ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Модель виявлення об'єктів на основі нейронної мережі»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології спеціальності
126 Інформаційні системи та технології ступеня вищої освіти
магістр
групи 126ІСТмд_22
Ярещенко Р. В.
Керівник: Слюсар В. І.
Рецензент: Муравльов В. В.

Полтава – 2023 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується розвитком додатків комп'ютерного зору на основі глибокого навчання для безпілотних та безекіпажних систем. Значне зменшення кількості аварій, адаптація для людей з обмеженими можливостями та зменшення заторів на дорогах – деякі з наочних прикладів цього. Враховуючи важливість безпілотних автомобілів, життєво важливо розробляти моделі глибокого навчання, які можуть точно сегментувати зручні для руху ділянки доріг а також об'єкти на них. Нещодавні досягнення в області глибокого навчання представили методи та техніки для ефективного вирішення завдань сегментації об'єктів на дорогах. Проте результати більшості з них не є задовільними для впровадження їх у практику. Все це свідчить про актуальність теми роботи.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в рамках науково-дослідної роботи «Управління стратегією інноваційного розвитку підприємств в контексті підвищення їх конкурентоспроможності на аграрному ринку, сталого розвитку та забезпечення продовольчої безпеки держави» (2021 р.), що фінансувалась господарськими договорами із замовниками, Концепції розвитку штучного інтелекту в Україні (розпорядження Кабінету Міністрів України № 1787-р від 29.12.2021), тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є підвищення ефективності методів сегментації проїзної частини доріг для виявлення на ній об'єктів.

Завданнями кваліфікаційної роботи є:

- аналіз особливостей реалізації сегментації об'єктів;
- розроблення моделі виявлення об'єктів на основі нейронної мережі;

– формування порівняльної оцінки продуктивності нейронних мереж на основі візуальних трансформерів;

– обґрунтування рекомендацій щодо вирішення завдання сегментації проїзної частини дороги.

Об'єктом дослідження є процес сегментації проїзного полотна за допомогою штучних нейронних мереж.

Предметом дослідження є точність сегментації нейронних мереж різної архітектури.

Методами дослідження в рамках визначення інструментарію для створення моделі виявлення обєктів, і техніко-економічного обґрунтування прийнятих рішень використовувався аналітичний метод досліджень, а для розробки архітектури нейронних мережі і формування датасету – моделювання.

Інформаційна база кваліфікаційної роботи сформована з ресурсів, що містять інформацію про архітектури YOLOv8, візуальні трансформери та компоненти, що їх реалізують, а також інструментарій для розробки та дослідження нейронних мереж.

Елементи наукової новизни роботи полягають у розробці моделі виявлення об'єктів на основі нейронної мережі Object Detection, порівняльній оцінка точності нейронних мереж на основі модифікованих архітектур SAM.

Практична значущість роботи полягає у розробці рекомендацій щодо використання запропонованої архітектури нейронної мережі в інтересах вирішення завдання сегментації проїзної частини дороги, які можуть бути використані для подальших досліджень за даною тематикою та при реалізації автономного водіння безпілотних транспортних засобів.

Апробація результатів відбувалася в рамках V-ої Міжнародної студентської конференції «Теоретичне та практичне застосування результатів сучасної науки» (жовтень 2023 р., м. Рівне), III-ої Міжнародної студентської конференції «Міждисциплінарні наукові дослідження та перспективи їх розвитку» (листопад 2023 р., м. Дніпро).

За результатами досліджень здійснено 2 публікації тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 75 сторінок формату А4. Вона містить 31 рисунок і 11 таблиць.

РОЗДІЛ 1
АНАЛІЗ ЗАВДАНЬ КОМП'ЮТЕРНОГО ЗОРУ В БЕЗПІЛОТНИХ ПЛАТФОРМАХ

1.1 Аналіз існуючих рішень з виявлення зони руху для безпілотних платформ

З появою штучного інтелекту (AI) і глибокого навчання, безпілотні автомобілі, дрони та літальні апарати (надалі, всі ці системи можна узагальнити терміном – UAV) набули популярності та привернули увагу як наукових кіл, так і промисловості [1]. Однією з головних причин такої еволюції є збільшення кількості дорожньо-транспортних пригод, які пов'язані з необережністю водія, включаючи перевищення швидкості, користування мобільним телефоном та стан сп'яніння за кермом [2]. У зв'язку з цим очікується, що зазначені платформи матимуть прогресивний, а також глибокий вплив на суспільство з точки зору безпечних і інтелектуальних послуг. Прийняття технології автономного керування може зустрітися з небажанням у громадському масштабі. Однак, UAV з технологією автономного керування стануть першою практичною інтеграцією соціально орієнтованих роботів у людське суспільство. За останні пару років автомобільна промисловість розвивалася навколо технології автономного водіння в обмеженому масштабі, включаючи утримання в смузі руху, адаптивний круїз-контроль і системи екстреного гальмування. Вони повинні бути оснащені надійними датчиками, сприйняттям і когнітивними технологіями, щоб задовольняти вимогам безпеки дорожнього середовища в реальному часі.

Зі швидким розвитком AI став стрімко розвиватись і комп'ютерний зір (CV). Глибоке навчання стало однією з основних технологій, яка дозволила безпілотним платформам виконувати численні завдання, зокрема виявлення та розпізнавання об'єктів, розуміння сцени та утримання смуги руху та ін.

Наприклад, модуль сприйняття UAV обробляє візуальний потік у режимі реального часу та виявляє розмітку дорожньої смуги, щоб оцінити кут повороту та прийняти оптимальну швидкість прискорення, бічний рух і можливе маневрування для забезпечення безпечної навігації по дорозі. При цьому, такі методи значною мірою покладаються на наявність розмітки дорожньої смуги для оцінки зони проїзду та оцінки кута повороту для контролю бокового руху UAV. На даний час, немає різноманітних загальнодоступних наборів даних, які б гарантували узагальнення методів оцінки кута повороту в складних дорожніх сценаріях у реальному часі. Крім того, розмітка смуги руху піддається впливу несприятливих погодних умов і з часом пошкоджується або зникає. Отже, згадані вище методи оцінки кута повороту UAV стають вразливими за постійно змінних умов освітленості та пов'язаних факторів, таких як наявність тіней, води, бруду на дорогах, що може призвести до смертельної ДТП на самокерованому автомобілі.

Виявлення дороги в реальному часі та керування поворотом вважаються ключовим елементом у виконанні завдань руху UAV. Тому доцільно проаналізувати існуючі методи виявлення дороги. Їх умовно можна розділити на групи:

1. Виявлення дороги на основі мультимодального датчика.
2. Виявлення дороги на основі монокулярного CV, що, в свою чергу, поділяється на виявлення дороги на основі смуги руху та виявлення дороги на основі семантичної сегментації.

1. Мультимодальне виявлення дороги на основі багатомодульного датчика. На початку розвитку UAV для виконання завдань водіння були запропоновані мультимодальні методи виявлення дороги на основі датчиків. Серед них 3D LiDAR використовувався для виявлення доріг завдяки його можливостям широкого діапазону вимірювань хмари точок. З цього приводу у [3] запропонований адаптивний метод виявлення дороги на основі хмари точок, який складається з трьох ключових кроків, а саме:

– виявлення точки розриву;

– ідентифікації регіону дороги;

– визначення зони руху.

В [4] автори запропонували наскрізну злиту мережу сегментації доріг для виявлення доріг. Запропонована мережа, заснована на мережі просторового поширення та трансформації, що складається з 3-ох частин:

– блоку хмари точок;

– блоку зображення;

– блоку синтезу.

Подібним чином, у [5] представлено метод виявлення доріг на основі LiDAR з використанням повністю згорткової нейронної мережі (FCN). У [6] розглянуто метод виявлення дороги на основі архітектури U-Net FCN. Однак, запропоновані архітектури навчені на обмеженому наборі даних. Таким чином, запропонований метод не здатний чітко виявляти дороги за різних погодних умов. В роботі [7] наведено метод поєднання інформації з камери та LiDAR на основі перетворення сферичних координат для виявлення дороги. Однак через обмежену кількість даних процес підвищення вибірки спостерігався з вищим рівнем хибнопозитивних і хибнонегативних результатів.

2. Монокулярне виявлення дороги на основі CV. Такі методи, зазвичай, базуються на методах машинного навчання, які вивчалися десятиліттями. Хоча існує багато типів датчиків, найпоширенішим для виявлення доріг є монокулярна камера, оскільки вона дешевша та більш інформативна [8]. Крім того, для досягнення наскрізного навчання автори застосували оптимізацію країв і геометричну деформацію, наприклад, метод виявлення дороги на основі нейронної мережі [9]. Однак запропоновані методи не застосовуються до різних дорожніх сцен у реальному світі. У [10] запропонували метод виявлення проїжджої частини дороги на основі сегментації екземплярів, що включає FCN і технологію відображення зворотної перспективи (IPM). Існує архітектура CNN для злиття даних Road-Seg, яка виділяє та поєднує особливості інформації про звичайну поверхню та зображення RGB для

точного виявлення дороги [11]. Так само, у [12] досліджено FCN для виявлення доріг. Однак запропонований метод має деякі обмеження, він не може правильно класифікувати різні типи дорожніх регіонів і не підходить для різних погодних умов.

В останні роки було запропоновано багато методів виявлення доріг на основі смуг руху. У зв'язку з цим у [13] представлений алгоритм виявлення дороги на основі CNN, враховуючи різні смуги руху. В дослідженні [14] запропонований метод виявлення дороги, заснований на алгоритмі покращення зображення для виявлення придатної для руху зони та смуг шосе в умовах поганого огляду (наприклад, у тумані або в вночі). Автори використали градації сірого та чіткість, щоб класифікувати зображення доріг на день, ніч, туман вдень і туман вночі. Однак продуктивність запропонованої техніки погано вплинула на безструктурні дороги, оскільки ці типи доріг не мають жодних меж доріг, дорожньої розмітки або фіксованих кольорів. Крім того, запропонований метод не зміг виявити дорогу в динамічному та складному середовищі, наприклад, за наявності водних плям, тіней на дорозі або наявності складних перешкод.

Методи виявлення доріг на основі семантичної сегментації були запропоновані для підвищення надійності існуючих систем виявлення доріг на основі CV. У [15] представлено метод виявлення дороги на основі семантичної сегментації для виявлення вільної зони, придатної для руху, для самокерованих автомобілів. У [16] розглянутий метод Hybrid-SRD на основі Transfer Learning для виявлення дороги. Запропонована архітектура поєднує операції згортки та підвищення вибірки. У [17] автори навели метод виявлення доріг, що складається з результатів моделей семантичної сегментації глибокого навчання для структурованих і неструктурованих доріг. В свою чергу, в [18] запропоновано мережу семантичної сегментації EdgeNet для визначення зони водіння для самокерованих автомобілів. Запропонована мережа EdgeNet, заснована на техніці coder-decoder, містить модуль аналізу каналів і втрат фронтів для підвищення точності запропонованої мережі. В

іншому дослідженні запропонований метод на основі семантичної сегментації для виявлення засніжених доріг для безпілотних автомобілів [19]. У даній структурі використовувалася піраміда і мережа атральної згортки для вилучення характеристик з зображень снігової дороги. Однак запропонований метод не забезпечує задовільного результату в умовах освітлення, наприклад, результуюча втрата зростає порівняно зі звичайними умовами, коли сонячне світло відбивається від снігу. В роботі [20] розглянутий метод виявлення дороги на основі колірної сегментації HSV для самокерованих автомобілів. Щоб отримати ширину дороги, автори використали детектор країв і метод перетворення Хафа. Однак запропонований метод застосовний лише до прямих доріг і не зміг виявити вигнуті дороги. Крім того, було помічено, що існує помилка виявлення в кінці дороги та на перехрестях доріг.

На основі проведених досліджень можливо зробити висновок, що для реалізації автономного руху UAV необхідно визначити зону, в який він може вільно пересуватись. Для формування цієї зони потрібно, щоб UAV міг самостійно виявляти об'єкти, тобто, вирішувати завдання Object Detection. Найбільш вдалим інструментом для цього є AI на основі нейронної мережі. На даний час, актуальною архітектурою для Object Detection є YOLOv8, окремі версії якої дозволяють проводити процедури сегментації. Як наслідок, надалі доцільно розглянути зазначену архітектуру.

1.2 Архітектура YOLOv8

YOLOv8 (рис. 1.1) – це остання версія в серії YOLO для Object Detection у реальному часі, яка забезпечує передову продуктивність у термінах точності та швидкості [21]. Ґрунтуючись на досягненнях попередніх версій, YOLOv8 вводить нові можливості та оптимізації для різних завдань щодо Object Detection у широкому спектрі додатків. використовує передові архітектури магістралі та сегменту вводу, що призводить до покращеного

вилучення ознак та продуктивності виявлення об'єктів. YOLOv8 застосовує сегментацію вводу від к. Ultralytics, що сприяє більш точному виявленню та ефективнішому процесу виявлення ознак.

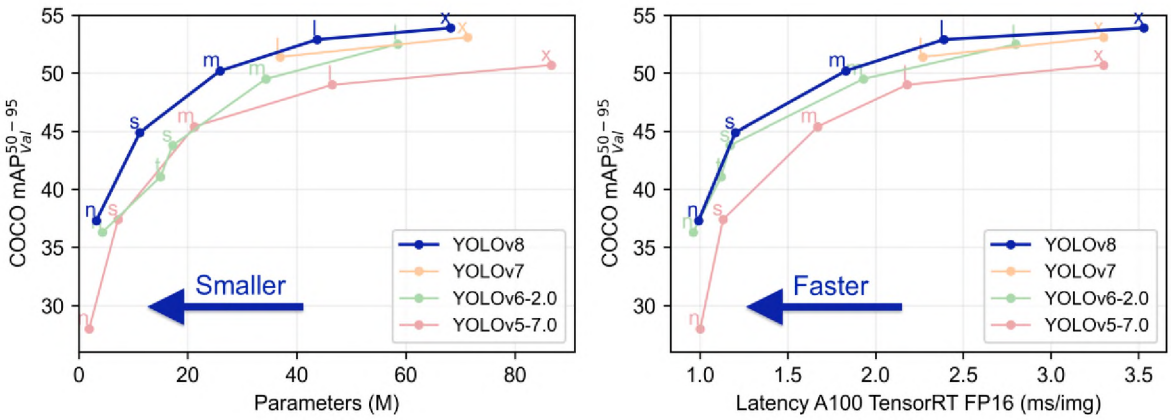


Рисунок 1.1 – Порівняння YOLOv8 з попередніми версіями

Дана версія YOLO має підтримує оптимальний баланс між точністю та швидкістю та підходить для задач виявлення об'єктів у режимі реального часу в різних сферах застосування. YOLOv8 пропонує різноманітну низку попередньо навчених моделей для різних завдань і вимог до продуктивності, що спрощує вибір відповідної моделі для конкретного випадку використання. Ці моделі спеціалізовані для конкретних завдань CV. На рис. 1.2 наведений огляд варіантів моделей, що демонструє гнучкість та надійність серії YOLOv8, а саме моделі для завдань виявлення, сегментації, визначення пози – попередньо навчені на наборі даних COCO, тоді як моделі для класифікації – попередньо навчені на наборі даних ImageNet. Це робить їх придатними для широкого спектру додатків у CV. Ці моделі розроблені для задоволення різних вимог, від Object Detection до складніших завдань, таких як сегментація екземплярів, визначення пози (ключових точок) та класифікація. Кожна варіація серії YOLOv8 оптимізована для відповідного завдання, забезпечуючи високу продуктивність і точність. Крім того, ці моделі сумісні з існуючою множиною режимів роботи, включаючи вивід, перевірку, навчання та експорт, що полегшує їх використання на різних етапах розгортання та

розробки. Надалі доцільно визначити основні властивості окремих варіантів мереж за вказаними завданнями.

Виявлення	Сегментація екземплярів	Поза/ключові точки	Класифікація
YOLOv8n YOLOv8s YOLOv8m YOLOv8l YOLOv8x	YOLOv8n-seg YOLOv8s-seg YOLOv8m-seg YOLOv8l-seg YOLOv8x-seg	YOLOv8n-pose YOLOv8s-pose YOLOv8m-pose YOLOv8l-pose YOLOv8x-pose YOLOv8x-pose-p6	YOLOv8n-cls YOLOv8s-cls YOLOv8m-cls YOLOv8l-cls YOLOv8x-cls

Рисунок 1.2 – Специфікація серії YOLOv8

Виявлення об'єктів – це завдання, яке включає ідентифікацію розташування та класу об'єктів на зображенні чи відео (рис. 1.3).

Рисунок 1.3 – Object Detection

Результат роботи детектора об'єктів – це набір рамок, що обмежують, які містять у собі об'єкти на зображенні, разом з мітками класів і рівнями достовірності для кожної рамки. Виявлення об'єктів є добрим вибором, коли необхідно визначити об'єкти інтересу в сцені, але не потрібно точно знати, де знаходиться об'єкт або його точну форму. В табл. 1.1 наведено основні відомості про моделі для Object Detection. Значення mAP^{val}_{50-95} наведено для одиночної моделі одиночного масштабу на датасеті COCO val2017. Швидкість усереднена за зображеннями COCO val на екземплярі Amazon EC2 P4d. Моделі автоматично завантажуються з останнього релізу Ultralytics Release під час першого використання.

Таблиця 1.1 – Характеристика YOLOv8 для Object Detection

Модель	Розмір (пікселів)	mAP^{val}_{50-95}	Швидкість для CPU ONNX (мс)	Швидкість для A100 TensorRT (мс)	Параметри ($\times 10^6$)	FLOPs (Блоки)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Крім вказаних моделей, на сайті к. Ultralytics пропонується також архітектура YOLO-NAS (рис. 1.4), яка розроблена к. Deci AI.

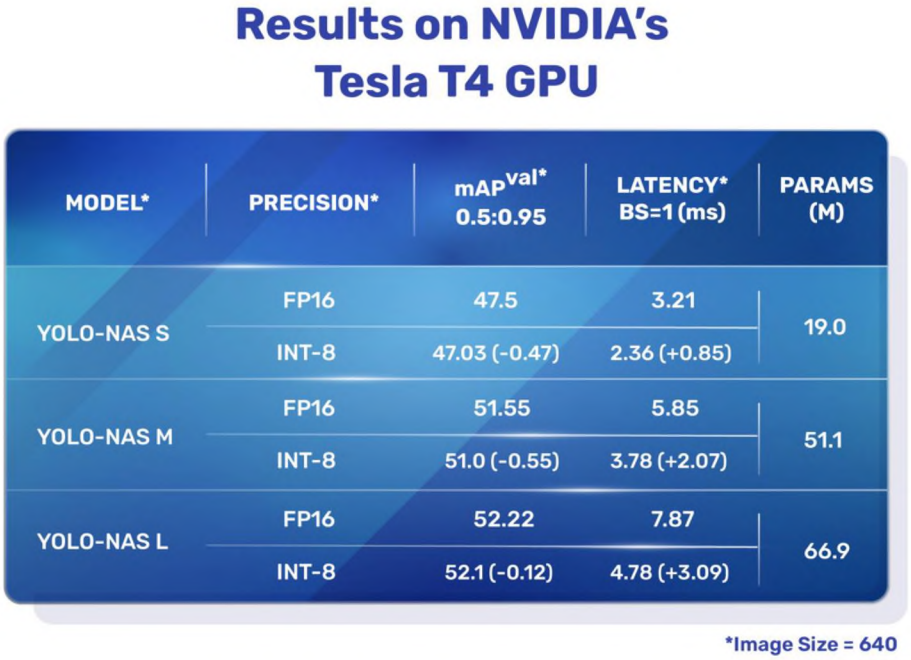


Рисунок 1.4 – YOLO-NAS

Вона є продуктом сучасної технології пошуку нейронних архітектур і спеціально розроблена для подолання обмежень попередніх моделей YOLO. Завдяки істотному поліпшенню підтримки квантування та компромісу між точністю та затримкою, YOLO-NAS є значним проривом у області виявлення об'єктів. Ця архітектура використовує блоки, що підтримують квантування, та селективне квантування для досягнення оптимальної продуктивності. Модель, коли перекладається квантовану версію INT8, має мінімальне падіння точності, що є значним поліпшенням порівняно з іншими моделями.

Ці досягнення призводять до оптимальної архітектури з можливостями виявлення об'єктів та відповідною продуктивністю. Вона має такі основні функції:

– базовий блок підтримує квантування, що дозволяє подолати одне із значних обмежень попередніх моделей YOLO;

– YOLO-NAS використовує просунуті схеми тренування та пост-тренувальне квантування для покращення продуктивності.

YOLO-NAS використовує оптимізацію AutoNAC та попереднє навчання на відомих наборах даних, таких як COCO, Objects365 та Roboflow 100. Це попереднє навчання робить модель оптимально придатною для рішень щодо виявлення об'єктів у виробничих середовищах. В табл. 1.2 наведено основні характеристики YOLO-NAS. Кожен варіант моделі розроблено для досягнення балансу між середньою точністю виявлення (mAP) та затримкою, допомагаючи оптимізувати завдання щодо виявлення об'єктів з точки зору продуктивності та швидкості.

Таблиця 1.2 – Характеристики YOLO-NAS

Модель	mAP	Затримка (мс)
YOLO-NAS S	47.5	3.21
YOLO-NAS M	51.55	5.85
YOLO-NAS L	52.22	7.87
YOLO-NAS S INT-8	47.03	2.36
YOLO-NAS M INT-8	51.01	3.78
YOLO-NAS L INT-8	52.1	4.78

К. Ultralytics зробила інтеграцію моделей YOLO-NAS у програми користувача на Python максимально простою. Для цього необхідно підключити пакет ultralytics, який надає зручний API на Python, щоб спростити весь процес.

Класифікація зображень – це найпростіше із 3-ох завдань і полягає у класифікації всього зображення за одним із попередньо визначених класів (рис. 1.5). Вихід класифікатора зображень – це один класовий ярлик та рівень довіри. Класифікація зображень корисна, коли вам потрібно знати тільки до

якого класу відноситься зображення, і не потрібно знати, де знаходяться об'єкти даного класу або яка їхня точна форма. Моделі YOLOv8 (табл. 1.3) для класифікації навчаються на наборі даних ImageNet. Значення точності вказують на точність моделі на валідаційному наборі даних ImageNet. Швидкість усереднена за зображеннями для валідації ImageNet, використовуючи екземпляр Amazon EC2 P4d.

Рисунок 1.5 – Завдання класифікації зображень

Таблиця 1.3 – Характеристики YOLO для класифікації

Модель	Розмір (пікселів)	Точність top5	Швидкість для CPU ONNX (мс)	Швидкість для A100 (мс)	Параметри ($\times 10^6$)	FLOPs (Блоки)
YOLOv8n-cls	224	87.01	12.9	0.31	2.7	4.3
YOLOv8s-cls	224	91.1	23.4	0.35	6.4	13.5
YOLOv8m-cls	224	93.2	85.4	0.62	17.01	42.7
YOLOv8l-cls	224	94.1	163.01	0.87	37.5	99.7
YOLOv8x-cls	224	94.3	232.01	1.01	57.4	154.8

Оцінка пози – це завдання, що полягає у визначенні розташування певних точок на зображенні, які, зазвичай, називають контрольними точками (рис. 1.6). Контрольні точки можуть представляти різні частини об'єкта, такі як суглоби, орієнтири та інші характерні риси. Розташування контрольних точок, зазвичай, представлено у вигляді набору координат: 2D $[x, y]$ або 3D $[x, y, visible]$. Результат роботи моделі оцінки пози – це набір точок, що представляють контрольні точки на об'єкті в зображенні, зазвичай, разом з

оцінками впевненості для кожної точки. В табл. 1.4 наведено характеристики YOLOv8 для оцінки пози.

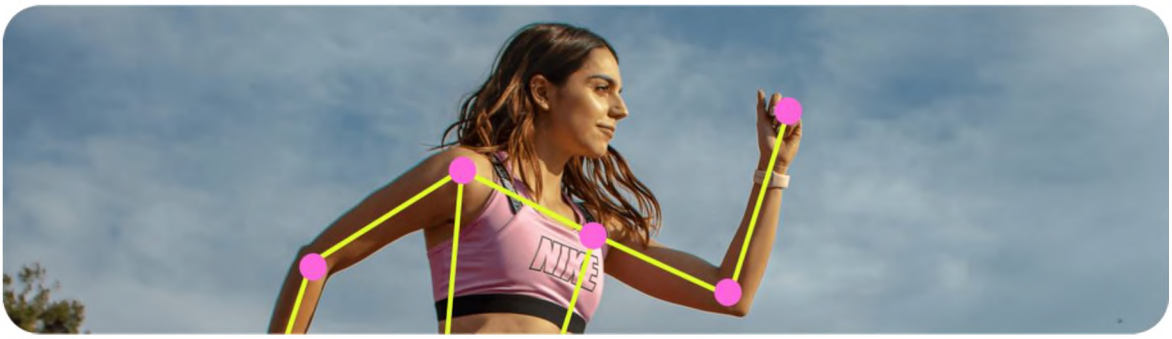


Рисунок 1.6 – Оцінка пози

Оцінка пози є хорошим вибором, коли вам потрібно ідентифікувати конкретні частини об'єкта в сцені та їх розташування відносно один одного. Оцінка пози може стати в нагоді при вирішенні завдань прогнозу траєкторії руху людей з метою забезпечення їх безпеки у випадку перетинання з траєкторією руху UAV.

Таблиця 1.4 – Характеристики YOLO для оцінки пози

Модель	Розмір (пікселів)	$mAP^{pose}_{50–95}$	Швидкість для CPU ONNX (мс)	Швидкість для A100 (мс)	Параметри ($\times 10^6$)	FLOPs (Блоки)
YOLOv8n-pose	640	80.1	131.8	1.18	3.3	9.2
YOLOv8s-pose	640	86.2	233.2	1.42	11.6	30.2
YOLOv8m-pose	640	88.8	456.3	2	26.4	81.0
YOLOv8l-pose	640	90.01	784.5	2.59	44.4	168.6
YOLOv8x-pose	640	90.2	1607.1	3.73	69.4	263.2
YOLOv8x-pose-p6	1280	91.2	4088.7	10.04	99.1	1066.4

Значення $mAP^{pose}_{50–95}$ для однієї моделі одиночного масштабу на наборі даних COCO Keypoints val2017. Швидкість усереднена за зображеннями COCO val на Amazon EC2 P4d інстансі. Для перетворення наявного набору даних з інших форматів (наприклад, COCO та ін.) у формат YOLO можна використовувати інструмент JSON2YOLO від к. Ultralytics.

1.3 Особливості реалізації сегментації

Сегментація, тобто розпізнавання пікселів зображення, що належать об'єкту, – базове завдання CV, яке використовується в широкому спектрі застосувань, від аналізу наукових знімків до редагування фотографій. На даний час, поширено використовується наскрізна сегментація на рівні пікселів на основі виявлення дороги та метод оцінки кута повороту для безпілотних автомобілів разом із точним анотованим набором даних на рівні пікселів для підтримки надійності під час руху по неструктурованих дорогах. дороги. Однак, вищезазначені завдання створюють кілька ускладнень, включаючи створення набору даних, попередню обробку даних, створення наземних істинних даних для розробки систем сегментації доріг і системи обчислення кута повороту. Як наслідок, може використовуватись конвеєр для вирішення цих складних завдань, як показано на рис. 1.7.

При цьому, U-Net є класичним алгоритмом для сегментації зображення з використанням FCN [22]. Основною особливістю U-Net-подібних мереж є симетричні пропускаючі з'єднання, які об'єднують низькорівневі карти функцій кодера з високорівневими картами функцій декодера. Просторова інформація, яка сприяє точності локалізації на рівні пікселів, поширюється з низькорівневих карт функцій і агрегується в контекстну інформацію високого рівня. На кожній стадії кодера застосовуються два згорткових шари 3×3 і функція активації ReLu, а потім використовується шар максимального об'єднання 2×2 для зменшення дискретизації сформованих карт функцій [23].

У частині декодера вихід кодера спочатку підвищується за допомогою операції DeConv, а потім результуючий вихід об'єднується з виходом каскаду дзеркального кодера перед обробкою за допомогою двох згорткових шарів 3×3 і функції активації ReLU. Нарешті, кожного разу, коли карти функцій знижуються за допомогою операції максимального об'єднання, деякі крайові функції обов'язково будуть втрачені, і втрачені функції не можна відновити під час операції підвищення дискретизації. Таким чином, щоб відновити

втрачені крайові характеристики, в оригінальній U-Net використовується метод зшивання ознак [23].

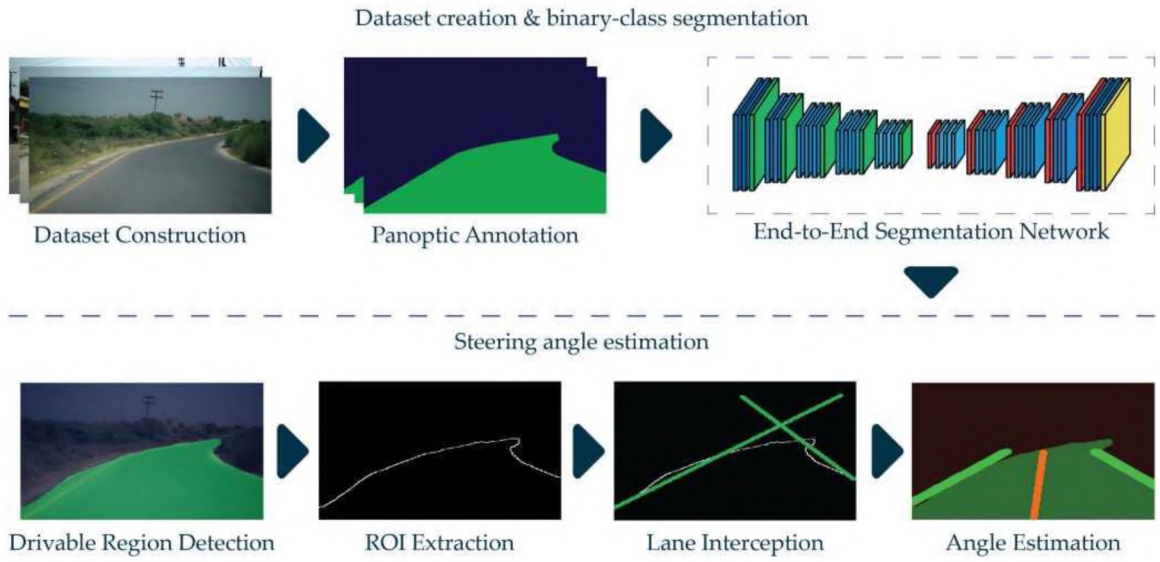


Рисунок 1.7 – Діаграма методу наскрізної сегментації

U-Net з механізмом уваги. Мета механізму уваги полягає в тому, щоб зосередитися на ключовій інформації та відкинути інші частини зображення. Тобто, сутність модуля уваги полягає в посиленні регіонів інтересу, одночасно пригнічуючи певні регіони, що не представляють інтерес [24]. Додавання механізму уваги може призвести до значного покращення рівня точності сегментації зображення та зменшує обчислювальні витрати. Механізм потрійної уваги (рис. 1.8) є одним з методів, які обчислюють ваги уваги шляхом захоплення міжвимірних взаємодій через структуру триплетної гілки [25]. Традиційна техніка обчислення уваги каналу включає спочатку обчислення вагів, а потім використання цих вагів для рівномірного масштабування цих карт функцій. Однак, важливо зазначити, що цей підхід вимагає просторового розкладання вхідного тензора на один піксель шляхом глобального середнього об'єднання, щоб визначити ваги для цих каналів. Оскільки немає взаємозалежності між розмірністю каналу та просторовим виміром при обчисленні уваги на одному піксельному каналі, це може призвести до великої втрати просторової інформації [26, 27].

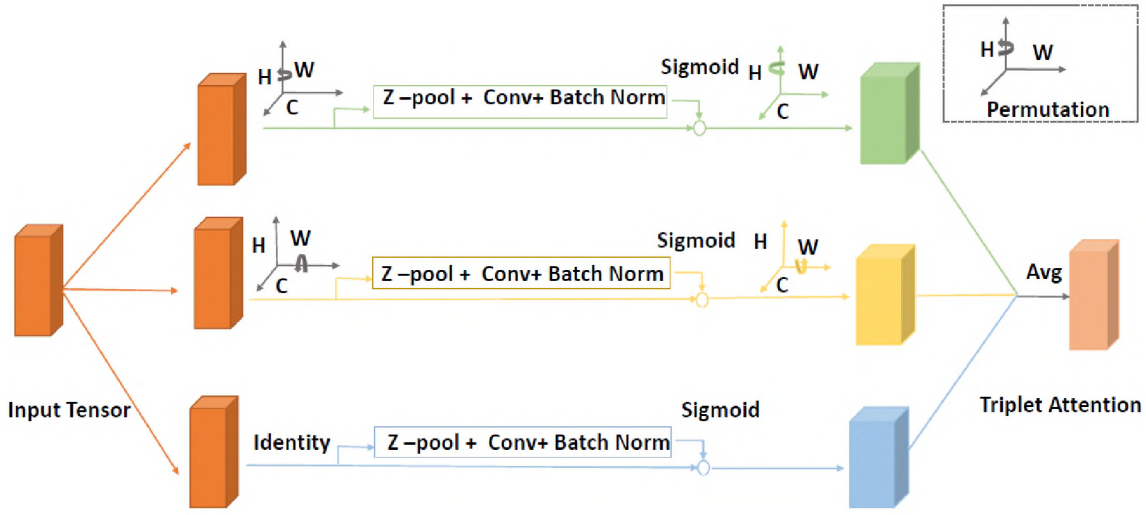


Рисунок 1.8 – Механізм потрійної уваги

Таким чином, концепція міжвимірної взаємодії була введена в триплетний механізм уваги, який дозволяє полегшити проблему втрати просторової інформації шляхом фіксації взаємодії між просторовим виміром і виміром вхідного тензорного каналу. Триплетний механізм уваги складається з кількох паралельних гілок (рис. 1.9).

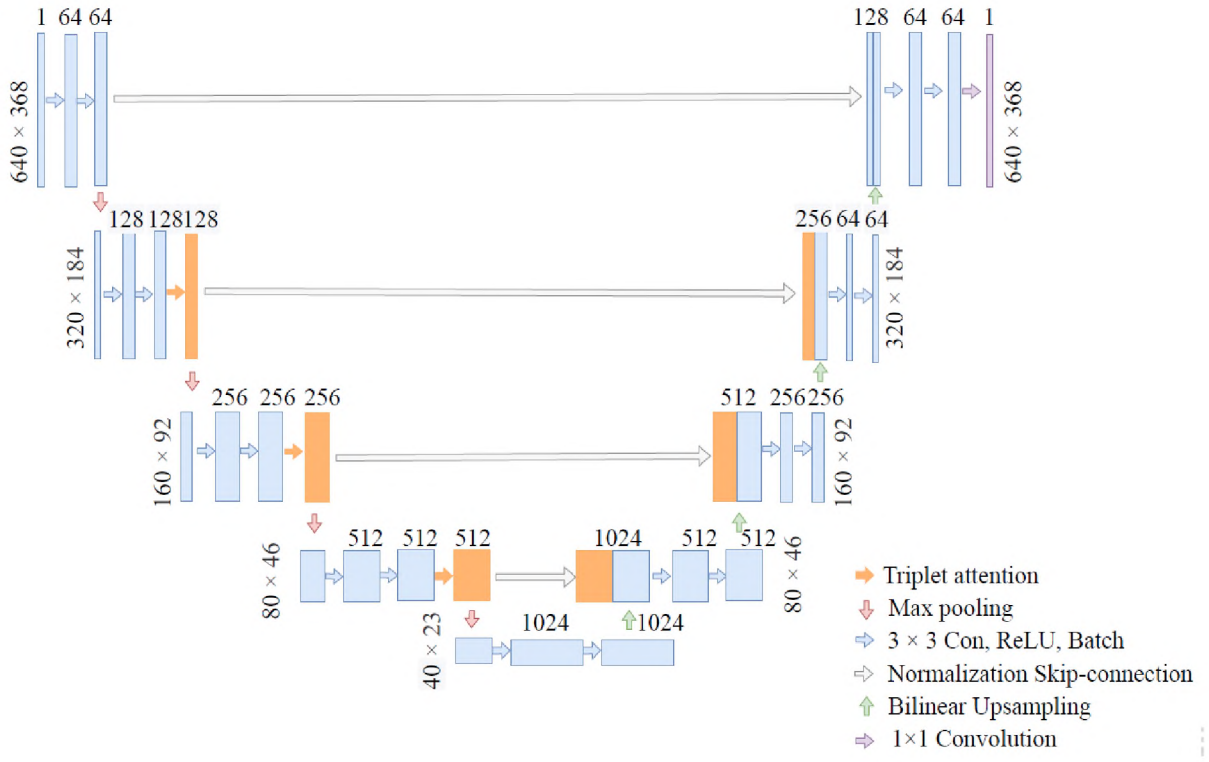


Рисунок 1.9 – U-Net з механізмом потрійної уваги

Форма результуючих виходів усіх гілок однакова. Щоб отримати остаточний результат механізму триплетної уваги, ми просто беремо середнє значення суми окремих вихідних даних гілок. Крім того, щоб обчислити увагу каналу, ми використовуємо сингулярні ваги, що вважається легким і ефективним методом. Зокрема, операція виконується шляхом введення скалярів для кожного каналу в тензорі, а потім використання сингулярних ваг для рівномірного масштабування цих карт функцій.

На практиці, ці сингулярні ваги обчислюються шляхом просторового розкладання вхідного тензора на один піксель на канал за допомогою глобального середнього об’єднання, що призводить до значної втрати просторової інформації [25].

Автори триплетної уваги представили модуль просторової уваги як додатковий метод для звернення уваги на окремі канали пікселів. Фактично, просторова увага зосереджується на розташуванні в каналі, а увага каналу спрямована на зосередження на каналі, дозволяючи взаємодію між виміром каналу та просторовим виміром.

Таким чином, триплетна увага – це потужний модуль уваги, який фіксує важливі характеристики в різних вимірах і обчислюється за допомогою каналу уваги та просторової уваги. Існуючи дослідження демонструють перевагу над базовими архітекторами нейронних мереж з точки зору таких показників, як точність пікселів і середнє перетин через об’єднання.

Крім того, U-Net з триплетною увагою створює відносно чіткіші карти сегментації за різних погодних умов. Крім того, застосування функцій втрати суміші може призвести до підвищення продуктивності. Незважаючи на те, що розмір параметра мережі менший, ніж у базових варіантах, сегментація в реальному часі все ще є дорогою з точки зору обчислень.

Також досить актуальною є сегментація екземплярів (рис. 1.10), яка йде на крок далі в порівнянні з виявленням об'єктів і включає ідентифікацію окремих об'єктів на зображенні та їх сегментацію від решти зображення.

Рисунок 1.10 – Сегментація екземплярів

Результатом моделі сегментації екземплярів є набір масок або контурів, що окреслюють кожен об'єкт на зображенні, разом із класовими мітками та коефіцієнтами впевненості для кожного об'єкта. Сегментація екземплярів корисна, коли потрібно знати не тільки де знаходяться об'єкти на зображенні, але і їх точну форму.

Висновки до розділу 1

Проведено аналіз впливу розвитку та застосування AI і глибокого навчання в UAV, а також обґрунтовано важливість цих технологій у покращенні безпеки на дорогах, зменшуючи аварії, спричинені необережністю водіїв. При цьому, досліджено потенційний вплив технологій автономного керування на суспільство, незважаючи на можливий опір громадськості щодо їхнього впровадження.

Для забезпечення автономного руху безпілотних апаратів (UAV), важливо, щоб вони могли самостійно ідентифікувати своє оточення. Це досягається за допомогою виявлення об'єктів, де AI на основі нейронних мереж грає ключову роль. Серед сучасних архітектур, YOLOv8 виділяється своєю здатністю оптимально балансувати точність та швидкість, що робить її ідеальною для виявлення об'єктів у реальному часі. YOLOv8 також пропонує широкий спектр попередньо навчених моделей для різних завдань, таких як виявлення об'єктів, сегментація і класифікація.

Класичним алгоритмом для сегментації зображення є мереж з архітектурою U-Net. В них може застосовуватись механізм уваги. Мета застосування такого механізму полягає в тому, щоб зосередитися на ключовій інформації та відкинути інші частини зображення. Його додавання може призвести до значного покращення рівня точності сегментації зображення та зменшення обчислювальних витрат. При цьому дослідження особливості реалізації потрійного механізму уваги, який фіксує важливі характеристики в різних вимірах і обчислюється за допомогою каналу уваги та просторової уваги. Для виявлення перешкод під час руху UAV в роботі запропоновано систему, що спирається на застосування нейронних мереж для вирішення завдань сегментації.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ВИЯВЛЕННЯ ОБ'ЄКТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

2.1 Застосування YOLOv8 для сегментації

YOLOv8 – це сучасна модель виявлення об’єктів, яка також може виконувати семантичну сегментацію вхідного зображення (рис. 2.1). Семантична сегментація – це завдання присвоєння мітки класу кожному пікселю на зображенні, наприклад людина, автомобіль, небо тощо.

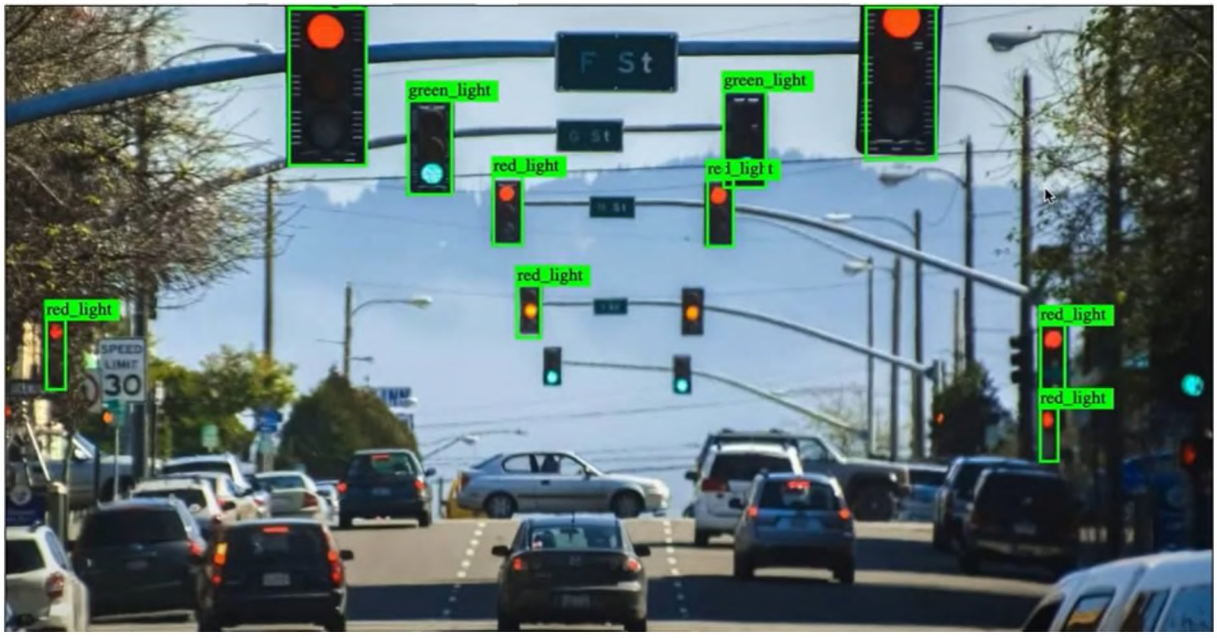


Рисунок 2.1 – Приклад сегментації світлофорів за допомогою YOLOv8

Це дозволяє моделі ідентифікувати точну форму та розташування кожного об’єкта на зображенні, а також відокремлювати різні примірники одного класу. YOLOv8 для сегментації використовує нову архітектуру, яка складається з екстрактора функцій CSPDarknet53, модуля C2f і модуля SegHead. CSPDarknet53 – це магістральна мережа, яка витягує функції високого рівня з вхідного зображення. Модуль C2f – це міжмасштабний модуль злиття, який поєднує функції з різних масштабів і покращує

представлення функцій. Модуль SegHead – це голова сегментації, яка передбачає маску сегментації для кожного класу та кожного масштабу. YOLOv8 для сегментації навчається на наборі даних COCO, який містить 80 класів об'єктів і 118000 зображень. Модель (табл. 2.1) досягає середньої точності (mAP) 36,8 % для сегментації та 44,6 % для виявлення з роздільною здатністю 640×640. Модель також дуже швидка: 155,7 мс на CPU та 1,47 мс на GPU. Значения для одиночной модели одиночного масштаба на наборе данных COCO val2017. Значення mAP^{val} для одиночної моделі поодинокого масштабу на наборі даних COCO val2017. Швидкість усереднена для зображень COCOval на екземплярі Amazon EC2 P4.

Таблиця 2.1 – Характеристика YOLOv8-seg

Модель	Розмір (пкс)	mAP^{box}_{50-95}	mAP^{mask}_{50-95}	Швидкість для CPU ONNX (мс)	Швидкість для A100 TensorRT (мс)	Параметри ($\times 10^6$)	FLOPs (Блоки)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Як відомо, Amazon EC2 P4d – це тип віртуального сервера, наданого Amazon Web Services (AWS), спеціально розробленого для машинного навчання та високопродуктивних обчислювальних завдань. Екземпляри P4d оснащені GPU NVIDIA A100 Tensor Core. Ці GPU високоефективні для робочих навантажень із глибоким навчанням і машинним навчанням, пропонуючи значні покращення потужності обробки та швидкості порівняно з попередніми поколіннями. P4d добре підходять для завдань високопродуктивних обчислень (HPC), наприклад, обчислювальна гідродинаміка, обчислювальне фінансування, сейсмічний аналіз та ін. завдання, які потребують великої обчислювальної потужності. Екземпляри P4d пропонують мережу з високою пропускною здатністю, що має вирішальне значення для масштабування робочих навантажень машинного

навчання та HPC. Вони підтримують адаптер AWS Elastic Fabric Adapter (EFA), який покращує здатність запускати програми, які вимагають високого рівня міжвузлового зв'язку. Вказані екземпляри мають значний обсяг пам'яті, що робить їх придатними для роботи з великими наборами даних, що часто є вимогою в задачах машинного навчання та аналізу даних. Як і інші екземпляри EC2, екземпляри P4d пропонують гнучкість збільшення або зменшення масштабу залежно від попиту, що допомагає оптимізувати витрати та продуктивність для різних робочих навантажень. P4d можна легко інтегрувати з іншими службами AWS, покращуючи їх функціональність і зручність використання. Це включає такі служби, як Amazon S3 для зберігання, Amazon RDS для баз даних і AWS Lambda для запуску коду у відповідь на події. Типові випадки використання екземплярів P4d включають навчання складних моделей машинного навчання, запуск великомасштабних паралельних симуляцій, візуалізацію графіки з високою роздільною здатністю та обробку великомасштабних геномних даних тощо. В цілому, екземпляри Amazon EC2 P4d є потужним ресурсом для компаній і дослідників, яким потрібні розширені обчислювальні можливості, особливо в таких сферах, як AI та HPC. Моделі YOLO можна використовувати в різних режимах залежно від конкретної проблеми, яку ви намагаєтеся вирішити. Ці режими включають:

– Train – для навчання моделі YOLOv8 на спеціальному наборі даних;

– Val – для перевірки моделі YOLOv8 після її навчання;

– Predict – для прогнозування за допомогою навченої моделі YOLOv8 на нових зображеннях або відео;

– Export – для експорту моделі YOLOv8 у формат, який можна використовувати для розгортання;

– Track – для відстеження об'єктів у реальному часі за допомогою моделі YOLOv8;

– Benchmark – для порівняльного аналізу швидкості та точності експорту YOLOv8 (ONNX, TensorRT тощо).

Навчальні налаштування для моделей YOLO охоплюють різні гіперпараметри та конфігурації, які використовуються під час процесу навчання. Ці параметри впливають на продуктивність, швидкість і точність моделі. Основні налаштування навчання включають розмір партії, швидкість навчання, імпульс і зниження ваги. Крім того, вибір оптимізатора, функції втрат і складу навчального набору даних може вплинути на процес навчання. Ретельне налаштування та експериментування з цими налаштуваннями мають вирішальне значення для оптимізації продуктивності.

Параметри передбачення для моделей YOLO охоплюють низку гіперпараметрів і конфігурацій, які впливають на продуктивність, швидкість і точність моделі під час висновку на основі нових даних. Ретельне налаштування та експериментування з цими параметрами необхідні для досягнення оптимальної продуктивності для конкретного завдання. Основні налаштування включають поріг довіри, поріг немаксимального придушення (NMS) і кількість класів, що розглядаються. Додатковими факторами, що впливають на процес передбачення, є розмір і формат вхідних даних, наявність додаткових функцій, таких як маски або кілька міток на поле, і конкретне завдання, для якого використовується модель.

Параметри доповнення для моделей YOLO стосуються різноманітних перетворень і модифікацій, застосованих до навчальних даних для збільшення різноманітності та розміру набору даних. Ці налаштування можуть впливати на продуктивність, швидкість і точність моделі. Деякі звичайні параметри доповнення YOLO включають тип та інтенсивність застосованих трансформацій (наприклад, випадкові повороти, обертання, кадрування, зміни кольору), ймовірність застосування кожної трансформації та наявність додаткових функцій, таких як маски або кілька міток на поле . Інші фактори, які можуть вплинути на процес розширення, включають розмір і склад вихідного набору даних і конкретне завдання, для якого використовується модель. Важливо ретельно налаштувати та експериментувати з цими параметрами, щоб переконатися, що доповнений

набір даних є різноманітним і достатньо репрезентативним для навчання високоефективної моделі. Ведення журналів, контрольні точки, побудова графіків і керування файлами є важливими положеннями під час навчання моделі YOLO. Корисно реєструвати різні показники та статистичні дані під час навчання, щоб відстежувати прогрес моделі та діагностувати будь-які проблеми, які можуть виникнути. Це можна зробити за допомогою бібліотеки журналу, такої як TensorBoard, або шляхом запису повідомлень журналу у файл. Рекомендовано регулярно зберігати контрольні точки моделі під час навчання. Це дозволяє відновити тренування з попереднього пункту, якщо процес навчання перервано або якщо хочете експериментувати з різними конфігураціями навчання. Візуалізація продуктивності моделі та прогресу навчання може бути корисною для розуміння того, як поводиться модель, і виявлення потенційних проблем. Це можна зробити за допомогою бібліотеки побудови графіків, такої як Matplotlib, або шляхом створення графіків за допомогою бібліотеки журналу, такої як TensorBoard. Керування різноманітними файлами, створеними під час процесу навчання, такими як контрольні точки моделі, файли журналу та графіки, може бути складним завданням. Важливо мати чітку та організовану файлову структуру, щоб відстежувати ці файли та полегшувати доступ до них та аналізувати їх за потреби. Ефективне ведення журналів, встановлення контрольних точок, побудова графіків і керування файлами можуть допомогти вам відстежувати прогрес моделі та полегшити налагодження та оптимізацію процесу навчання. Повний список доступних аргументів, що можуть зандобитись для конфігурації режиму навчання наведено в [28]. Формат набору даних для сегментації YOLO можна знайти детально в [29]. Щоб конвертувати свій існуючий набір даних з інших форматів (наприклад, COCO тощо) у формат YOLO, будь ласка, використовуйте інструмент JSON2YOLO від к. Ultralytics. Під час валідації аргументи передавати не потрібно, оскільки model зберігає data та аргументи навчання як атрибути моделі. Кінцевою метою навчання моделі є її розгортання для реальних програм. Режим експорту в к. Ultralytics

YOLOv8 пропонує широкий спектр опцій для експорту навченої моделі в різні формати, що робить її придатною для розгортання на різних платформах і пристроях (рис. 2.2), що дозволяє досягти максимальної сумісності та продуктивності.

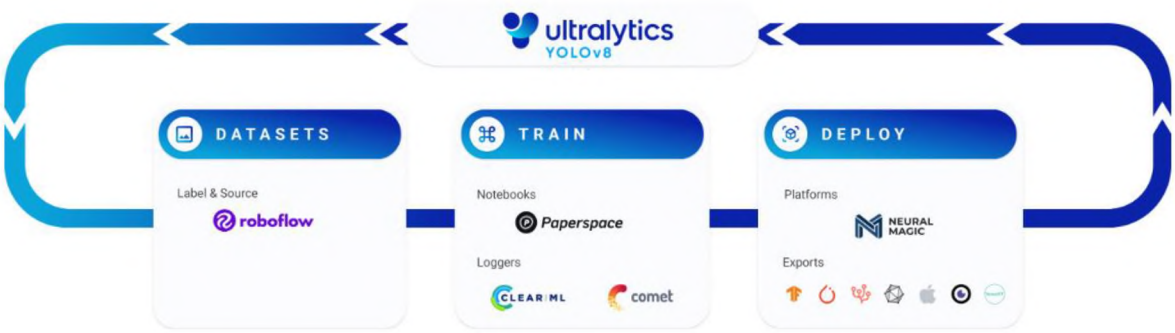


Рисунок 2.2 – Експорт моделі за допомогою к. Ultralytics YOLO

Режим експорту YOLOv8 варто вибирати з кількох причин:

– універсальність – експорт у різні формати, включаючи ONNX, TensorRT, CoreML;

– продуктивність – максимально можна отримати 5-кратне прискорення GPU за допомогою TensorRT і 3-кратне прискорення CPU за допомогою ONNX або OpenVINO;

– сумісність – власна модель користувача буде універсальною для розгортання в багатьох апаратних і програмних середовищах.

– простота використання – наявний простий CLI та API Python для швидкого та простого експорту моделі.

Для експорту існують прості команди в різні формати. При цьому можна експортувати моделі з можливістю пакетного виводу. Експортовані моделі оптимізовано для швидшого часу виводу.

Параметри експорту для моделей YOLO стосуються різних конфігурацій і параметрів, які використовуються для збереження або експорту моделі для використання в інших середовищах або платформах. Ці параметри можуть впливати на продуктивність, розмір і сумісність моделі з

різними системами. Деякі поширені параметри експорту YOLO включають формат експортованого файлу моделі (наприклад, ONNX, TensorFlow SavedModel), пристрій, на якому модель буде запущено (наприклад, CPU, GPU), і наявність додаткових функцій, таких як маски або кілька міток на коробка. Інші фактори, які можуть вплинути на процес експорту, включають конкретне завдання, для якого використовується модель, і вимоги чи обмеження цільового середовища чи платформи. Важливо ретельно розглянути та налаштувати ці параметри, щоб гарантувати, що експортована модель оптимізована для запланованого використання та може ефективно використовуватися в цільовому середовищі.

2.2 Модель Segment Anything Model

Модель Segment Anything Model (SAM) привернула значну увагу завдяки своїй високій продуктивності передачі з Zero-Shot [30] і високій універсальності для багатьох програм зору (наприклад, редагування зображень із дрібним контролем). Zero-Shot Learning (ZSL) – це метод, при якому модель навчається розпізнавати нові класи або патерни, яких не було в навчальному наборі даних. Це особливо корисно у завданнях класифікації, де може бути багато різних класів об'єктів та зібрати достатню кількість прикладів для кожного класу складно чи дорого. Багато з програм сегментації потрібно запускати на кінцевих пристроях з обмеженими ресурсами, наприклад мобільних телефонах, бортових комп’ютерах.

Основною метою проекту Segment Anything стало зниження потреби в експертизі моделювання під конкретні завдання, обчислювальні ресурси для навчання та анотування власних даних з метою сегментації зображень. Для реалізації цього бачення вирішувалось завдання створення базисної моделі для сегментації зображень: модель з можливістю введення промптів, навченої різноманітних даних і здатної адаптуватися під конкретні завдання

аналогічно тому, як промпти використовують у моделях обробки природної мови. Однак дані сегментації, необхідні для навчання такої моделі, на відміну від зображень, відео та текстів, не були доступні онлайн або деінде. Тому в процесі роботи над Segment Anything реалізовувалась паралельно розробити модель сегментації загального призначення, що керується промптами, і використовувати її для створення датасета сегментації безпрецедентного масштабу. SAM навчили загального розумінню, що таке об'єкти; вона здатна генерувати маски для будь-якого об'єкта на будь-якому зображенні та у будь-якому відео, у тому числі для об'єктів та типів зображень, які не зустрічалися їй під час навчання. SAM досить узагальнена, щоб покривати широкий спектр областей застосування і може використовуватися без необхідності додаткового навчання (цю властивість часто називають ZSL) у вихідному вигляді в нових «предметних областях» зображень, підводні знімки або дослідження клітин під мікроскопом. У майбутньому SAM можна буде використовувати для розширення можливостей додатків у різних областях, які потребують пошуку та сегментації будь-якого об'єкта на будь-якому зображенні. Для спільноти дослідників AI SAM може стати компонентом великих AI-систем з метою вироблення більш загального мультимодального розуміння світу, наприклад, для розпізнавання одночасно візуального та текстового вмісту вебсторінки. У сфері AR/VR модель SAM дозволить вибирати об'єкт на основі напряму погляду користувача та «перетворювати» його на 3D. Для творців контенту SAM може розширити можливості творчих завдань, наприклад вилучення областей зображень для колажів або монтажу відео. Також SAM можна використовувати в наукових дослідженнях природних явищ на Землі і навіть у космосі, наприклад, з метою виявлення у відео тварин або об'єктів для вивчення та трасування. Можливості моделі широкі і з радістю дізнаватимемося нові способи її застосування, про які поки що навіть не здогадуємося.

Архітектура Segment Anything на основі промптів забезпечує гнучку інтеграцію з іншими системами. SAM може отримувати вхідні промпти,

наприклад, напрям погляду користувача зі шолома AR/VR. У минулому, для вирішення будь-яких видів завдань сегментації існувало два класи методик. Перша (інтерактивна сегментація) дозволяла сегментувати будь-який клас об'єктів, але вимагала втручання користувача, що ітеративно вказує маску. Друга (автоматична сегментація) дозволяла сегментувати конкретні задані заздалегідь категорії об'єктів (наприклад, котів або стільці), проте вимагала суттєвих обсягів розмічених вручну об'єктів для навчання (наприклад, тисяч або навіть десятків тисяч прикладів сегментованих котів), а також обчислювальних ресурсів та технічної експертизи для навчання моделі сегментації Жодна з цих двох методик не забезпечувала узагальненого, повністю автоматичного підходу до сегментації. SAM стала об'єднанням цих двох класів методик. Це єдина модель, здатна з легкістю виконувати як інтерактивну, і автоматичну сегментацію. Керований промптами інтерфейс моделі можна гнучко використовувати у широкому спектрі задач сегментації завдяки простому створенню потрібного промпту моделі (клацань миші, прямокутників, тексту тощо). Крім того, SAM навчена на різнобічному високоякісному датасеті з більш ніж одного мільярда масок (зібраних у рамках цього проекту), що дозволяє узагальнювати модель до нових типів об'єктів та зображень, які не спостерігалися під час навчання. Така здатність до узагальнення означає, що фахівцям більше не знадобиться збирати власні дані сегментації та виконувати тонке налаштування моделі під свій спосіб застосування. Загалом ці можливості дозволяють узагальнювати SAM на нові завдання та нові предметні області. Такої гнучкості у сфері сегментації зображень вдалося досягти вперше.

SAM дозволяє користувачам сегментувати об'єкти простим клацанням миші або інтерактивно клацаючи по точках, що включаються в об'єкт і виключаються з нього. Також як промпт моделі може використовуватися обмежуючий прямокутник. У разі виникнення невизначеності щодо сегментованого об'єкта SAM може видавати кілька валідних масок: це важлива і необхідна здатність вирішити задачу сегментації в реальному світі.

SAM здатна автоматично знаходити всі об'єкти на зображенні та створювати їх маски. SAM може генерувати в реальному часі маску сегментації для будь-якого промпту після попереднього обчислення ембеддингу зображення, що дозволяє взаємодіяти з моделлю в реальному часі.

У обробці природної мови, а з недавнього часу – і в CV одним із найцікавіших напрямків розвитку є базисні моделі, які можуть виконувати навчання zero-shot та few-shot для нових датасетів та завдань за допомогою техніки «промптингу».

Розробники навчили SAM повертати валідну маску сегментації для будь-якого промпта, а промпт може бути точками фону/переднього плану, приблизний прямокутник або маска, довільний текст або, в загальному випадку, будь-яка інформація, що повідомляє, що потрібно сегментувати на зображенні. Вимога повернення валідної маски просто означає, що навіть коли промпт неоднозначний і може ставитись до кількох об'єктів (наприклад, точка на сорочці може означати або сорочку, або людину в ній), результатом має бути бінарна маска для одного з цих об'єктів. Це завдання використовується для попереднього навчання моделі та для вирішення подальших загальних задач сегментації за допомогою промптингу.

За нашими спостереженнями, завдання попереднього навчання та інтерактивний збір даних накладають певні обмеження на архітектуру моделі. Зокрема, для ефективного анотування модель має виконуватися в реальному часі на CPU у веббраузері, щоб анотатори могли інтерактивно використовувати SAM. Хоча обмеження на середовище виконання передбачає компроміс між якістю та середовищем виконання, на практиці ця проста архітектура дає хороші результати.

Всередині системи кодувальник зображень створює одноразовий ембеддинг для зображення, а легковажний кодувальник в реальному часі перетворює будь-який промпт вектор ембеддинга. Ці два джерела інформації комбінуються в легковажному декодери, що прогнозує маски сегментації. Після обчислення ембеддингу зображення SAM може створити у веббраузері

сегмент для будь-якого промпту лише за 50 мс (рис. 2.3). У веббраузері SAM, по суті, перетворює ознаки зображення та безліч ембеддингу промпту для створення маски сегментації. Для навчання SAM потрібне було величезне і різноманітне джерело даних, якого на початку нашої роботи не існувало. Випущений нами датасет сегментації на сьогодні є найбільшим. Дані були зібрані за допомогою SAM. Зокрема, розмітники використовували SAM для інтерактивної розмітки, а потім розмічені ними дані, у свою чергу, використовувалися для оновлення SAM. Автори повторювали цей цикл багато разів, щоб ітеративно вдосконалювати модель і датасет.

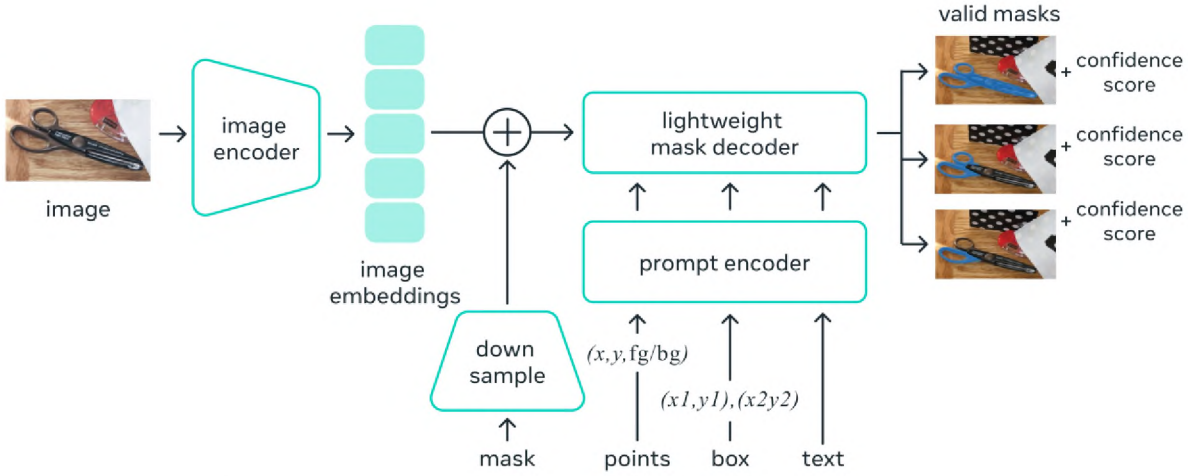


Рисунок 2.3 – Сутність роботи SAM по промпту

Завдяки SAM збирання нових масок сегментації став швидким, як ніколи. При використанні нашого інструменту для інтерактивного анотування маски потрібно приблизно 14 секунд. Процес анотування масок всього в два рази повільніше, ніж анотування прямокутників, що обмежують, яке при використанні найшвидших інструментів анотування займає приблизно 7 секунд. Якщо порівнювати з попередніми великомасштабними проектами збору даних сегментації, наша модель у 6,5 разів швидше, ніж повністю ручне анотування масок з багатокутників у COCO і вдвічі швидше, ніж попередній найбільший проект анотування даних, в якому також використовувалася допомога моделі.

Однак застосування інтерактивного анотування масок не забезпечить достатнього масштабування для створення датасета з одного мільярда масок. Тому для створення датасета SA-1B використаний двигун обробки даних. Цей двигун складається з 3-ох «передач». На першій передачі модель допомагає інструкціям, як це описано вище. Друга передача – це суміш повністю автоматичного анотування та анотування за допомогою моделі, що дозволяє збільшити різноманітність масок, що збираються. Остання передача двигуна обробки даних – це повністю автоматичне створення даних, що дозволяє масштабувати датасет. Готовий датасет містить більше 1,1 мільярда масок сегментації, зібраних приблизно з 11 мільйонів ліцензованих і зберігають конфіденційність зображень (рис. 2.4). SA-1B (рис. 2.5) містить у чотириста разів більше масок, ніж будь-який інший датасет сегментації; ручна перевірка показала, що маски мають високу якість і різноманітність, а в деяких випадках навіть можна порівняти за якістю з масками з попередніх, набагато менших датасетів, повністю розмічених вручну.

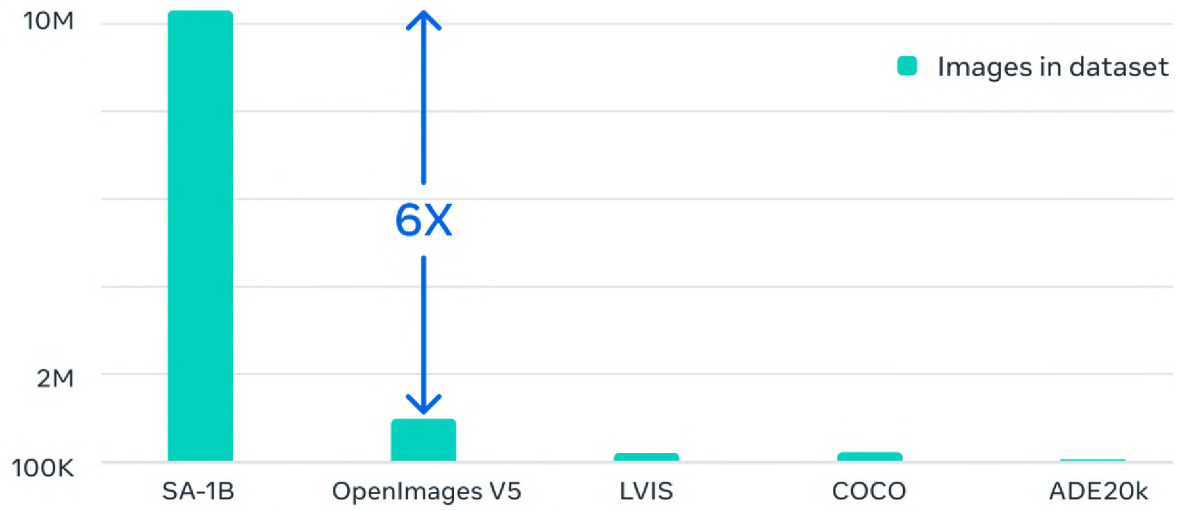


Рисунок 2.4 – Порівняння розмірів датасетів

Можливості Segment Anything стали результатом навчання на мільйонах зображень та масок, зібраних за допомогою двигуна обробки даних. У результаті отримано датасет із понад мільярда масок сегментації, що в чотириста разів більше, ніж найбільший датасет сегментації до нього.

Зображення для SA-1B бралися з джерела фотографій з різних країн, щоб забезпечити безліч географічних регіонів і рівнів доходу. Хоча деякі географічні регіони все ще представлені недостатньо широко, SA-1B містить більше і різноманітність зображень, ніж у будь-якому іншому датасеті сегментації до нього.

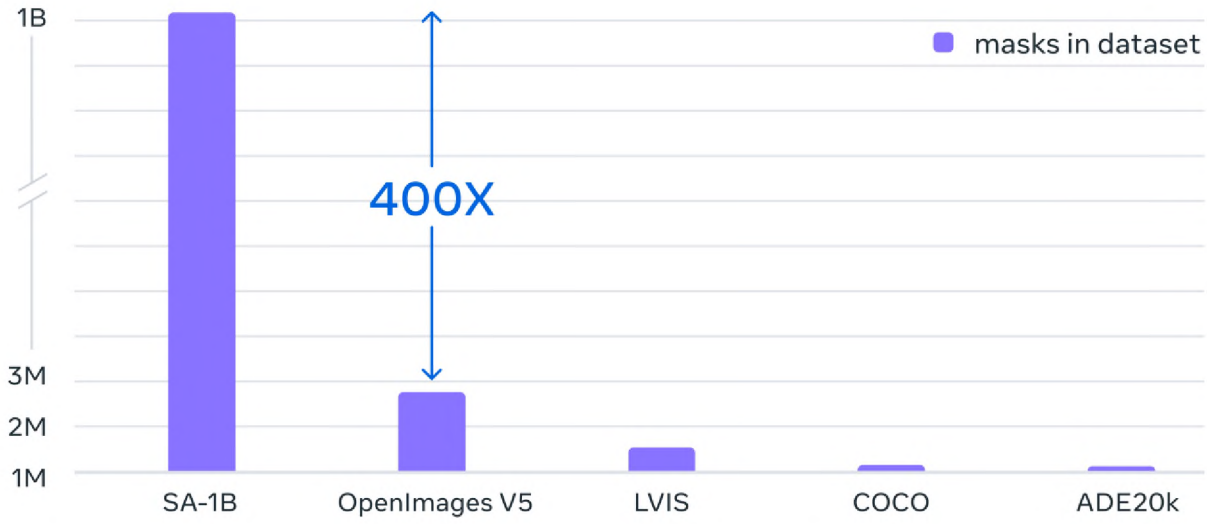


Рисунок 2.5 – Порівняння кількості маско у датасетах

Можливі перекоси даної моделі за гендерним співвідношенням, кольором шкіри та віковим діапазоном людей, і з'ясували, що точність результатів SAM для різних груп схожа. це зробить роботу з моделлю більш корисною для застосування у реальному світі.

Завдяки датасету SA-1B стало можливим наше дослідження, але він може допомогти іншим дослідникам навчати базисні моделі для сегментації зображень. Ці дані зможуть стати фундаментом для нових датасетів з додатковими анотаціями, наприклад, з текстовим описом, який відповідає кожній масці.

У майбутньому SAM можна буде використовувати для розпізнавання повсякденних предметів через AR-окуляри, які допомагають користувачам нагадуваннями та підказками (рис. 2.6). SAM може вплинути на широкий спектр предметних областей може допомогти фермерам у господарюванні або біологам у їхніх дослідженнях (рис. 2.7). Наведені результати досліджень

та датасет можуть прискорити вивчення сегментації та ширшого розуміння зображень та відео.

Рисунок 2.6 – Спільне використання SAM і VR/AR

Керована промптами модель сегментації може виконувати завдання сегментації, працюючи компонентом більшої системи. Можливість комбінування -– це потужний інструмент, що дозволяє використовувати одну модель різними способами, можливо навіть для вирішення завдань, невідомих на момент проектування моделі.

Рисунок 2.7 – Варіанти застосування SAM

Компонентна архітектура системи, яка стала можливою завдяки таким технікам, як prompt engineering, дозволить застосовувати її в ширшому спектрі областей, ніж системи, які навчаються на виконання конкретного набору завдань, і що SAM зможе стати потужним компонентом у таких сферах, як AR/VR, створення контенту, наука та системи AI загального призначення. У майбутньому зв'язок між розумінням зображень на рівні

пікселів і більш високорівневим семантичним розумінням візуального контенту стане тісніше, що дозволить створювати ще потужніші системи AI.

З моменту появи SAM, з'явилося багато проектів і документів, спрямованих на дослідження SAM з різних точок зору. Враховуючи те, що SAM стверджує, що може сегментувати будь-що, низка робіт повідомила про його недостатню ефективність у реальних ситуаціях, зокрема медичних зображеннях [31], закамуфльованих об'єктів [32] та прозорих об'єктів [33]. Тобто SAM добре працює в загальних налаштуваннях, але не в вищезазначених складних налаштуваннях.

Інший важливий напрямок досліджень зосереджено на вдосконаленні SAM для підвищення його практичності. В роботі [34] показано, що вихідними масками SAM можна легко маніпулювати за допомогою примусових атак за допомогою зловмисно створених примусових збурень.

Робота [35] – далі проводить всебічну оцінку надійності SAM, починаючи від передачі стилю та загальних пошкоджень до локальної оклюзії та змагального збурення.

Інший напрям роботи зосереджений на демонстрації універсальності SAM. В роботі [36] розглянуто комбінацію Grounding DINO з SAM для сегментації будь-чого за допомогою введення тексту. Зокрема, він покладається на Grounding DINO для створення обмежувальної рамки з тексту, а потім згенеровану рамку можна використовувати як підказку для сегментування маски. SAM передбачає маски без міток і роботи [37] поєднують SAM з іншими моделями, такими як CLIP [38], щоб семантично сегментувати будь-що.

Крім сегментації об'єктів, багато робіт також показали свою універсальність в інших сферах, включаючи редагування [41].

Окрім двовимірного зору, дослідження SAM також було розширено до 3-вимірної реконструкції об'єктів [42], демонструючи його можливості щодо допомоги у створенні 3D-моделі з одного зображення.

2.3 Модель Fast Segment Anything Model

Модель Fast Segment Anything Model (FastSAM) [43] – це рішення на базі CNN, призначене для вирішення задачі сегментації об'єктів у реальному часі. Це завдання розроблено для сегментації будь-якого об'єкта на зображенні на основі різних можливих запитів користувача. FastSAM значно знижує обчислювальні вимоги, при цьому зберігаючи конкурентоспроможність роботи, що робить її практично придатною для різних завдань CV.

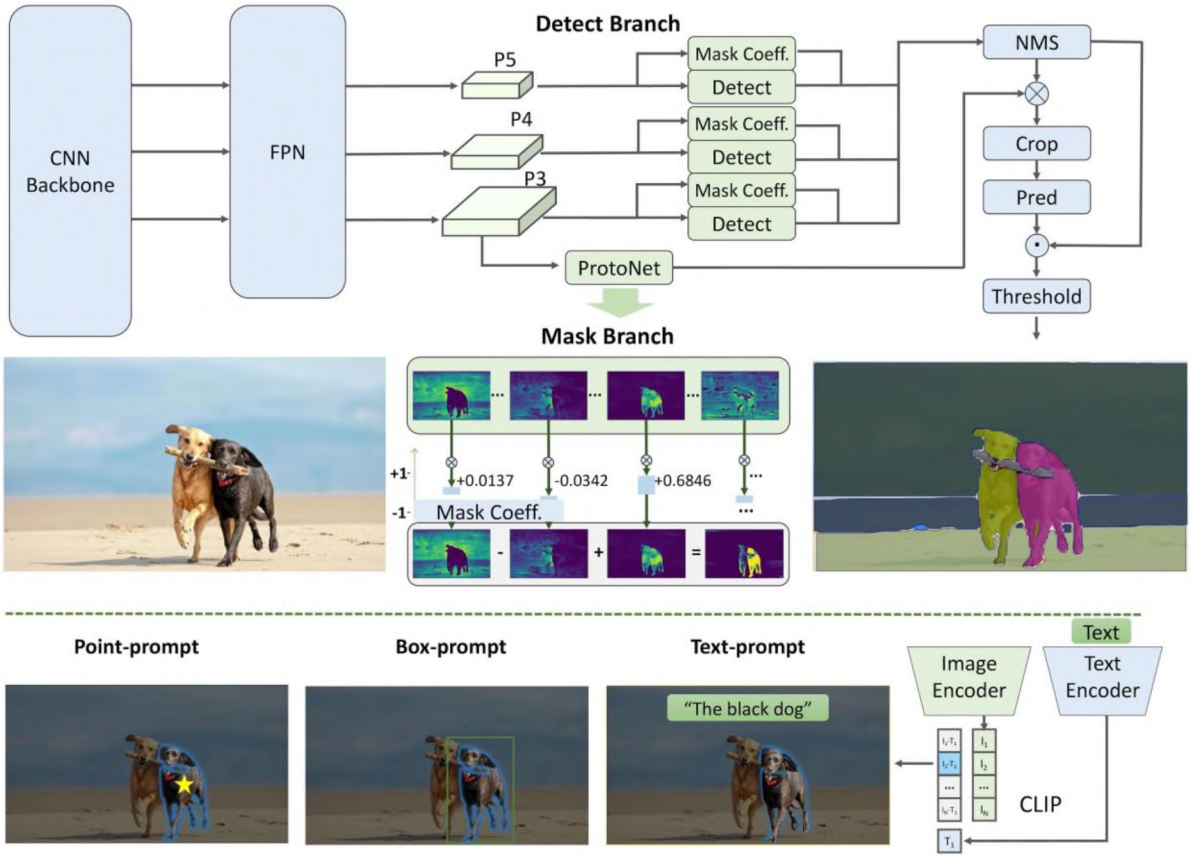


Рисунок 2.8 – Модель FastSAM

FastSAM розроблена для подолання обмежень моделі Segment Anything (SAM), важкої моделі Transformer, яка потребує значних обчислювальних ресурсів. FastSAM поділяє завдання сегментації об'єктів на два послідовні етапи: сегментація всіх екземплярів та вибір, що базується на запитах

користувача. На першому етапі використовується YOLOV8-seg для створення сегментаційних масок всіх екземплярів на зображенні. На другому етапі модель виводить область інтересу, відповідну запиту.

1. Рішення в реальному часі. Завдяки ефективності обчислень на базі CNN, FastSAM забезпечує вирішення задачі сегментації об'єктів у реальному часі, що робить її цінною для застосування у промислових додатках, що потребують швидких результатів.

2. Ефективність та продуктивність. FastSAM забезпечує значне зниження обчислювальних та ресурсних вимог, не погіршуючи якість роботи. Вона досягає порівнянної продуктивності з моделлю SAM, але вимагає значно менше обчислювальних ресурсів, що дозволяє використовувати її в реальному часі.

3. Сегментація на основі запитів користувача. FastSAM може виконувати сегментацію будь-якого об'єкта на зображенні, ґрунтуючись на різних можливих запитах користувача, що забезпечує гнучкість та пристосованість до різних сценаріїв.

4. Заснована на YOLOV8-seg. FastSAM заснована на моделі YOLOV8-seg, яка є детектором об'єктів з гілкою сегментації екземплярів. Це дозволяє ефективно створювати сегментаційні маски всіх екземплярів на зображенні.

5. Високі результати на показниках: Під час виконання завдання пропозиції об'єктів на наборі даних MS COCO FastSAM досягає високих показників продуктивності при значно більшій швидкості роботи, ніж SAM на одному графічному процесорі NVIDIA RTX 3090, що свідчить про її ефективність та здатність.

6. Практичне застосування. Запропонований підхід надає нове практичне рішення для великої кількості завдань комп'ютерного зору з дуже високою швидкістю, яка в десятки або сотні разів перевищує швидкість існуючих методів.

7. Можливість стиснення моделі. FastSAM демонструє можливість суттєво знизити обчислювальні витрати, ввівши штучну перевагу в структуру

моделі, відкриваючи нові можливості створення великомасштабних архітектур моделей для загальних завдань CV. На даний час, доступні такі моделі: FastSAM-s і FastSAM-x. Вони містять конкретні заздалегідь навчені вагі та підтримують завдання та сумісність з різними режимами роботи, наприклад, такими як навчання, валідація, вивід, експорт. Моделі FastSAM легко інтегрувати у ваші програми на Python. Ultralytics надає зручний інтерфейс користувача API і команди CLI для спрощення розробки. Слід зверніть увагу, що FastSAM підтримує лише виявлення та сегментацію єдиного класу об'єктів. Це означає, що модель розпізнаватиме і сегментуватиме всі об'єкти як один і той же клас. Тому, під час підготовки набору даних потрібно перетворити всі ідентифікатори категорій об'єктів у 0.

2.4 Модель MobileSAM

Надалі доцільно розглянути ще один полегшений варіант SAM – MobileSAM [44]. Щоб зробити SAM зручним для мобільних пристроїв, потрібно замінити важкий кодер зображень на легкий. Однак класичний спосіб навчання такого нового SAM, як в оригінальній документації для SAM, призводить до незадовільної продуктивності, особливо коли доступні обмежені джерела навчання. Головним чином це спричинено спільною оптимізацією кодера зображення та декодера маски у випадку відокремленої дистиляції. Тому доцільно розподіляти знання з важкого кодувальника зображень (ViT-H в оригінальному SAM) до легкого кодувальника зображень, який може бути автоматично сумісний із декодером маски в оригінальному SAM. Навчання можна завершити на одному GPU менш ніж за один день, і отриманий полегшений SAM – MobileSAM, який більш ніж у 60 разів менший, але працює на рівні з оригінальним SAM. Що стосується швидкості виводу, з одним GPU MobileSAM працює близько 10 мс на зображення: 8 мс на кодері зображення та 4 мс на декодері маски. Завдяки

гарній продуктивності наш MobileSAM приблизно в 5 разів швидший за FastSAM і в 7 разів менший, що робить його більш придатним для мобільних додатків. MobileSAM може працювати відносно гладко на CPU. Код для проекту надається на сайті MobileSAM, де демонстрація показує, що MobileSAM може відносно безперешкодно працювати на CPU. SAM складається з двох компонентів: кодувальника зображень на основі ViT і декодера маски з підказками, які працюють послідовно (рис. 2.9).

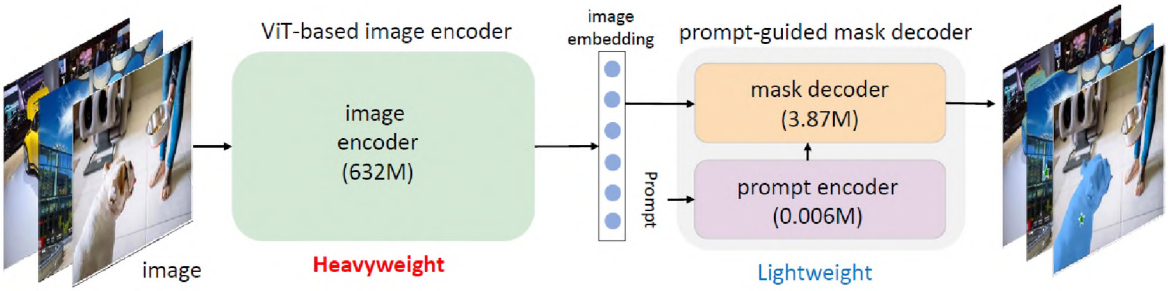


Рисунок 2.9 – Узагальнена структура моделі SAM

З моменту появи SAM привернув значну увагу з багатьох причин. По-перше, це перший випадок, що CV може слідувати за NLP шляхом, який поєднує базову модель із швидким проектуванням. По-друге, це перша, яка виконує сегментацію без міток, фундаментальне завдання CV, яке є паралельним прогнозу міток. Крім того, це фундаментальне завдання робить SAM сумісним з іншими моделями для реалізації розширених додатків CV, таких як сегментація з керуванням текстом і редагування зображень із детальним керуванням. Однак багато з таких випадків використання потрібно запускати на кінцевих пристроях з обмеженими ресурсами. Як показано в офіційній демонстрації, із вбудованою обробкою зображення SAM може працювати на пристроях з обмеженими ресурсами, оскільки декодер маски легкий. Великий кодер зображень робить обчислення конвеєра SAM важким. Але можна отримати легкий SAM, придатний для мобільних пристроїв з обмеженими ресурсами, який тому називається MobileSAM. Параметри SAM з різними варіантами кодера зображень наведені в табл. 2.2. Згідно [45],

навчання нового SAM з ViT-L або ViT-B як кодувальником зображень вимагає 128 GPU протягом кількох днів. Така ресурсомістка перепідготовка може бути нетривіальним тягарем для відтворення або покращення їх результатів.

Таблиця 2.2 – Параметри SAM з різними кодерами зображення.

Параметр	SAM (ViT-H)	SAM (ViT-L)	SAM (ViT-B)
ViT-based encoder	632M	307M	86M
Prompt-guided encoder	0.006M	0.006M	0.006M

Ця проблема оптимізації, в основному, виникає через поєднану оптимізацію кодера зображення та декодера маски. Керуючись цим розумінням, пропонується роз'єднати оптимізацію кодера зображення та декодера маски. Якщо говорити конкретно, то спочатку передаємо знання від стандартного кодувальника зображень ViT-H до малого ViT. Після цього можемо точно налаштувати декодер маски в оригінальному SAM для кращого узгодження з кодувальником дистильованого зображення. Варто підкреслити, що оптимізація вирівнювання є необов'язковою, оскільки той факт, що полегшений кодер зображень виведено з кодувальника зображень за замовчуванням, гарантує його притаманне вирівнювання з декодером маски за замовчуванням.

Ранні додатки для мобільного CV в основному базувалися на легких CNN. Основна ідея MobileNet полягає в розділенні звичайного блоку згортки на згортку по глибині та поточкову згортку, що значно скорочує параметри режиму та час обчислення. З появою [46], численні роботи намагалися зробити його легким і ефективним, наприклад: [47] містить ViT-Huge (ViT-H), ViT-Large (ViT-L), ViT-Base (ViT-B). Менші ViT представлені в [48] і позначаються як Deit-Small (Deit-S) і Deit-Tiny (Deit-T) ViT-Small і ViT-Tiny. В [49] представлено MobileViT – поєднання ViT зі стандартними згортками для покращення продуктивності, яка перевершує MobileNetv2. Основна мотивація полягає в тому, щоб використати можливості місцевого

представлення CNN, і ця практика супроводжується численними подальшими роботами, спрямованими на підвищення швидкості моделі, включаючи EfficientFormer, EfficientViT, Next-ViT і Tiny-ViT. Нещодавній прогрес у легкій і швидшій системі ViT доповнює відокремлену дистиляцію, щоб зробити SAM придатним для кінцевих (мобільних) пристроїв з обмеженими ресурсами.

З огляду на те, що кодувальник зображень за замовчуванням у SAM базується на ViT-H, простий спосіб отримати Mobile SAM – це слідувати офіційному каналу в [45] для перенавчання нового SAM із меншим кодувальником зображень, наприклад заміна ViT-H на менший ViT-L або навіть менший ViT-B. Перетворюючи проблему пошуку нового конвеєра SAM на відокремлену дистиляцію, підхід має перевагу в тому, що він простий і ефективний, а також відтворюваний за низькою ціною (на одному GPU менше ніж за день). Декодер маски з підказками в оригінальному SAM має менше 4M параметрів і тому вважається легким. Враховуючи вбудоване зображення, оброблене кодувальником, SAM може працювати в пристроях з обмеженими ресурсами, оскільки декодер маски легкий. Однак кодер зображень за замовчуванням в оригінальному SAM базується на ViT-H із понад 600 M параметрами, що є дуже важким і робить увесь конвеєр SAM несумісним із мобільними пристроями. Таким чином, ключ до отримання зручного для мобільних пристроїв SAM полягає в заміні важкого кодувальника зображень на полегшений, який також автоматично зберігає всі свої функції та характеристики оригінального SAM. Нижче ми детально розглянемо запропонований нами метод досягнення мети цього проекту.

Простий спосіб реалізації ідеї для перенавчання нового SAM із меншим кодувальником зображень згідно [45] навчання SAM із кодувальником зображень ViT-H займає 68 годин на 256 GPU A100. Заміна ViT-H на ViT-L або ViT-B зменшує кількість необхідних GPU до 128, що все ще є недосяжні за можливостями для багатьох дослідників у спільноті, щоб відтворити або покращити свої результати. Маски в наборі даних 11-T надаються попередньо

підготовленим SAM (з кодувальником зображень ViT). По суті, цей процес перепідготовки є дистиляцією знань [50], який передає знання від SAM на основі ViT-H до SAM із меншим кодувальником зображення (рис. 2.10).

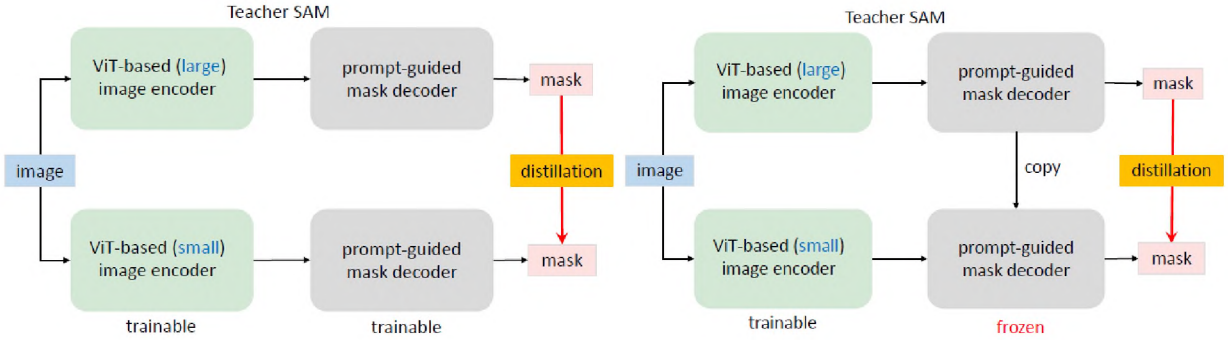


Рисунок 2.10 – Відмінності SAM і MobileSAM

При переході від оригінального SAM до варіанту з меншим кодувальником зображення, труднощі в основному полягають у поєднаній оптимізації кодера зображення та комбінованого декодера. Інтуїтивно зрозуміло, що оптимізація кодера зображення залежить від якості декодера зображення, і навпаки. Коли обидва модулі в SAM перебувають у поганому стані, складніше навчити їх обидва до нормального стану. Як наслідок, використовуючи алгоритм [51] можна розділити завдання на два підзавдання: дистиляція кодувальника зображень і точне налаштування декодера маски. Оскільки декодер маски в оригінальному SAM вже легкий, можна використовувати його архітектуру. Це надає перевагу легкого використання комбінованого декодера для точного налаштування замість навчання його з нуля. Щоб полегшити проблему оптимізації пов'язаної дистиляції, простим способом є оптимізація кодера зображення за допомогою скопійованого та замороженого декодера маски (див. рис. 2.10). Операція заморожування може допомогти запобігти погіршенню якості декодера маски через поганий кодер зображення. Тобто використовується напівзв'язана дистиляція, оскільки оптимізація кодера зображення ще не повністю відокремлена від декодера маски. Емпірично виявлено, що ця оптимізація все ще є складною, оскільки вибір підказки є випадковим, що робить декодер маски змінним і, таким

чином, збільшує складність оптимізації. Таким чином, пропонується дистилювати кодувальник малих зображень безпосередньо з ViT-H в оригінальному SAM, не вдаючись до комбінованого декодера, який називається розв'язаною дистиляцією (рис. 2.11). Ще одна перевага виконання дистиляції при вбудовуванні зображення полягає в тому, що можна прийняти просту втрату MSE замість використання комбінації фокальних втрат для прогнозування маски, як у [45].

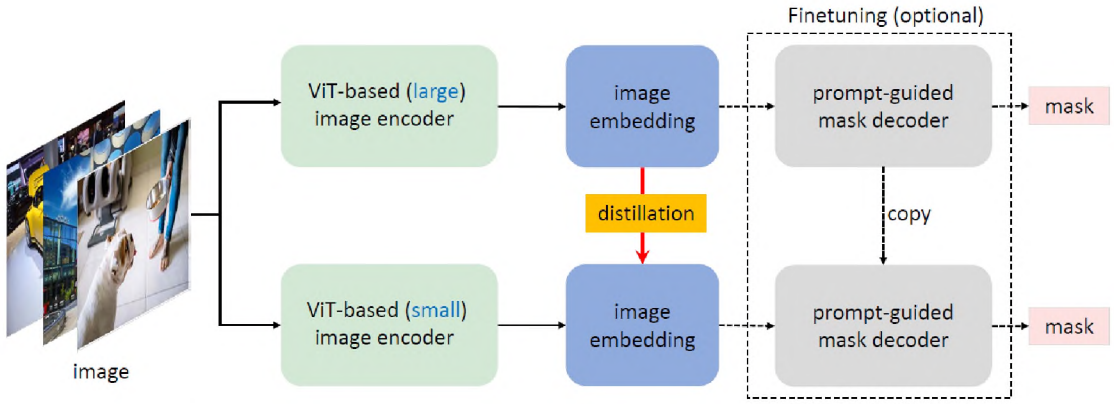


Рисунок 2.11 – Відокремлена дистиляція для SAM

На відміну від напівзв'язаної дистиляції, описана вище роз'єднана дистиляція дає легкий кодер зображення. Очікується, що точне налаштування декодера маски на кодувальнику заморожених легких зображень або спільне їх точне налаштування може додатково підвищити продуктивність.

2.5 Синтезована модель виявлення об'єктів на основі нейронної мережі

На основі проведених досліджень можна реалізувати модель виявлення об'єктів на основі нейронної мережі, що дозволить побудувати систему виявлення перешкод під час руху UAV. Вона складається з 3-ох основних

етапів, показаних на рис. 2.12. На першому етапі система збирає зображення або відеодані через камеру UAV та застосовує етапи попередньої обробки, такі як стабілізація, зменшення шуму та покращення, щоб покращити зображення, якість і зручність використання.

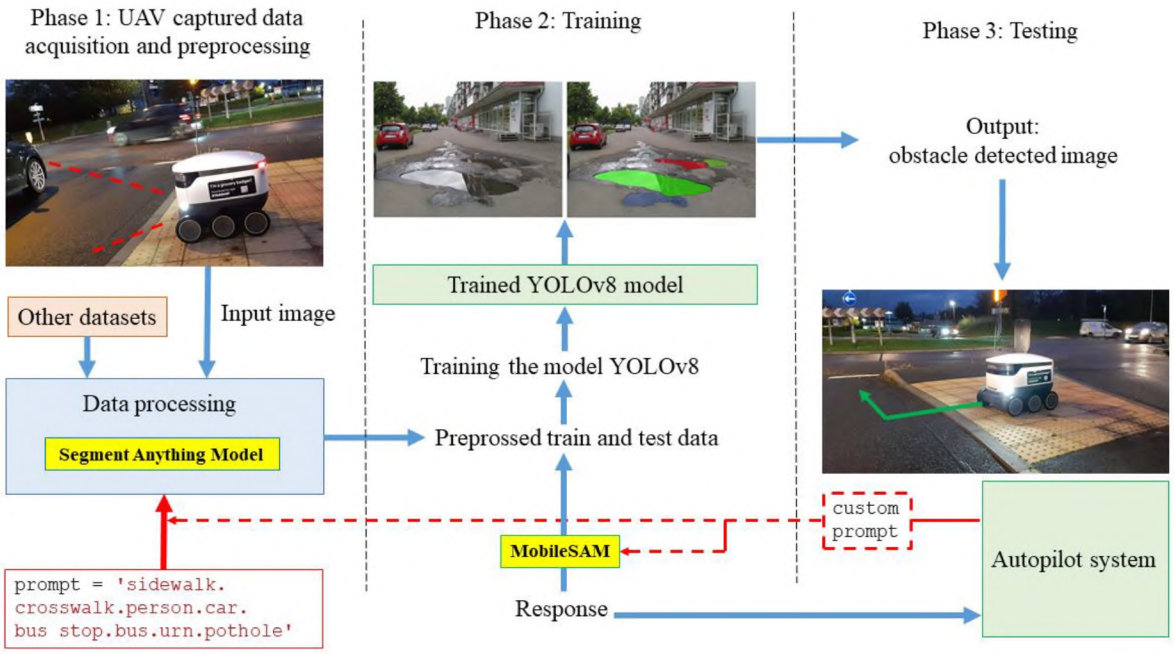


Рисунок 2.12 – Модель виявлення об'єктів на основі нейронної мережі

На другому етапі попередньо оброблені дані обробляються для створення наборів даних для навчання та тестування. Щоб забезпечити достатньо різноманітний і репрезентативний набір даних, цей крок охоплює дії, включаючи створення обмежувальної рамки, анотації об’єктів і підходи до розширення даних. При цьому необхідно вирішувати завдання сегментації (рис. 2.13). Потім запропонована модель на основі YOLOv8 навчається за допомогою підготовленого набору даних. Нарешті, на третьому етапі навчена модель YOLOv8 розгортається та тестується для виявлення об’єктів у реальному часі за допомогою UAV. Модель обробляє пряме відео з UAV і прогнозує наявність і розташування цікавих об’єктів у реальному часі. Ця фаза дозволяє виявляти об’єкти під час роботи UAV, дозволяючи негайно реагувати або приймати рішення на основі виявлених об’єктів у реальному

часі. Якщо виникають певні ситуації суперечливого, незрозумілого виявлення об'єкту система автопілоту формує запити до MobileSAM, яка розміщується на борту UAV.

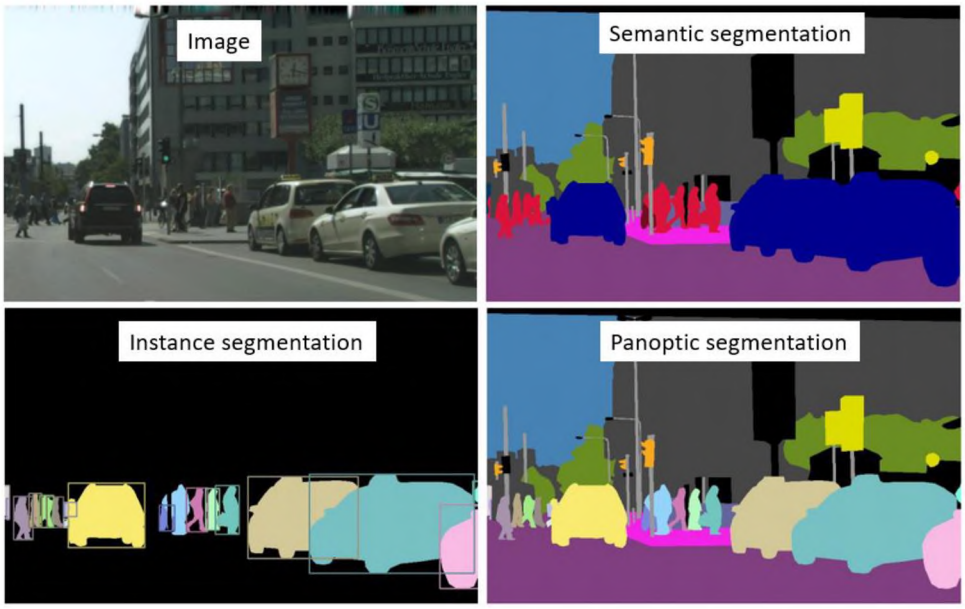


Рисунок 2.13 – Варіанти завдання сегментації

Такий підхід може підвищити якість роботи автопілоту, наприклад, у випадку відсутності каналів комунікації з центром моніторингу або оператором. Загалом, ця структура забезпечує систематичний підхід до збору даних за допомогою UAV, попередньої обробки даних, навчання моделі виявлення об'єктів і виконання виявлення об'єктів у реальному часі за допомогою UAV.

Висновки до розділу 2

Проведено дослідження версій YOLOv8, що орієнтовані на виконання завдань сегментації.

Для SAM складається з кодувальника зображень на основі ViT і декодера масок із підказками. Кодер зображення приймає зображення як

вхідні дані та генерує вбудовування, яке потім передається в декодер маски. Декодер маски генерує маску, щоб вирізати будь-який об’єкт із фону на основі підказки, наприклад точки (або прямокутника). Крім того, SAM дозволяє генерувати кілька масок для одного запиту для вирішення проблеми неоднозначності, що забезпечує цінну гнучкість.

Існують кілька варіантів реалізації SAM, що відрізняються від оригінальної версії скороченням вимог до обчислювальних ресріву, що залучаються на етапі навчання. До таких відносяться FastSAM і MobileSAM. В останній моделі виконана заміна стандартного ViT-H на полегшений кодер зображень. Такий підхід є актуальним для інтеграції проєктів AI+CV для кінцевих пристроїв і систем, наприклад таких, що використовуються в UAV.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО РЕАЛІЗАЦІЇ МОДЕЛІ ВИЯВЛЕННЯ ОБ'ЄКТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Порівняння MobileSAM з оригінальним SAM і FastSAM

Для оцінки продуктивності MobileSAM, ми обчислюємо mIoU між двома масками, згенерованими SAM і MobileSAM в одній точці підказки. Інтуїтивно зрозуміло, що більший mIoU вказує на вищу ефективність прогнозування маски, припускаючи, що маска, створена ViT-H, є фактичною. Для поєднаної дистиляції приймаємо SAM з ViT-B, наданий в оригінальному SAM [45]. Для MobileSAM з роз'єднаною дистиляцією тренується модель на 2 GPU (по два зразки на GPU для економії обчислювальних ресурсів) на 0,1 % зразків зображень набору даних SA-1B для 55000 ітерацій. Відокремлена дистиляція займає менше 1 % обчислювальних ресурсів, ніж сполучена дистиляція, при цьому досягається гарна продуктивність *mIoU*: 0,75 проти 0,72 для об'єднаної дистиляції (усереднене для 200 зразків) – табл. 3.1. Оскільки ViT-B все ще є недосяжним для мобільних пристроїв, тому доцільно використовувати TinyViT (з параметрами 5M) на основі запропонованої розділеної дистиляції.

Таблиця 3.1 – Порівняння поєднаної дистиляції та розділеної дистиляції для SAM з ViT-B як кодувальником зображення.

	SAM (coupled distillation)	SAM (decoupled distillation)
mIoU	0.72	0.75
Training GPUs	128	2
Batch size	128	4
Iterations	180k	55k
Training Data	11M	11K

Роздільна дистиляція працює краще та вимагає менше 1 % обчислювальних ресурсів, ніж сполучена дистиляція. Як і магістраль на

основі ViT, ViT-Tiny має такі ж параметри, як Deit-Tiny, але працює краще. Наприклад, на ImageNet-1K Deit-Tiny досягає точності 72,2 %, тоді як ViT-Tiny досягає 79,1 %. Тому ми використовуємо ViT-Tiny для підтвердження концепції, щоб продемонструвати ефективність запропонованої нами відокремленої дистиляції для навчання легкого MobileSAM, який може бути набагато швидшим, ніж оригінальний SAM. Прийнятий полегшений кодер зображень складається з 4-ох етапів, які поступово зменшують роздільну здатність. Перший етап побудований блоками згортки з оберненими залишками, а решта три ступені складаються з трансформаторних блоків. На початку моделі є 2 блоки згорток із кроком 2 для зменшення роздільної здатності. Операція зменшення дискретизації між різними етапами обробляється блоками згортки з кроком 2. У MobileSAM встановлено крок 2 в останній згортці зменшення дискретизації на 1, щоб остаточна роздільна здатність відповідала такій як у кодувальника зображень ViT-H оригінального SAM. Швидкість визначення параметрів MobileSAM підсумовано в табл. 3.2. Швидкість висновку вимірюється на одному GPU.

Таблиця 3.2 – Порівняння параметрів і швидкості для кодувальника зображень в оригінальному SAM і MobileSAM

Характеристика	Orirginal SAM	MobileSAM
Parameters	632M	5.78M
Speed	452 мс	8 мс

Для відокремленого навчання на кодері зображень полегшений кодер навчається з 1 % набору даних SA-1B [45] протягом 8 епох на одному GPU. Спостерігається, що більше обчислень витрачається на прямий процес на кодері зображень викладача, враховуючи, що він значно важчий, ніж прийнятий полегшений кодер зображень. Щоб зробити дистиляцію швидшою потрібно запустити процес пересилання лише один раз. З одним GPU можна отримати свій варіант MobileSAM менш ніж за день. Очікується, що навчання MobileSAM з більшою кількістю GPU протягом тривалого часу

дасть кращу продуктивність. Початкове дослідження виконання тонкого налаштування декодера маски ще більше покращує продуктивність MobileSAM. Для кількісної оцінки дистильованого SAM обчислюємо mIoU між масками, передбаченими оригінальним SAM і MobileSAM. Для основних результатів використовуються прогнозовані маски з двома типами підказок: точкою та прямокутним. Результати з підказкою «точка» показані на рис. 3.1, а результати з підказкою в полі – на рис. 3.2 [44].

Рисунок 3.1 – Прогноз маски з однією точкою як підказкою [45]

При цьому, спостерігається, що MobileSAM робить задовільний прогноз маски, подібний до вихідного SAM. Абляційне дослідження впливу тренувального обчислення на продуктивність SAM. Результати в табл. 3.3 показують, що за однакової кількості ітерацій збільшення розміру партії підвищує продуктивність моделі. Крім того, за розміром пакету продуктивність також виграє від більшої кількості ітерацій оновлення за рахунок збільшення епох навчання. При цьому, всі експерименти проводяться на одному GPU. збільшення кількості GPU для забезпечення більшого розміру батчу або подальше збільшення ітерацій може додатково підвищити продуктивність.

Рисунок 3.2 – Прогноз маски з полем як підказкою [45]

Роль підказки полягає в тому, щоб вказати, що сегментувати в зображенні. Теоретично будь-який об'єкт можна сегментувати, якщо підказку встановлено належним чином, тому це називається «сегментувати будь-що».

Таблиця 3.3 – Дослідження впливу тренувальних обчислень на продуктивність MobileSAM

batch size	epochs	Iterations	*mIoU*
4	2	50k	0.7057
8	4	50k	0.7286
8	8	100k	0.7447

FastSAM безпосередньо генерує варіант маски за допомогою YOLOv8 без підказок. Щоб увімкнути сегментацію з підказками, розроблено алгоритм відображення, який вибирає маску з наборів масок пропозиції. FastSAM складається з гілки виявлення на основі YOLOv8 і гілки сегментації на основі YOLACT для створення пропозиції маски без підказок. Він має 68 млн параметрів і займає 40 мс для обробки зображення. Навпаки, MobileSAM має менше 10M параметрів, що значно менше. Що стосується швидкості виводу, на одному GPU обробка зображення займає 40 мс, у той час як у MobileSAM лише 10 мс, що в 4 рази швидше, ніж у FastSAM (табл. 3.4).

Таблиця 3.4 – Порівняння FastSAM і MobileSAM

Характеристика	FastSAM	MobileSAM	Співвідношення
Розмір	68M	9.66M	≈ 7
Швидкість	64ms	12ms	≈ 5

Як оригінальний SAM, так і MobileSAM використовують той самий декодер масок, керований підказками (табл. 3.5).

Таблиця 3.5 – Порівняння FastSAM і MobileSAM

Характеристика	Оригінальний SAM	MobileSAM
Розмір	3.876M	3.876M
Швидкість	4 мс	4 мс

Надалі порівнюємо *mIoU* між прогнозованими масками з вихідним SAM. FastSAM пропонується прогнозувати маску з кількома точками, для яких вибираємо одну для переднього плану, а іншу для фону. Результати показують, що mIoU для FastSAM набагато менший, ніж для MobileSAM, що свідчить про те, що передбачення маски FastSAM сильно відрізняються від прогнозу вихідного SAM. Крім того, *mIoU* для FastSAM дуже швидко зменшується, коли відстань між двома точками підказки. Це головним чином викликано тим фактом, що FastSAM часто не може передбачити об'єкт, коли точка підказки переднього плану встановлена занадто близько до точки підказки фону. Результати для завдання «сегментувати все» показано на рис. 3.3. Є два основних спостереження. По-перше, результати MobileSAM добре узгоджуються з результатами оригінального SAM. Навпаки, результати FastSAM часто менш задовільні. Наприклад, FastSAM часто не може передбачити деякі об'єкти, наприклад, дах на першому зображенні. Крім того, пропозицію маски іноді важко інтерпретувати (наприклад, маска для сцени на першому зображенні та для неба на другому зображенні.). По-друге, FastSAM часто створює маски з негладкими межами, тоді як оригінальний SAM і MobileSAM не мають цієї проблеми.

Рисунок 3.3 – Порівняння результатів усіх сегментів [45]

Отриманий MobileSAM зменшує параметри кодера в 100 разів, а загальні параметри ще в 60 разів. При цьому, MobileSAM працює нарівні з оригінальними важкими SAM, що є значним кроком у просуванні SAM для мобільних додатків. Для виводу за допомогою MobileSAM одне зображення займає лише 8 мс на кодувальнику зображень і 4 мс на декодері маски. Варто підкреслити, що MobileSAM приблизно в 5 разів швидший і в 7 разів менший, ніж аналогічний FastSAM, досягаючи гарної продуктивності.

3.2 Підготовка даних для навчання та навчання моделі YOLOv8

Щоб навчити модель, потрібно підготувати анотовані зображення та розділити їх на набори даних для навчання та перевірки. Треба

використовувати навчальний набір для навчання моделі та набір перевірки для перевірки результатів дослідження та вимірювання якості навченої моделі. Можна помістити 80 % зображень у навчальний набір і 20 % у набір перевірки. Щоб спростити процес анотації зображень, існує багато програм, які можна використовувати для візуального коментування зображень для машинного навчання. Існує також багато онлайн-інструментів, які можуть виконувати всю цю роботу, наприклад, Roboflow Annotate [51]. Використовуючи цей сервіс, просто потрібно завантажити зображення, намалювати на них рамки та встановити класи для кожної рамки. Потім інструмент автоматично створить файли анотацій, розділить ваші дані на набори даних для навчання та перевірки та створить файл дескриптора YAML. Потім можна експортувати та завантажити анотовані дані як ZIP-файл. Для реальних ситуацій на дорозі для UAV ця база даних має бути набагато більшою. Щоб навчити хорошу модель, повинні мати сотні чи тисячі анотованих зображень. Крім того, при підготовці бази зображень треба її роботи збалансованою. Датасет повинен мати однакову кількість об'єктів кожного класу. Інакше навчена на ньому модель може передбачити один клас краще, ніж інший. Після того, як дані будуть готові, їх копіювати у папку з кодом Python, який використовується для навчання, і повертаємось до свого блокнота Jupyter, щоб почати процес навчання. Після того, як дані будуть готові, їх потрібно пропустити через модель. Для навчання використовуємо інший спеціальний набір даних, який містить світлофори та дорожні знаки [52]. Цей набір даних дозволяє навчити YOLOv8 виявляти різні об'єкти на дорогах (рис. 3.4). При його завантаженні вибирається формат «YOLOv8». У архіві з Roboflow є додатковий «test» набір даних, який не використовується в процесі навчання. Зображення з нього можна використовувати для додаткової перевірки самостійно після навчання. Архів розпакується до папки з вашим кодом Python і виконайте метод train, щоб запустити навчальний цикл:

```
model.train(data="data.yaml", epochs=30)
```

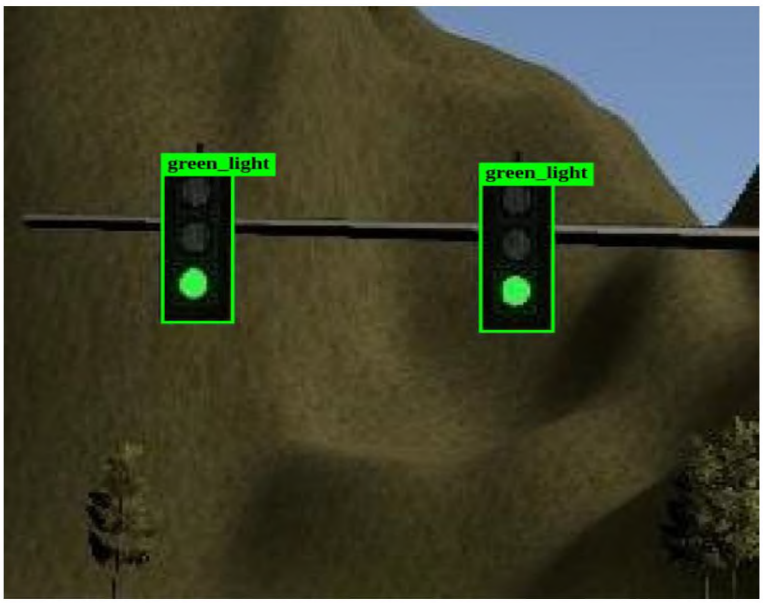


Рисунок 3.4 – Демонстрація виявлення світлофорів

Параметр data є єдиною необхідною опцією. Ви повинні передати йому файл дескриптора YAML. Параметр epochs визначає кількість циклів навчання (100 за замовчуванням). Існують інші параметри, які можуть вплинути на процес і якість навченої моделі. Кожен цикл навчання складається з двох фаз: фази навчання та фази перевірки. Під час фази навчання метод train виконує наступне:

– витягує випадкову групу зображень із навчального набору даних (кількість зображень у групі можна вказати за допомогою параметра batch);

– пропускає ці зображення через модель і отримує кінцеві обмежувальні прямокутники всіх виявлених об’єктів та їхніх класів;

– передає результат до функції втрати, яка використовується для порівняння отриманого результату з правильним результатом із файлів анотацій для цих зображень (функція втрат обчислює величину помилки);

– результат функції втрат передається в optimizer для коригування ваг моделі на основі кількості помилок у правильному напрямку. Це зменшує кількість помилок у наступному циклі. За умовчанням використовується оптимізатор SGD, але можна спробувати інші, наприклад Adam, щоб побачити різницю.

Під час етапу перевірки train виконує наступне:

– витягує зображення з набору даних перевірки;

– пропускає їх через модель і отримує виявлені обмежувальні рамки для цих зображень;

– порівнює отриманий результат із справжніми значеннями для цих зображень із текстових файлів анотацій;

– розраховує точність моделі на основі різниці між фактичними та очікуваними результатами.

Хід і результати кожної фази для кожної епохи відображаються на екрані. Таким чином ви можете побачити, як модель навчається та вдосконалюється від епохи до епохи. При виконанні коду train під час навчального циклу можна побачите результат, наприклад, рис. 3.5.

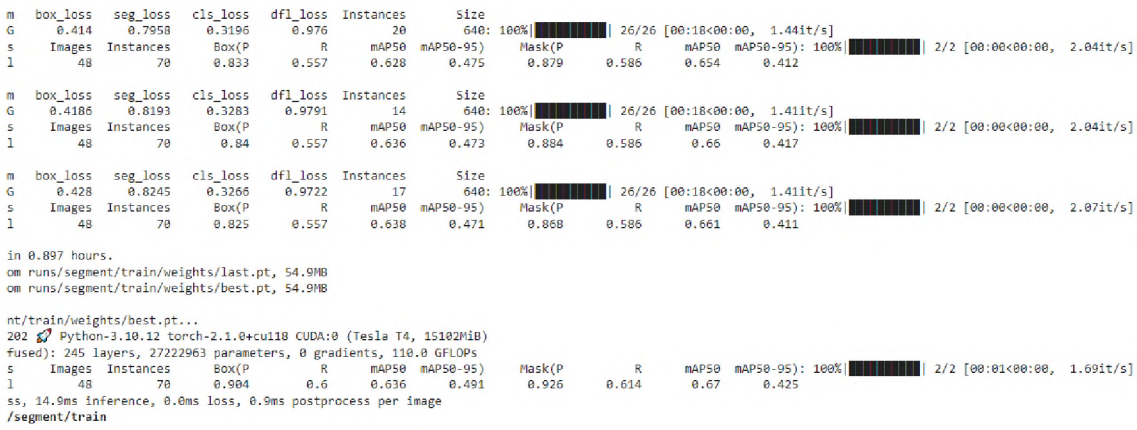


Рисунок 3.5 – Процес навчання нейронної мережі

Для кожної епохи відображається зведення як для фаз навчання, так і для фаз перевірки: рядки 1 і 2 показують результати фази навчання, а рядки 3 і 4 показують результати фази перевірки для кожної епохи. Етап навчання включає обчислення кількості помилок у функції втрат, тому найціннішими показниками тут є box_loss та cls_loss:

- box_loss показує кількість помилок у виявлених обмежувальних рамках.

- cls_loss показує кількість помилок у виявлених класах об'єктів.

Оскільки модель може правильно визначити координати обмежувальної рамки навколо об’єкта, але неправильно визначити клас об’єкта в цій рамці. Якщо модель справді дізнається щось із даних, ви побачите, що ці значення зменшуються від епохи до епохи. На попередньому знімку екрана box_loss зменшився: 0,7751, 0,7473, 0,742, а cls_loss також зменшився: 0,702, 0,6422, 0,6211. На етапі перевірки він обчислює якість моделі після навчання, використовуючи зображення з набору даних перевірки. Цінним показником якості є mAP_{50-95}, що є середньою точністю. Якщо модель навчається і вдосконалюється, точність повинна зростати від епохи до епохи. Результати навчання (у т. ч. графіки – рис. 3.6) можна знайти у папці «runs/...». Дана інформація відображається в кінці процесу навчання).

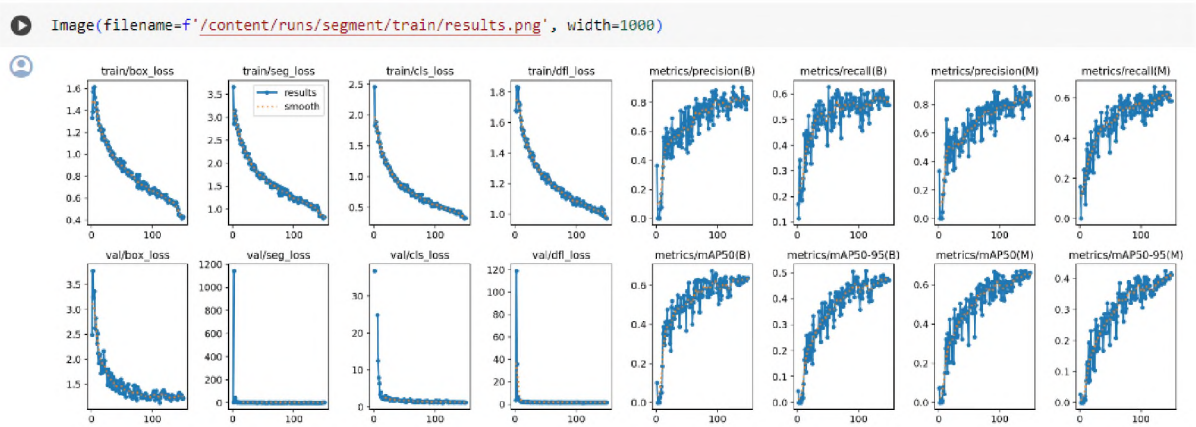


Рисунок 3.6 – Графіки результатів навчання

Якщо після останньої епохи не отримано прийнятної точності, можна збільшити кількість епох і запустити навчання знову. Крім того, можете налаштувати інші параметри, наприклад batch, lr0, lrf або змінити optimizer. Тут немає чітких правил, що робити, але є маса рекомендацій. На додаток до метрик, які показуються під час процесу навчання, на диск записується багато статистики. Коли починається навчання, він створює підпапку runs/detect/train в поточній папці, і після кожної епохи записує до неї різні файли журналу. Він також експортує навчену модель після кожної епохи до файлу /runs/detect/train/weights/last.p та модель з найвищою точністю до

файлу /runs/detect/train/weights/best.pt. Отже, після завершення навчання можете отримати файл best.pt для використання у виробництві.

3.3 Програмна реалізація синтезованої моделі

Для UAV є умовно «заборонені» об'єкти (ті, з якими він повинен уникати контакту), наприклад, люди, машини, ями, люки, лавки, урни, комплекси зупинок та ін. А є умовно «дозволені» об'єкти, наприклад, тротуари, пішохідні переходи. Після запуску початкового варіанту YOLOv8 переконались, що за умовчанням мережа «знає» людей і машини, але не детектує, наприклад, тротуар і перехід для пішоходів. Щоб UAV безпечно добиратися до місць призначення, необхідно адаптувати YOLOv8 до користувацьких даних і вимог.

Для реалізації розглянутої в роботі моделі виявлення об'єктів на основі нейронної мережі використовуємо хмарну версію блокноту Jupyter – Google Colab. Спочатку завантажуємо необхідні бібліотеки і модулі, наприклад: torch torchvision, ultralytics, YOLO, moviepy, os, gdown, zipfile, lang_sam, matplotlib, cv2, numpy, glob. Далі завантажуємо ваги попередньо навченої моделі, що зберігаються у архіві ves.zip. потім завантажуємо фрагменти відеострім (наприклад, з наявних файлів: IMG_20231028_151652_597.mp4, IMG_20231028_150852_876.mp4). Наступним кроком є формування функцій для підказок (промптів) для LangSAM, наприклад:

```
# люди та машини
def yolo_detect_person():
    !yolo task=detect mode=predict model=yolov8n.pt conf=0.6 source=/content/IMG_luki.mp4
save=True
# пішохідні переходи
def yolo_detect_perehod():
    !yolo task=detect mode=predict model=/content/best_perehod_segment.pt conf=0.2
source=/content/IMG_20231028_151652_597.mp4 save=True
# тротуари
def yolo_detect_trotuar():
    !yolo task=detect mode=predict model=/content/best_trotuar_500_150ep_m.pt conf=0.5
source=/content/IMG_20231028_150852_876.mp4 save=True
# тротуари
def yolo_detect_trotuar_asfalt():
    !yolo task=detect mode=predict model=/content/best_trotuar_500_150ep_m.pt conf=0.5
source=/content/IMG_20231028_013203_037.mp4 save=True
```

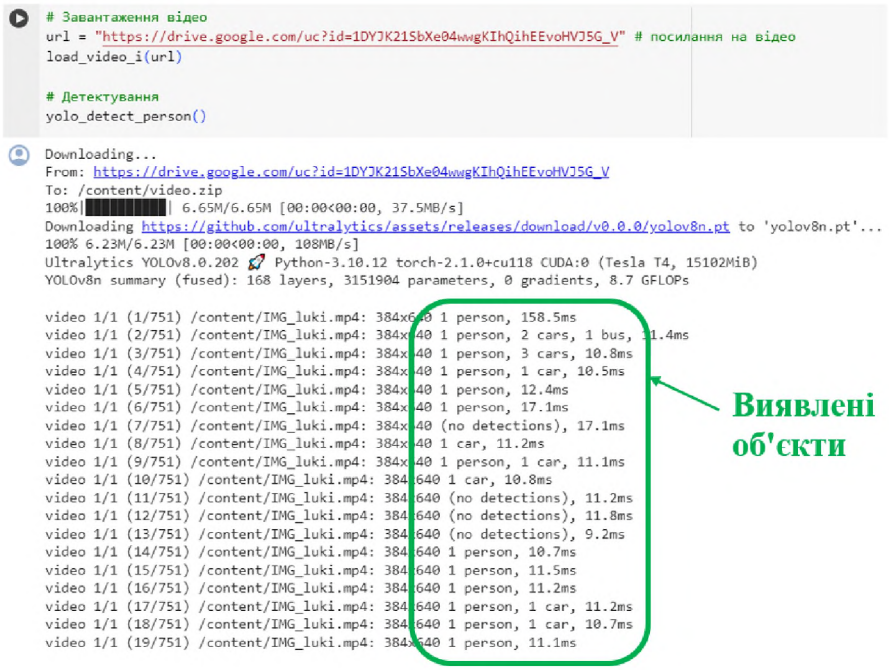
Потім формуємо промпт (підказку) для LangSAM, що містить назви об’єктів на які повинен реагувати UAV через крапку:

```
test_prompt = 'sidewalk.crosswalk.person.car.bus stop.bus.urn.pothole'
```

Надалі формуємо код, що безпосередньо виконує сегментацію необхідних об’єктів, наприклад, тротуару:

```
# sidewalk
!wget -q https://narodfm.ru/images/novosti/medium/cb10d9f5d6a57f582b37706624029ba4.jpg -O
sidewalk_1.jpg
sidewalk_1_path = '/content/sidewalk_1.jpg'
!wget -q https://img3.teletype.in/files/2f/12/2f1297cb-bd1d-48b4-940d-5ae47ebbe3b3.jpeg -O
sidewalk_2.jpg
sidewalk_2_path = '/content/sidewalk_2.jpg'
!wget -q https://vr-vyksa.ru/media/images/DSC_0797_RuFxVFe.width-1600.watermark-lb-10x10-0.6.jpg
-O sidewalk_3.jpg
sidewalk_3_path = '/content/sidewalk_3.jpg'
!wget -q https://upload.wikimedia.org/wikipedia/commons/6/60/Sidewalk_with_bike_path.JPG -O
sidewalk_4.jpg
sidewalk_4_path = '/content/sidewalk_4.jpg'
!wget -q https://st1.stpulscen.ru/images/product/040/380/958_medium2.jpg -O sidewalk_5.jpg
sidewalk_5_path = '/content/sidewalk_5.jpg'
sidewalk = [sidewalk_1_path, sidewalk_2_path, sidewalk_3_path, sidewalk_4_path, sidewalk_5_path]
```

Після запуску коду перевіряємо процес завантаження відео та працездатність розробленої моделі (рис. 3.7).




```
# Завантаження відео
url = "https://drive.google.com/uc?id=1DYJK21SbXe04wwgKIhQihEEvoHVJ5G_V" # посилання на відео
load_video_i(url)

# Детектування
yolo_detect_person()
```

```
Downloading...
From: https://drive.google.com/uc?id=1DYJK21SbXe04wwgKIhQihEEvoHVJ5G_V
To: /content/video.zip
100%|██████████| 6.65M/6.65M [00:00<00:00, 37.5MB/s]
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt to 'yolov8n.pt'...
100% 6.23M/6.23M [00:00<00:00, 108MB/s]
Ultralytics YOLOv8.0.202 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs

video 1/1 (1/751) /content/IMG_luki.mp4: 384x640 1 person, 158.5ms
video 1/1 (2/751) /content/IMG_luki.mp4: 384x640 1 person, 2 cars, 1 bus, 11.4ms
video 1/1 (3/751) /content/IMG_luki.mp4: 384x640 1 person, 3 cars, 10.8ms
video 1/1 (4/751) /content/IMG_luki.mp4: 384x640 1 person, 1 car, 10.5ms
video 1/1 (5/751) /content/IMG_luki.mp4: 384x640 1 person, 12.4ms
video 1/1 (6/751) /content/IMG_luki.mp4: 384x640 1 person, 17.1ms
video 1/1 (7/751) /content/IMG_luki.mp4: 384x640 (no detections), 17.1ms
video 1/1 (8/751) /content/IMG_luki.mp4: 384x640 1 car, 11.2ms
video 1/1 (9/751) /content/IMG_luki.mp4: 384x640 1 person, 1 car, 11.1ms
video 1/1 (10/751) /content/IMG_luki.mp4: 384x640 1 car, 10.8ms
video 1/1 (11/751) /content/IMG_luki.mp4: 384x640 (no detections), 11.2ms
video 1/1 (12/751) /content/IMG_luki.mp4: 384x640 (no detections), 11.8ms
video 1/1 (13/751) /content/IMG_luki.mp4: 384x640 (no detections), 9.2ms
video 1/1 (14/751) /content/IMG_luki.mp4: 384x640 1 person, 10.7ms
video 1/1 (15/751) /content/IMG_luki.mp4: 384x640 1 person, 11.5ms
video 1/1 (16/751) /content/IMG_luki.mp4: 384x640 1 person, 11.2ms
video 1/1 (17/751) /content/IMG_luki.mp4: 384x640 1 person, 1 car, 11.2ms
video 1/1 (18/751) /content/IMG_luki.mp4: 384x640 1 person, 1 car, 10.7ms
video 1/1 (19/751) /content/IMG_luki.mp4: 384x640 1 person, 11.1ms
```

Рисунок 3.7 – Перевірка працездатності моделі виявлення об’єктів

На рис. 3.8 наведено результати роботи нейронної мережі YOLOv8. Вони свідчать про адекватність запропонованої моделі виявлення об’єктів. Приклад програмного коду на мові Python наведений у Додатку А.

Рисунок 3.7 – Перевірка роботи моделі

Під час досліджень встановлено, що використаний в процесі навчання набір зображень потребує розширення та збагачення через, те що зустрічаються випадку коли неправильно розпізнаються тротуари з плиткою та калюжами.

3.4 Техніко-економічне обґрунтування прийнятих рішень

Запропонована в роботі модель виявлення об'єктів використовує сучасну архітектуру YOLOv8. Для її донавчання використовується SAM. Для розширення функціоналу UAV пропонується використання MobileSAM. Вона може використовуватись як додатковий інструмент для підвищення продуктивності магістральної нейронної мережі та системи автопілоту в цілому. Все це, дозволяє виявляти об'єкти під час роботи UAV, дозволяючи негайно реагувати або приймати рішення на основі виявлених об'єктів у реальному часі.

Згідно п. 1.1, для реалізації мінімального рівня монокулярного CV необхідний відеосенсор. Для інтеграції архітектури YOLO необхідно узгоджувати наявні обчислювальні ресурси, вимоги щодо швидкості виявлення об'єктів. Мінімальний рівень продуктивності такого компромісного рішення може спиратись на версії YOLO-NAS S або YOLOv8n. Для цього можна використовувати рішення Edge AI. Компанії Intel, Google і Nvidia пропонують апаратні платформи (прискорювачі) в малих форм-факторах (рис. 3.8), наприклад, Nvidia Jetson Nano, Google Coral USB, Intel Movidius NCS2.

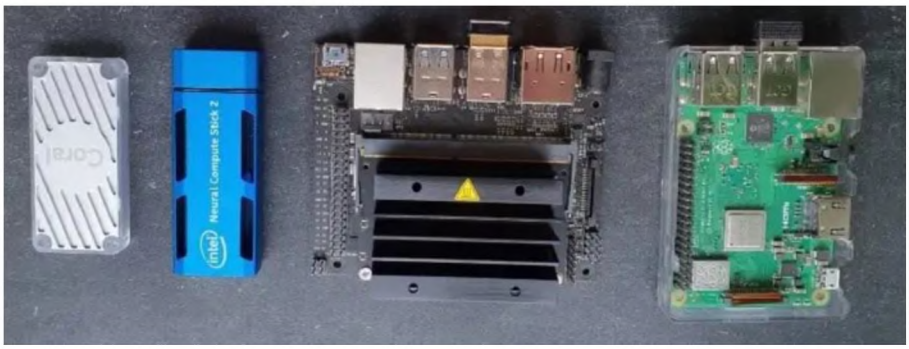


Рисунок 3.8 – Рішення Edge AI від Intel, Google і Nvidia

Вартість такого пристрою не перевищує 45000 грн (станом на листопад 2023 р.). Хоча всі три мають як слабкі, так і сильні сторони, все залежить від програми, бюджету та наявності наборів навичок. Враховуючі наявність

довідкового матеріалу по нейронним мережам YOLO та моделям SAM, необхідний робочий час на розробку коду складає 160 год. Згідно [53], витрати на заробітну плату програміста Python з відповідними навичками та досвідом роботи з проєктами AI складає 48000 грн. Як наслідок, сумарні витрати очікуються на рівні:

$$V = V_1 + V_2 \tag{3.1}$$

де V_1 – вартість апаратного сегменту;

V_2 – витрати на заробітну плату.

Таким чином, орієнтовні витрати на створення макету прототипу для реалізації запропонованої моделі виявлення об'єктів складають $V = 93000$ грн. При цьому розглядається ситуація наявності готової бази анатованих зображень. Приведення датасету до висунутих вимог може відбуватись за рахунок розширення та збагачення на основі синтетичних даних.

Висновки до розділу 3

В роботі проведено порівняння моделей для сегментації: SAM, FastSAM і MobileSAM. Для формування оцінки визначались не тільки параметри продуктивності нейронних мереж, але і необхідні обчислювальні ресурси. MobileSAM працює нарівні з оригінальною версією SAM. До того ж MobileSAM швидше і менше від FastSAM. Це дозволяє зробити висновок про доцільність використання MobileSAM на кінцевих пристроях, в тому числі, і компонентах систем керування UAV.

Для формування рекомендацій щодо практичної реалізації моделі виявлення об'єктів на основі нейронної мережі детально розглянуто процедури виконання завдання Object Detection на базі YOLOv8. Враховуючі наявність API високого рівня від к. Ultralytics, які за замовчуванням постачаються з пакетом YOLOv8, такий підхід дозволяє значно скоротити час

на розробку програмного сегменту моделі виявлення об’єктів. Ці API базуються на структурі PyTorch, яка використовувалася для створення більшої частини сучасних нейронних мереж.

В ході досліджень основна увага приділялась питанню донавчання YOLOv8 на датасеті користувача та спільному використанню її з версіями SAM. При цьому, використовувались підказки для сегментації об’єктів, які погана обробляє базова попередньо навчена модель YOLOv8.

Отримані результати підтвердили працездатність запропонованої моделі виявлення об’єктів та адекватність теоретичних положень, що в розглянуті в роботі.

ВИСНОВКИ

В роботі проведено аналіз впливу розробки та застосування AI та глибокого навчання на. Ці технології мають важливе значення для підвищення безпеки дорожнього руху.

Обґрунтовано важливість реалізації можливості UAV самостійно ідентифікувати оточення. Це досягається за допомогою вирішення завдань виявлення об’єктів, де AI на основі нейронних мереж відіграє ключову роль. Серед різних сучасних архітектур, студент виділяє YOLOv8, зазначаючи її здатність оптимально збалансувати точність і швидкість, що робить її ідеальною для виявлення об’єктів у реальному часі. YOLOv8 також пропонує широкий спектр попередньо навчених моделей для різних завдань, включаючи виявлення об’єктів, сегментацію та класифікацію.

В свою чергу, для самостійної ідентифікації оточення можливо реалізувати сегментацію об’єктів з використанням мереж на основі архітектури U-Net. Ці мережі ефективні завдяки механізму уваги, який дозволяє зосереджуватися на важливих аспектах зображення та ігнорувати неважливі, значно підвищуючи точність сегментації та знижуючи обчислювальні витрати. Досліджено варіант U-Net, що містить потрійний механізм уваги, який працює у різних вимірах, використовуючи канальну та просторову увагу.

В якості основного інструменту для сегментації об’єктів розглядаються версії SAM, що включає кодер зображень на основі візуального трансформеру та декодер маски, який генерує маску для виділення об'єктів із фону на основі підказок. SAM може генерувати кілька масок для одного запиту, що забезпечує гнучкість у роботі з неоднозначними зображеннями. Для зниження обчислювальних вимог під час навчання доцільно використовувати не оригінальну версію SAM, MobileSAM. Остання використовує легший кодер зображень, що робить його ідеальним для інтеграції в кінцеві пристрої та системи, такі як UAV.

Модель виявлення об'єктів на основі нейронної мережі дозволяє побудувати систему виявлення перешкод під час руху UAV. Вона складається з 3-ох основних етапів. При цьому, вона містить SAM з метою підвищення продуктивності моделі виявлення об’єктів. Використання MobileSAM дозволяє виявляти об’єкти під час роботи UAV, дозволяючи негайно реагувати або приймати рішення на основі виявлених об’єктів у реальному часі у випадку ситуації суперечливого, незрозумілого виявлення об’єкту.

Для формування рекомендацій щодо практичної реалізації виявлення об'єктів на основі нейронної мережі проведено перевірку працездатності та адекватності запропонованої моделі. Отримані результати підтвердили висунуті в роботі положення. Орієнтовні витрати на створення прототипу моделі виявлення об’єктів складають ≈ 93000 грн.

Таким чином, результатами роботи є розробка моделі виявлення об'єктів на основі нейронної мережі Object Detection, порівняльна оцінка точності нейронних мереж на основі модифікованих архітектур SAM, рекомендації щодо використання запропонованої архітектури нейронної мережі в інтересах вирішення завдання сегментації проїзної частини дороги. Вони можуть бути використані для подальших досліджень за даною тематикою та при реалізації автономного водіння безпілотних транспортних засобів.