34. https://doi.org/10.23939/csn2019.01.024.
2. Kushnir, D., & Paramud, Y. (2020). The algorithm of Cyber-Physical system targeting on a movable object using the smart sensor unit. *Advances In Cyber-Physical Systems*, 5(1), 16-22. https://doi.org/10.23939/acps2020.01.016.
3. Кушнір, Д., & Парамуд, Я. (2020). Алгоритм оперативного наведення засобів вимірюваьно – керувального вузла кіберфізичної системи на рухомий об’єкт. Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 2(1), 44-52. https://doi.org/10.23939/csn2020.01.044.
4. Кушнір, Д. (2021). Методи та засоби покращення точності розпізнавання об’єктів на мобільній платформі iOS в реальному часі. Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 3(1), 80-88. https://doi.org/10.23939/csn2021.01.080.
5. Kushnir, D. (2022) Methods and means for small dynamic objects recognition and tracking. *Computers, Materials & Continua,* 73(2), 3649–3655. https://doi.org/10.32604/cmc.2022.030016.
6. Kushnir, D., & Paramud, Y. (2020). Model for real-time object searching and recognizing on mobile Platform. 2020 IEEE 15Th International Conference On Advanced Trends In Radioelectronics, Telecommunications And Computer Engineering (TCSET). https://doi.org/10.1109/tcset49122.2020.235407.
7. Kushnir, D., Ocherklevich, O., & Paramud, Y. (2021). Deep Neural Network Model for Text Semantic Analysis Based on Word Embeddings. *2021 11Th International Conference On Advanced Computer Information Technologies (ACIT)*. https://doi.org/10.1109/acit52158.2021.9548393.
8. Vavruk, E., & Kushnir, D. (2018). Mobile system for text recognition and translation with using Microsoft Cognitive API. *8th International youth science forum «Litteris et Artibus», 81–84*. https://ena.lpnu.ua/handle/ntb/51711.

9. Ваврук, Є., & Кушнір, Д. (2018). Система розпізнавання та перекладу текстової інформації в мобільних додатках з використанням бібліотеки Microsoft Cognitive OCR. Вісник Національного університету “Львівська політехніка”. Серія: Комп’ютерні системи та мережі, 1(905), 33-42. https://doi.org/10.23939/csn2018.905.033.
10. Andrushchak, N., Vynnyk, D., Melnyk, M., Bajurko, P., Kushnir, D., Haiduchok, V., ... & Yashchyshyn, Y. Impact of optical Illumination on transmission of subterahertz electromagnetic waves by Bi 12 GeO 20 crystals. *Acta Physica Polonica A*, 4(141), 415-419. https://doi.org/10.12693/APhysPolA.141.415.
11. Borak, T., Kushnir, D., & Paramud, Y. (2022). Microprocessor Subsystem of the Smart House to Control the Multichannel Irrigation of the Room Plants. *Advances In Cyber-Physical Systems*, 7(1), 1-7. https://doi.org/10.23939/acps2022.01.001.
12. Warwick, K. (2013). Artificial intelligence: the basics, Routledge. https://doi.org/10.4324/9780203802878.
13. Hu, J., Niu, H., Carrasco, J., Lennox, B., & Arvin, F. (2020). Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Transactions on Vehicular Technology, 69(12), 14413-14423 doi: 10.1109/TVT.2020.3034800.
14. Yilmaz, A., Demircali, A. A., Kocaman, S., & Uvet, H. (2020). Comparison of Deep Learning and Traditional Machine Learning Techniques for Classification of Pap Smear Images. arXiv preprint arXiv:2009.06366.
15. Oliveira, R., Araújo, R. C., Barros, F. J., Segundo, A. P., Zampolo, R. F., Fonseca, W., ... & Brasil, F. S. (2017). A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges. Journal of Microwaves, Optoelectronics and Electromagnetic Applications, 16, 628-645.
16. Narayan, S. (1997). The generalized sigmoid activation function: Competitive supervised learning. Information sciences, 99(1-2), 69-82.
17. Banerjee, C., Mukherjee, T., & Pasiliao Jr, E. (2020, April). The multi-phase ReLU activation function. In Proceedings of the 2020 ACM Southeast

Conference. 239-242. https://doi.org/10.1145/3374135.3385313.

18. Misra, D. (2019). Mish: A self regularized non-monotonic activation function. arXiv preprint arXiv:1908.08681.
19. Joshi, V., Das, A., Sun, E., Mehta, R. R., Li, J., & Gong, Y. (2021, August). Multiple Softmax Architecture for Streaming Multilingual End-to-End ASR Systems. In Interspeech (pp. 1767-1771).
20. Sweta Shaw. (October, 2022). Activation Functions Compared With Experiments.https://wandb.ai/shweta/Activation%20Functions/reports/Activation-Functions-Compared-With-Experiments--VmlldzoxMDQwOTQ.
21. Elgendy, M. (2020). Deep learning for vision systems (1st ed.). Manning. https://books.google.com.ua/books?hl=uk&lr=&id=sDszEAAAQBAJ&oi=fnd&pg=PR13&dq=Elgendy,+M.+(2020).+Deep+learning+for+vision+systems+(1st+ed.).+Manning&ots=8CCANreUy5&sig=xjoeMM3SCna2es_sEG2fI90mmXQ&redir_esc=y#v=onepage&q&f=false.
22. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
23. Hoang, N. D. (2020). Image processing-based spall object detection using gabor filter, texture analysis, and adaptive moment estimation (Adam) optimized logistic regression models. Advances in Civil Engineering, 2020.
24. Sah, S. (2020). Machine learning: a review of learning types. https://doi.org/10.20944/preprints202007.0230.v1.
25. Nguyen, N. V., Rigaud, C., & Burie, J. C. (2019, February). Semi-supervised Object Detection with Unlabeled Data. In VISIGRAPP (5: VISAPP). 289-296. https://doi.org/10.5220/0007345602890296.
26. Chen, B., Chen, W., Yang, S., Xuan, Y., Song, J., Xie, D., ... & Zhuang, Y. (2022). Label Matching Semi-Supervised Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 14381-14390. https://doi.org/10.48550/arXiv.2206.06608.
27. Cebeci, Z., & Yildiz, F. (2015). Comparison of k-means and fuzzy c-means algorithms on different cluster structures. Journal of Agricultural Informatics,

6(3). https://doi.org/10.17700/jai.2015.6.3.196.

28. Bonaccorso, G. (2020). Mastering Machine Learning Algorithms: Expert techniques for implementing popular machine learning algorithms, fine-tuning your models, and understanding how they work. Packt Publishing Ltd.
29. Asgher, U., Khalil, K., Khan, M. J., Ahmad, R., Butt, S. I., Ayaz, Y., ... & Nazir, S. (2020). Enhanced accuracy for multiclass mental workload detection using long short-term memory for brain–computer interface. Frontiers in neuroscience, 14, 584. https://doi.org/10.3389/fnins.2020.00584.
30. Masooma Memon. (November, 2022). ANN vs CNN vs RNN: Neural Networks Guide. https://levity.ai/blog/neural-networks-cnn-ann-rnn.
31. Mayank Mishra. (August, 2020). Convolutional Neural Networks, Explained. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.
32. Arc. (December, 2018). An Introduction to Convolutional Neural Networks. https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05.
33. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. (2014). UC Berkeley, ICSI. Rich feature hierarchies for accurate object detection and semantic segmentation.580-587. http://openaccess.thecvf.com/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf.
34. Ross Girshick. (2015) Microsoft Research. Fast R-CNN.http://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.
35. Shaoqing Ren, Kaiming He, Ross Girshick, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.
36. Public Content. Histogram of Oriented Gradients (HOG) Descriptor https://software.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/computer-vision/feature-detection-

functions/histogram-of-oriented-gradients-hog-descriptor.html.

37. Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. Advances in neural information processing systems, 29. https://arxiv.org/pdf/1605.06409.pdf.
38. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector https://arxiv.org/pdf/1512.02325.pdf.
39. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916. https://arxiv.org/pdf/1406.4729.pdf)%C3%AC%20%CB%9C.
40. J. Redmon, S. Divvala, R. Girshick and A. Farhadi. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 779-788, https://doi.org/10.1109/CVPR.2016.91.
41. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," ArXiv, 2018, https://arxiv.org/ abs/1804.02767.
42. A. Bochkovskiy, J. Redmon, S. Sinigardi, T. Hager, J. JaledMC et al. (2021). AlexeyAB/darknet:YOLOv4 (version yolov4). Zenodo. https://doi.org/10.5281/zenodo.562267.
43. W. Kin-Yiu, “Implementation of Scaled-YOLOv4 using PyTorch framework (v1.0.0),” Zenodo, 2021, https://doi.org/10.5281/zenodo.5534091.
44. G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, T. Xie et al. (2021). Ultralytics/YOLOv5:v6.0–YOLOv5n ‘Nano’ models. Roboflow integration. TensorFlow export, OpenCV DNN support (v6.0). Zenodo. https://doi.org/10.5281/zenodo.5563715.
45. Wang, C. Y., Yeh, I. H., & Liao, H. Y. M. (2021). You only learn one representation: Unified network for multiple tasks. arXiv preprint arXiv:2105.04206.
46. Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., ... & Liu, W. (2021). You

only look at one sequence: Rethinking transformer in vision through object detection. Advances in Neural Information Processing Systems, 34, 26183-26197.

47. Li, M., Zhang, Z., Lei, L., Wang, X., & Guo, X. (2020). Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster R-CNN, YOLO v3 and SSD. Sensors, 20(17), 4938. https://doi.org/10.3390/s20174938.
48. Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., & Ouni, K. (2019, February). Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. In 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS) (pp. 1-6). IEEE. https://doi.org/10.48550/arXiv.1812.10968.
49. Zhao, K., & Ren, X. (2019, May). Small aircraft detection in remote sensing images based on YOLOv3. In IOP Conference Series: Materials Science and Engineering (Vol. 533, No. 1, p. 012056). IOP Publishing. https://doi.org/10.1088/1757-899X/533/1/012056.
50. Kim, J. A., Sung, J. Y., & Park, S. H. (2020, November). Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In 2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia) (pp. 1-4). IEEE. https://doi.org/10.1109/ICCE-Asia49877.2020.9277040.
51. Dorrer, M. G., & Tolmacheva, A. E. (2020, November). Comparison of the YOLOv3 and Mask R-CNN architectures' efficiency in the smart refrigerator's computer vision. In Journal of Physics: Conference Series (Vol. 1679, No. 4, p. 042022). IOP Publishing. https://doi.org/10.1088/1742-6596/1679/4/042022.
52. Rahman, E. U., Zhang, Y., Ahmad, S., Ahmad, H. I., & Jobaer, S. (2021). Autonomous vision-based primary distribution systems porcelain insulators inspection using UAVs. Sensors, 21(3), 974. https://doi.org/10.3390/s21030974.
53. Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., ... & Wen, S. (2020). PP-YOLO: An effective and efficient implementation of object detector. arXiv preprint arXiv:2007.12099. https://doi.org/10.48550/arXiv.2007.12099.
54. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed

and accuracy of object detection. arXiv preprint arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934.

55. Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430. https://doi.org/10.48550/arXiv.2107.08430.
56. Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs. Sensors, 22(2), 464. https://doi.org/10.3390/s22020464.
57. Kadry, S. N., Kyriazis, D., Varvarigou, T., & Konstanteli, K. G. (2014). Achieving real-time in distributed computing. International Journal of Grid and High Performance Computing, 6(1), 93-95.
58. Jiang, X., Wang, H., Chen, Y., Wu, Z., Wang, L., Zou, B., ... & Wu, Z. (2020). Mnn: A universal and efficient inference engine. Proceedings of Machine Learning and Systems, 2, 1-13.
59. Zhang, X., Wang, Y., & Shi, W. (2018). {pCAMP}: Performance Comparison of Machine Learning Packages on the Edges. In USENIX workshop on hot topics in edge computing (HotEdge 18).
60. Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q., & Alcaraz-Calero, J. M. (2022). Smartphone-based real-time object recognition architecture for portable and constrained systems. Journal of Real-Time Image Processing, 19(1), 103-115. https://doi.org/10.1007/s11554-021-01164-1.
61. Shahid, A., & Mushtaq, M. (2020, November). A survey comparing specialized hardware and evolution in TPUs for neural networks. In 2020 IEEE 23rd International Multitopic Conference (INMIC). 1-6. IEEE. https://doi.org/10.1109/INMIC50486.2020.9318136.
62. Karthikeyan, N. G. (2018). Machine Learning Projects for Mobile Applications: Build Android and IOS Applications Using TensorFlow Lite and Core ML. Packt Publishing Ltd.
63. Sosnovshchenko, O., & Baiev, O. (2018). Machine Learning with Swift: Artificial Intelligence for IOS. Packt Publishing Ltd.

64. Олійник, О. В., & Вовченко, А. М. (2021). Ефективність використання технології metal performance shaders для обробки числових даних. InterConf. 39. https://ojs.ukrlogos.in.ua/index.php/interconf/article/view/8096.
65. Thakkar, M. (2019). Beginning machine learning in iOS: CoreML Framework. Apress. https://doi.org/10.1007/978-1-4842-4297-1.
66. Ahremark, J., & Bazso, S. (2022). Benchmarking a machine learning model in the transformation from PyTorch to CoreML. https://www.diva-portal.org/smash/record.jsf?dswid=4230&pid=diva2%3A1703963.
67. Dewantoro, G., Mansuri, J., & Setiaji, F. D. (2021). Comparative Study of Computer Vision Based Line Followers Using Raspberry Pi and Jetson Nano. Jurnal Rekayasa Elektrika, 17(4). https://doi.org/10.17529/jre.v17i4.21324.
68. Puchtler, P., & Peinl, R. (2020, September). Evaluation of deep learning accelerators for object detection at the edge. In German Conference on Artificial Intelligence (Künstliche Intelligenz). Springer, Cham. 320-326. https://doi.org/10.1007/978-3-030-58285-2_29.
69. Hedman, J., & Xiao, X. (2016, January). Transition to the cloud: a vendor perspective. In 2016 49th Hawaii International Conference on System Sciences (HICSS). IEEE. 3989-3998. https://doi.org/10.1109/HICSS.2016.494.
70. Dua, R., Raja, A. R., & Kakadia, D. (2014, March). Virtualization vs containerization to support paas. In 2014 IEEE International Conference on Cloud Engineering. IEEE. 610-614. https://doi.org/10.1109/IC2E.2014.41.
71. Aguilera, L., & Jacobson, D. (2022, July). Deep Learning CNN Implementation on Packed Malware for Cloud Cross Domain Solution Filters. In 2022 International Conference on Data Science and Its Applications (ICoDSA). IEEE. 192-197. https://doi.org/10.1109/ICoDSA55874.2022.9862936.
72. Van Eyk, E., Iosup, A., Seif, S., & Thömmes, M. (2017, December). The SPEC cloud group's research vision on FaaS and serverless architectures. In Proceedings of the 2nd International Workshop on Serverless Computing. 1-4. https://doi.org/10.1145/3154847.3154848.
73. B. Gro√ü, M. Kreutzer and T. Durand, "OpenDataCam 3.0.2: An open source tool

to quantify the world".

74. Abdulghafoor, N. H., & ABDULLAH, H. N. (2021). Real-time moving objects detection and tracking using deep-stream technology. Journal of Engineering Science and Technology, 16(1), 194-208.
75. Mohammed, C. M., & Zeebaree, S. R. (2021). Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. International Journal of Science and Business, 5(2), 17-30.
76. Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal. 239(2). 2.
77. Xu, P., Shi, S., & Chu, X. (2017, August). Performance evaluation of deep learning tools in docker containers. In 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM). IEEE. 395-403. https://doi.org/10.1109/BIGCOM.2017.32.
78. Parmar, M. (2016). A survey of video object tracking methods. IJEDR. 4(1). 519-524. https://www.ijedr.org/papers/IJEDR1601086.pdf.
79. Patel, S. K., & Mishra, A. (2013). Moving object tracking techniques: A critical review. Indian Journal of Computer Science and Engineering, 4(2), 95-102.
80. Cavallaro, A., & Maggio, E. (2011). Video tracking: theory and practice. John Wiley & Sons.
81. Welch, G. F. (2020). Kalman filter. Computer Vision: A Reference Guide, 1-3. https://doi.org/10.1007/978-3-030-03243-2_716-1.
82. Taufique, A. M. N., Minnehan, B., & Savakis, A. (2020). Benchmarking deep trackers on aerial videos. Sensors, 20(2), 547. https://doi.org/10.3390/s20020547.
83. Zhang, X., Chen, X., Sun, W., & He, X. (2021). Vehicle Re-Identification Model Based on Optimized DenseNet121 with Joint Loss. Cmc-computers materials & continua, 67(3), 3933-3948. https://doi.org/ 10.32604/cmc.2021.016560.
84. Sun, W., Dai, L., Zhang, X., Chang, P., & He, X. (2022). RSOD: Real-time small object detection algorithm in UAV-based traffic monitoring. Applied Intelligence, 52(8), 8448-8463. https://doi.org/10.1007/s10489-021-02893-3.
85. Wu, M., Cao, X., & Guo, S. (2020). Accurate detection and tracking of ants in

indoor and outdoor environments. bioRxiv. https://doi.org/10.1101/2020.11.30.403816.

86. Cao, X., Guo, S., Lin, J., Zhang, W., & Liao, M. (2020). Online tracking of ants based on deep association metrics: method, dataset and evaluation. Pattern Recognition, 103, 107233. https://doi.org/10.1016/j.patcog.2020.107233.
87. Bochinski, E., Senst, T., & Sikora, T. (2018, November). Extending IOU based multi-object tracking by visual information. In 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) IEEE. 1-6. https://doi.org/ 10.1109/AVSS.2018.8639144.
88. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831. https://doi.org/10.48550/arXiv.1603.00831.
89. Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., ... & Leal-Taixé, L. (2021). Motchallenge: A benchmark for single-camera multiple target tracking. International Journal of Computer Vision, 129(4), 845-881. https://doi.org/10.1007/s11263-020-01393-0.
90. Пантелєєв, О. С., & Олійник, В. В. (2018). Метод визуального мультитрекинга в реальном времени на основе корреляционных фильтров. Адаптивні системи автоматичного управління, 1(32), 97-106. https://doi.org/10.20535/1560-8956.32.2018.145620.
91. Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010, June). Visual object tracking using adaptive correlation filters. In 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE. 2544-2550. https://doi.org/10.1109/CVPR.2010.5539960.
92. Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2012, October). Exploiting the circulant structure of tracking-by-detection with kernels. In European conference on computer vision Springer, Berlin, Heidelberg. 702-715. https://doi.org/10.1007/978-3-642-33765-9_50.
93. Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis

and machine intelligence, 37(3), 583-596. https://doi.org/10.1109/TPAMI.2014.2345390.

94. Danelljan, M., Häger, G., Khan, F. S., & Felsberg, M. (2016). Discriminative scale space tracking. IEEE transactions on pattern analysis and machine intelligence, 39(8), 1561-1575. https://doi.org/10.1109/TPAMI.2016.2609928.
95. Danelljan, M., Häger, G., Khan, F., & Felsberg, M. (2014). Accurate scale estimation for robust visual tracking. In British Machine Vision Conference, Nottingham, September. 1-5. https://doi.org/10.5244/C.28.65.
96. Danelljan, M., Robinson, A., Shahbaz Khan, F., & Felsberg, M. (2016, October). Beyond correlation filters: Learning continuous convolution operators for visual tracking. In European conference on computer vision. 472-488. https://doi.org/10.1007/978-3-319-46454-1_29.
97. Lukezic, A., Vojir, T., ˇCehovin Zajc, L., Matas, J., & Kristan, M. (2017). Discriminative correlation filter with channel and spatial reliability. In Proceedings of the IEEE conference on computer vision and pattern recognition. 6309-6318.
98. Liu, S., Liu, D., Srivastava, G., Połap, D., & Woźniak, M. (2021). Overview and methods of correlation filter algorithms in object tracking. Complex & Intelligent Systems, 7(4), 1895-1917. https://doi.org/10.1007/s40747-020-00161-4.
99. Wojke, N., & Bewley, A. (2018, March). Deep cosine metric learning for person re-identification. In 2018 IEEE winter conference on applications of computer vision (WACV). IEEE. 748-756. https://doi.org/10.1109/WACV.2018.00087.
100. Xiang, S., Nie, F., & Zhang, C. (2008). Learning a Mahalanobis distance metric for data clustering and classification. Pattern recognition, 41(12), 3600-3612. https://doi.org/10.1016/j.patcog.2008.05.018.
101. Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP) IEEE. 3645-3649. https://doi.org/10.1109/ICIP.2017.8296962.
102. Roderick, M., MacGlashan, J., & Tellex, S. (2017). Implementing the deep

q-network. arXiv. https://doi.org/10.48550/arXiv.1711.07478.

103. Do, C., Gordillo, C., & Burgard, W. (2018, October). Learning to pour using deep deterministic policy gradients. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 3074-3079. https://doi.org/ 10.1109/IROS.2018.8593654.

104. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347.

105. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. arXiv. https://doi.org/10.48550/arXiv.1812.05905.

106. Unlu, E., Zenou, E., Riviere, N., & Dupouy, P. E. (2019). Deep learning-based strategies for the detection and tracking of drones using several cameras. IPSJ Transactions on Computer Vision and Applications, 11(1), 1-13. https://doi.org/10.1186/s41074-019-0059-x.

107. Skrodzki, M. (2019). The kd tree data structure and a proof for neighborhood computation in expected logarithmic time. arXiv. https://doi.org/10.48550/arXiv.1903.04936.

108. Kuhn, H. W. (2010). The hungarian method for the assignment problem. In 50 Years of Integer Programming 1958-2008. Springer, 6, 29-47.

109. Melnyk, A. (2016). Cyber-physical systems multilayer platform and research framework. Advances in cyber-physical systems, (1, Num. 1), 1-6. https://doi.org/10.23939/acps2016.01.001.

110. Botchkaryov, O., Golembo, V., Paramud, Y., & Yazuk, V. (2019). Cyber-physical systems: technologies of data collection. monography–O. Botchkaryov, V. Golembo, Y. Paramud, V. Yazuk. Editorial chiev: prof. A. Melnyk. Lviv. Magnolia 2008. 10-12.

111. Koubâa, A., & Qureshi, B. (2018). Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet. IEEE Access, 6, 13810-13824. https://doi.org/10.1109/ACCESS.2018.2811762.

112. Ding, G., Wu, Q., Zhang, L., Lin, Y., Tsiftsis, T. A., & Yao, Y. D. (2018). An amateur drone surveillance system based on the cognitive Internet of Things. IEEE Communications Magazine, 56(1), 29-35. https://doi.org/10.1109/MCOM.2017.1700452.

113. Pons, P., Jaen, J., & Catala, A. (2015, November). Developing a depth-based tracking system for interactive playful environments with animals. In Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology. 1-8. https://doi.org/10.1145/2832932.2837007.

114. Пуйда, В. Я., & Стоян, А. О. (2020). Дослідження методів виявлення об’єктів на відеозображеннях. Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 2(1), 80-87. https://doi.org/10.23939/csn2020.01.080ю.

115. Пуйда, В. Я. (2022). Система технічного зору для досліджень в області дефектоскопії матеріалів та виробів. Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки, 4(1), 122130. https://doi.org/10.23939/csn2022.01.122.

116. Xu, Y., Osep, A., Ban, Y., Horaud, R., Leal-Taixé, L., & Alameda-Pineda, X. (2020). How to train your deep multi-object tracker. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 6787-6796.

117. Wang, C. Y., Liao, H. Y. M., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 390-391.

118. Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8759-8768). https://doi.org/10.1109/CVPR.2018.00913.

119. Arthur, D., & Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Stanford.

120. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020, April).

Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI conference on artificial intelligence. 34(7). 12993-13000. https://doi.org/10.1609/aaai.v34i07.6999.

121. Chaudhury, S., & Yamasaki, T. (2021). Robustness of adaptive neural network optimization under training noise. IEEE Access, 9, 37039-37053. https://doi.org/10.1109/ACCESS.2021.3062990.

122. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., ... & Ferrari, V. (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. International Journal of Computer Vision, 128(7), 1956-1981. https://doi.org/10.1007/s11263-020-01316-z.

123. Kushnir, D. (2022), Ants dataset (indoor/outdoor Messor Structor) + trained YOLOv4 weights. Mendeley Data. https://doi.org/10.17632/zprk7wkf9j.1.

124. D. T. Zutalin (2021). LabelImg. GitHub. https://github.com/tzutalin/labelImg.

125. Gantulga, B., Munkhtsetseg, N., Garmaa, D., & Batbayar, S. (2020). Using Five Principles of Object-Oriented Design in the Transmission Network Management Information. In Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceedings of the 15th International Conference on IIH-MSP in conjunction with the 12th International Conference on FITAT, July 18-20, Jilin, China, Volume 1 (pp. 325-333). Springer Singapore. https://doi.org/10.1007/978-981-13-9714-1_36.

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у фахових виданнях України:

- Kushnir, D., & Paramud, Y. (2020). The algorithm of Cyber-Physical system targeting on a movable object using the smart sensor unit. *Advances In Cyber-Physical Systems*, 5(1), 16-22. https://doi.org/10.23939/acps2020.01.016.
- Кушнір, Д. (2021). Методи та засоби покращення точності розпізнавання об’єктів на мобільній платформі iOS в реальному часі. *Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 3(1), 80-88. https://doi.org/10.23939/csn2021.01.080.
- Кушнір, Д., & Парамуд, Я. (2019). Методи пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі ios в реальному часі. *Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 1(1), 24-34. https://doi.org/10.23939/csn2019.01.024.
- Кушнір, Д., & Парамуд, Я. (2020). Алгоритм оперативного наведення засобів вимірюваьно – керувального вузла кіберфізичної системи на рухомий об’єкт. *Комп’ютерні системи та мережі. — Львів : Видавництво Львівської політехніки*, 2(1), 44-52. https://doi.org/10.23939/csn2020.01.044.

Статті у наукових періодичних виданнях інших держав, які входять до міжнародних наукометричних баз:

- Kushnir, D. (2022) Methods and means for small dynamic objects recognition and tracking. *Computers, Materials & Continua*, 73(2), 3649-3655. https://doi.org/10.32604/cmc.2022.030016.

Матеріали міжнародних наукових та науково-практичних конференцій, збірники яких входять до міжнародних наукометричних баз:

- Kushnir, D., & Paramud, Y. (2020). Model for real-time object searching and recognizing on mobile Platform. 2020 IEEE 15Th International Conference On Advanced Trends In Radioelectronics, Telecommunications And Computer Engineering (TCSET). *IEEE*, 127-130.

https://doi.org/10.1109/tcset49122.2020.235407.

- Kushnir, D., Ocherklevich, O., & Paramud, Y. (2021). Deep Neural Network Model for Text Semantic Analysis Based on Word Embeddings. *2021 11Th International Conference On Advanced Computer Information Technologies (ACIT)*. *IEEE*, 718-721. https://doi.org/10.1109/acit52158.2021.9548393.

Матеріали міжнародних наукових та науково-практичних конференцій:

- Vavruk, E., & Kushnir, D. (2018). Mobile system for text recognition and translation with using Microsoft Cognitive API. *8th International youth science forum «Litteris et Artibus»*, 81-84. https://ena.lpnu.ua/handle/ntb/51711.

Наукові праці, які додатково відображають наукові результати дисертації:

- Ваврук, Є., & Кушнір, Д. (2018). Система розпізнавання та перекладу текстової інформації в мобільних додатках з використанням бібліотеки Microsoft Cognitive OCR. *Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні системи та мережі*, 1(905), 33-42. https://doi.org/10.23939/csn2018.905.033.
- Andrushchak, N., Vynnyk, D., Melnyk, M., Bajurko, P., Kushnir, D., Haiduchok, V., ... & Yashchyshyn, Y. Impact of optical Illumination on transmission of subterahertz electromagnetic waves by Bi 12 GeO 20 crystals. *Acta Physica Polonica A*, 4(141), 415-419. https://doi.org/10.12693/APhysPolA.141.415
- Borak, T., Kushnir, D., & Paramud, Y. (2022). Microprocessor Subsystem of the Smart House to Control the Multichannel Irrigation of the Room Plants. *Advances In Cyber-Physical Systems*, 7(1), 1-7. https://doi.org/10.23939/acps2022.01.001.

ДОДАТОК Б. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ

«ЗАТВЕРДЖУЮ»
Проректор з наукової роботи
Національного університету
"Львівська політехніка"
_____ І.В. Демидов
"06" 09. 2022 р.

АКТ

використання наукових результатів дисертаційного дослідження аспіранта кафедри ЕОМ Кушніра Дмитра Олександровича на тему: «Методи та засоби пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі в реальному часі», представленої на здобуття ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія» при виконанні науково-дослідної роботи, яка фінансувалась за кошти державного бюджету МОН України

Комісія у складі голови – начальника НДЧ д.т.н., ст.досл. Небесного Р.В. та членів: зав. відділу науково-організаційного супроводу наукових досліджень к.т.н. Лазько Г.В., заст. начальника планово-фінансового відділу Чулой Т.М. та завідувача кафедри ЕОМ, д.т.н. професора Мельника А.О. цим актом підтверджують, що результати дисертаційного дослідження Кушніра Д.О., зокрема:

— модель для розпізнавання заданого класу елементів на твердотілих і нанокомпозитних матеріалах;
— метод фільтрації та мемоїзації результатів розпізнавання;
— засіб контейнеризації та масштабування системи розпізнавання,

використано у 2021 році при виконанні НДР «Інноваційне використання твердотілих і нанокомпозитних матеріалів для керування субтерагерцовим випроміненням» (шифр ДБ/СубТера, номер державної реєстрації 0119U100609) при дослідженні характеристик ТГЦ фільтрів та модуляторів та при розробленні засобів аналізу кристалічних матеріалів в ТГЦ діапазоні.

Отримані автором результати використано при розробленні матеріалів третього розділу «Застосування кристалічних матеріалів в ТГЦ діапазоні та потенційні споживачі виготовлених матеріалів та можливих рішень для керування субтерагерцовим випромінюванням» заключного звіту з НДР.

Голова комісії:

Начальник НДЧ
д.т.н., ст.досл. Р.В. Небесний

Члени комісії:
Зав. відділу НОСНД, к.т.н. Г.В. Лазько

Заст. нач. відділу ПФВ Т.М. Чулой

Завідувач кафедри ЕОМ,
д.т.н., професор А.О. Мельник

Керівник ДБ СубТера И.Н. Жидецький

«ЗАТВЕРДЖУЮ»
проректор з науково-педагогічної роботи Національного університету "Львівська політехніка"
_____ Олег ДАВЧИДЧАК
"__" квітня 2023 р.

АКТ

про впровадження в навчальний процес результатів дисертаційної роботи на здобуття наукового ступеня доктора філософії Кушніра Дмитра Олександровича

Цей акт складений про те, що результати дисертаційної роботи Кушніра Дмитра Олександровича на тему «Методи та засоби пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі в реальному часі», представленої на здобуття наукового ступеня доктора філософії, впроваджено у навчальний процес кафедри «Електронних обчислювальних машин» Національного університету "Львівська політехніка". Матеріали дисертаційного дослідження використовуються під час викладання дисципліни «Цифрова обробка сигналів».

Зокрема, у навчальному процесі використовуються запропонований Кушніром Дмитром Олександровичем:

– методи фільтрації результатів розпізнавання об'єктів на зображенні (дисципліна «Цифрова обробка сигналів» для студентів освітньо-кваліфікаційного рівня "бакалавр", що навчаються за напрямком 123 "Комп'ютерна інженерія", спеціалізацією 123.04 «Кіберфізичні системи», тема 13 «Опрацювання цифрових зображень»).

Завідувач кафедри електронних обчислювальних машин, д.т.н., професор — Анатолій МЕЛЬНИК

доцент кафедри електронних обчислювальних машин, к.т.н., доцент — Євгеній ВАВРУК

доцент кафедри електронних обчислювальних машин, к.т.н., доцент — Ярослав ПАРАМУД