Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею»

КвРКІП. 2202138.22.02.42 ПЗ

Виконав: студент 2 курсу, група КІ2м-22-2 _____ Рудик І.В.
Підпис Ініціали, прізвище

Керівник д. ф., доцент _____ Павлова О.О
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
22 05 2024 р.

Хмельницький, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"01" 09 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Рудику Івану Вікторовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею

Керівник проекту (роботи) Павлова О.О., д.ф., доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.01.2024 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз існуючих рішень у кіберфізичних системах комп'ютерного зору

Застосування нейронних мереж для обробки похибок у кіберфізичних системах комп'ютерного зору

Метод та алгоритм обробки похибок отриманих відеозображень нейронною мережею

Результати обробки похибок отриманих відеозображень нейронною мережею

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_01_»_____09_____2023р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	04.09.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	04.10.2023	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	02.11.2023	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2023	виконано
5	Робота над науковою статтею	06.02.2024	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	16.02.2024	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	18.04.204	виконано
8	Оформлення пояснювальної записки згідно вимог	21.04.2024	виконано
9	Попередній захист ДРМ	29.04.2024	виконано
10	Захист ДРМ на засіданні ЕК	09.05.2024	

Студент _____ Рудик І.В
Підпис Ініціали, прізвище

Керівник роботи _____ Павлова О. О.
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею.

Автор роботи: Рудик Іван Вікторович.

Керівник роботи: Павлова Ольга Олександрівна, д.ф., ст. викладач

Пояснювальна записка: 85 с., 49 рис., 16 табл., 4 дод., 73 джерела.

НЕЙРОННА МЕРЕЖА, КОМП'ЮТЕРНИЙ ЗІР, КІБЕРФІЗИЧНА СИСТЕМА, ОБ'ЄКТ, ОБРОБКА.

Об'єктом дослідження є підвищення якості роботи кіберфізичної системи комп'ютерного зору за рахунок обробки похибок у відеозображеннях.

Предметом дослідження є застосування нейронної мережі для обробки похибок у відеозображеннях.

Метою кваліфікаційної роботи магістра є забезпечення механізмів обробки похибок у відеозображеннях за допомогою нейронної мережі у кіберфізичній системі комп'ютерного зору

Поставлена мета досягається розв'язанням таких основних задач:

- розробка методів та алгоритмів обробки похибок у відеозображеннях;
- підготовка навчального датасету використовуючи відеозображення з цільових відеокамер;
- тренування моделі нейронної мережі;
- розробка програмно-технічного засобу для обробки похибок.

Наукова новизна отриманих результатів полягає у вдосконаленні існуючих методів та алгоритмів обробки нейронною мережею похибок у отриманих відеозображеннях.

Практична цінність отриманих результатів полягає в розробці програмно-технічного засобу для обробки похибок детекції об'єктів у отриманих відеозображеннях.

За темою дипломної роботи взято участь:

- у двох Всеукраїнських конференціях (АПКН-2023 у м. Хмельницький в листопаді 2023 року та IT&I у м. Миколаїв у лютому 2024 року) та опубліковано тези за матеріалами цих конференцій;
- у міжнародній конференції IntelITSIS-2024 та опубліковано статтю, яка індексується у наукометричній базі Скопус.

А також подано до друку тези для участі у Всеукраїнській конференції "Ольвійський форум" (22-24 червня м. Миколаїв).

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІТ – Інформаційні технології

ОТ – Операційні технології

КФС – Кіберфізичні системи

SoS System of Systems (система систем)

КЗ Комп'ютерний зір

КЗКФС Комп'ютерний зір у кіберфізичних системах

ШІ – Штучний інтелект

НМ – Нейронні мережі

Fps – frames per second (кадрів на секунду)

ВСТУП

Робота в напрямку автоматизації роботи систем відеонагляду ведеться практично від моменту появи таких систем. Апартатно-програмні рішення постійно вдосконалюються, але на практиці у світі існує величезний парк працюючих систем відеонагляду, що, з одного боку, забезпечують доволі низький відсоток правильної автоматичної детекції об’єктів внаслідок дії різних факторів [73] , а з іншого – є технічно цілком справними, і з комерційної точки зору їх повна заміна на сучасні рішення є недоцільною.

Протягом двох останніх десятиліть розвиток штучних нейронних мереж переживає справжній бум [4], що спричинено низкою факторів, не в останню чергу завдяки прогресу в мікроелектронній галузі. Завдяки цьому використання нейронних мереж стало доступним як для наукових досліджень, так і для комерційного застосування для широкого кола завдань, в тому числі і завдань, пов’язаних з детекцією об’єктів у відео, їх класифікацією, трекінгом тощо.

Актуальність даної роботи полягає в розробці кіберфізичної системи, яка б дозволила за рахунок використання навченої нейронної мережі зменшити похибку детекції об’єктів, в першу чергу людей, в гетерогенних системах відеоспостереження.

Метою кваліфікаційної роботи магістра є забезпечення механізмів обробки похибок у відеозображеннях за допомогою нейронної мережі у кіберфізичній системі комп’ютерного зору

Поставлена мета досягається розв’язанням таких основних задач:

– розробка методів та алгоритмів обробки похибок у відеозображеннях;

– підготовка навчального датасету використовуючи відеозображення з цільових відеокамер;

– тренування моделі нейронної мережі;

– розробка програмно-технічного засобу для обробки похибок.

Об’єктом дослідження є підвищення якості роботи кіберфізичної системи комп’ютерного зору за рахунок обробки похибок у відеозображеннях.

Предметом дослідження є застосування нейронної мережі для обробки похибок у відеозображеннях

Наукова новизна отриманих результатів полягає у вдосконаленні існуючих методів та алгоритмів обробки нейронною мережею похибок у отриманих відеозображеннях.

Практична цінність отриманих результатів полягає в розробці програмно-технічного засобу для обробки похибок детекції об’єктів у отриманих відеозображеннях.

За темою дипломної роботи взято участь:

- у двох Всеукраїнських конференціях (АПКН-2023 у м. Хмельницький в листопаді 2023 року та IT&I у м. Миколаїв у лютому 2024 року) та опубліковано тези за матеріалами цих конференцій;
- у міжнародній конференції IntelITSIS-2024 та опубліковано статтю, яка індексується у наукометричній базі Скопус.

А також подано до друку тези для участі у Всеукраїнській конференції "Ольвійський форум" (22-24 червня м. Миколаїв).

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У КІБЕРФІЗИЧНИХ СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Сучасний стан галузі кіберфізичних систем

Кіберфізичні системи (КФС) - це інтелектуальні системи, які включають в себе спроєктовані взаємодіючі мережі фізичних та обчислювальних компонентів. Ці тісно взаємопов'язані та інтегровані системи надають нові функціональні можливості для покращення якості життя та уможливлюють технологічний прогрес у критично важливих сферах, таких як персоналізована охорона здоров'я, реагування на надзвичайні ситуації, управління дорожнім рухом, реагування на надзвичайні ситуації, управління транспортними потоками, "розумне" виробництво, оборона і національна безпека, постачання та використання енергії.

Вплив КФС буде революційним і всепроникним; це очевидно вже сьогодні на прикладі створення нових автономних транспортних засобів, інтелектуальних будівель, інтелектуальних енергетичних систем, роботів та інтелектуальних медичних пристроїв тощо. Реалізація повного потенціалу КФС вимагає інтероперабельності між їх гетерогенними компонентами і підсистемами. [1]

Проектування КФС виходить за межі звичайного проектування продуктів, систем і додатків, яке традиційно проводиться за відсутності значного або всеохоплюючого взаємозв'язку. Для цього існує багато причин, деякі з них наведено нижче.

Традиційно інформаційні технології (IT) обробляли дані та здійснювали зв'язок, тоді як операційні технології (OT) зосереджувались на керуванні фізичними процесами. КФС долають цю різницю, інтегруючи як IT, так і OT. Датчики та збір даних цілком належать до сфери OT. IT бере на себе управління завдяки своїй потужності в аналізі даних, складних обчисленнях та можливостях зв'язку. Одним із ключових аспектів КФС є наявність часових обмежень. На відміну від традиційних IT-систем, КФС часто повинні реагувати та приймати рішення протягом дуже конкретних часових рамок, щоб забезпечити належне функціонування фізичної системи. Наприклад, в автономному автомобілі обробка

даних датчиків та коригування керма повинні відбуватися дуже швидко, щоб уникнути аварій. По суті, КФС представляють собою зміну парадигми, де IT та OT працюють разом безперебійно для створення інтелектуальних систем, які можуть здійснювати моніторинг, аналіз та керування фізичними процесами в режимі реального часу.

КФС може бути системою систем (System of Systems , SoS). Системи систем за своєю природою передбачають об'єднання різних доменів - цілей, часу та даних. Розуміння КФС як потенційних SoS підкреслює складність інтеграції різноманітних систем з різними цілями, що працюють на різних часових шкалах та використовують різні формати даних. Ефективне проектування SoS для КФС потребує ретельного розгляду цих міждоменних мостів, щоб забезпечити успішну колаборацію та загальну функціональність системи.

Емерджентна (неочікувана) поведінка є можливою для КФС. Розуміння поведінки, яка не може бути зведена до однієї підсистеми КФС, але виникає через взаємодію, можливо, багатьох підсистем КФС, є одним із ключових завдань аналізу. Наприклад, пробка на дорозі є шкідливою емерджентною поведінкою; оптимальний розподіл енергії розумною мережею, де споживачі електроенергії та виробники працюють разом, є бажаним позитивним ефектом.

КФС характеризуються їхньою взаємодією з робочим середовищем. Елементи КФС, спільно чи окремо, вимірюють параметри оточення, а потім обчислюють і впливають на своє оточення, як правило, змінюючи одну або більше спостережуваних властивостей (таким чином забезпечуючи замкнутий цикл керування). Середовище КФС зазвичай включає людей, і люди функціонують інакше, ніж інші компоненти КФС. В таких випадках архітектура повинна підтримувати різні режими взаємодії людини з КФС, включаючи: людину як контролера КФС або партнера в управлінні; людину як користувача КФС; людину як споживача продуктів діяльності КФС; і людину як безпосереднього об’єкта діяльності КФС.

Три аспекти повинні бути враховані при проектуванні, розробкці та експлуатації КФС: технічний, людський/соціальний та організаційний [4]. У

технічному аспекті - апаратне та програмне забезпечення повинно бути узгоджене з відповідною архітектурою. Крім того, КФС повинна бути інтегрована в існуючу фізичну та цифрову інфраструктуру; бажана сумісність з іншими системами повинна забезпечуватися за допомогою норм і стандартів. Людський/соціальний аспект включає в себе інтеграцію людей в КФС, або взаємодію людини з КФС. У цьому вимірі безпека у використанні та врахування етичних питань припроектуванні системи має важливе значення. визначається вбудовуванням КФС у цілі та контекст застосування різних інституційних структур та середовищ. Крім того, КФС можна розділити на три рівні застосування з точки зору розміру системи та охоплення. На мікрорівні КФС використовуються в індивідуальному контексті або в малих групах, як правило, обмежуючись місцевою територією. На мезорівні застосування КФС здійснюється в масштабах всієї організації і може мати міжрегіональні системні виміри. На макрорівні КФС розгортаються, часто як нестабільні системи систем, у сценаріях застосування, які охоплюють цілі національні економіки або є ще більш далекосяжними і розробляються на трансрегіональному або глобальному рівні В рамках цих рівнів, в контексті цифрової трансформації, КФС використовуються в різних сферах. До них належать міський розвиток (Smart City), охорона здоров'я (Smart Health), мобільність (Smart Mobility), управління будівлями (Smart Home) і, найпоширеніше, створення промислової вартості (Smart Manufacturing). Як зазначалося раніше, завдяки такому широкому спектру застосувань на різних рівнях і в різних сферах, КФС можна вважати технологією загального призначення. Відмінною характеристикою технологій такого роду є те, що вони можуть використовуватися широко і міжфункціонально з високим рівнем корисності. Як і попередні технології загального призначення, такі як паровий двигун, конвеєрні лінії або комп'ютери, КФС, у поєднанні з іншими технологіями цифрової епохи, мають потенціал вивільнити сплеск продуктивності, що може спричинити промислову революцію.

Рисунок 1.1 ілюструє переходи станів та їхній зв'язок у системі КФС [2]. У верхній частині рисунку показано, що дана система має початковий логічний

стан, який являє собою вектор логічних станів параметрів <L 1 ... L n>. На цей логічний стан впливають за допомогою логічних перетворень (Transformations of Logic, TL), що включають в себе обмін інформацією та операціями над цією інформацією за алгоритмами, рівняннями та іншими логічними процесами, в результаті чого утворюється новий логічний стан.

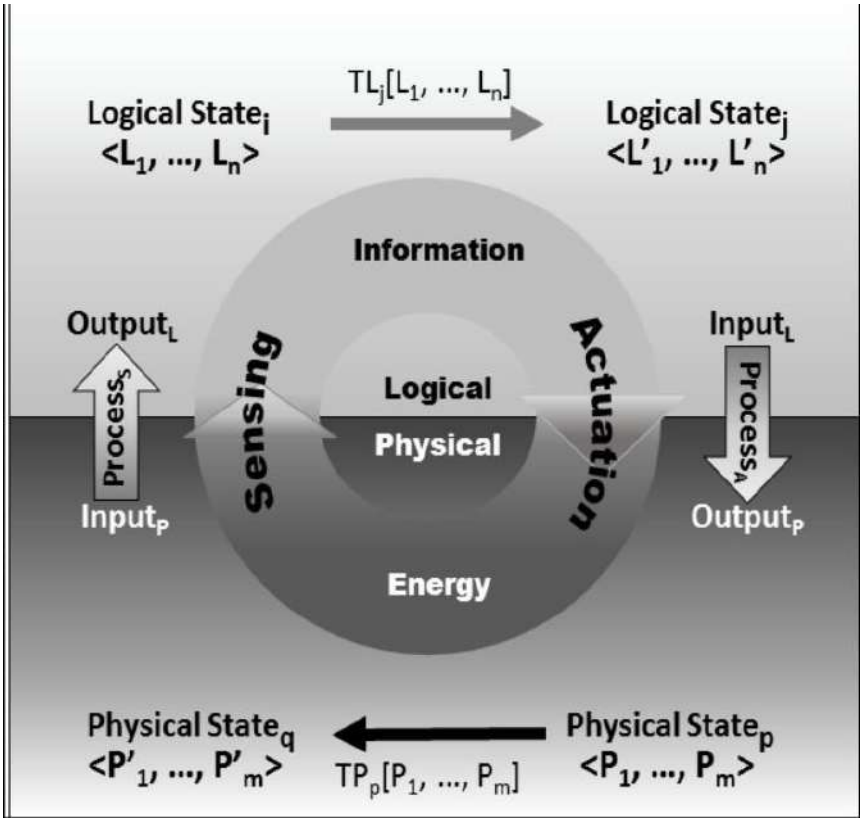


Рисунок 1.1 - Переходи станів та їхній зв'язок у системі КФС [1]

У нижній частині малюнка показано, що дана КФС система також має початковий фізичний стан, який є вектором параметрів фізичного стану <P 1... P m>. Цей фізичний стан керується за допомогою фізичних перетворень (Transformations of Physics, TP), що включають обмін та перетворення енергії, в результаті чого виникає новий фізичний стан. Енергія в цій моделі включає всі ентальпійні та ентропійні джерела та всі форми, потенційні і кінетичні, включаючи механічну, електричну, хімічну, теплову тощо.

Праворуч і ліворуч на рисунку показано зв'язок логічного та фізичного станів за допомогою перетворювачів (transducers). Датчики (ліворуч) реагують на

зміну фізичного стану (де Input P, наприклад, це рівень аналогового сигналу), виробляючи нове цифрове представлення цього стану (Output L, наприклад, нові значення параметрів, таких як температура і т.д.) для використання логічної частини системи. Обробка (Process S), необхідна для отримання Output L, передбачає застосування сенсорної моделі, яка включає формування сигналу, аналого-цифрове перетворення, калібрування, квантування та об'єднання метаданих. Обробка може здійснюватися локально інтелектуальним перетворювачем (transducer) або віддалено системою перетворювачів.

Приводи (виконавчі пристрої, actuators) забезпечують зворотну функцію, реагуючи на зміну логічного стану (Input L), шляхом створення нового фізичного стану (Output P), наприклад, новим рівнем аналогового сигналу на реле для використання у фізичній системі.

Хоча датчики і приводи функціонально інверсні, вони відрізняються у двох важливих аспектах щодо невизначеності. По-перше, Input P піддається фізичній невизначеності, тоді як Input L піддається обчислювальній невизначеності. По-друге, модель датчика і модель виконавчого механізму для обробки вносять свої власні форми невизначеності, які не є еквівалентними. Керування цими різними джерелами невизначеності та їх взаємодією для забезпечення якості проектування, в тому числі у критично важливих для безпеки додатках, залишається важливою сферою досліджень.

Підсумовуючи, суть цієї моделі взаємодії полягає в тому, що будь-яка значуща зміна логічного стану системи КФС призводить до зміни фізичного стану, і навпаки.

1.2 Технології та сфери застосування комп’ютерного зору у кіберфізичних системах

Комп'ютерний зір у кіберфізичних системах (КФС) являє собою поєднання обчислювальних алгоритмів і фізичних процесів, де збір і аналіз даних на основі зору відіграють вирішальну роль у забезпеченні інтелектуальної та автономної

роботи. Ці системи є високоінтегрованими, фізичні та обчислювальні компоненти тісно взаємодіють. Застосування комп'ютерного зору в КФС значно розширилося завдяки розвитку штучного інтелекту, сенсорних технологій і обчислювальних потужностей. Огляд ключових технологій і сфер застосування наведено в таблицях 1-2.

Таблиця 1.1 - Технології комп’ютерного зору, що застосовуються в КФС

Технологія	Опис	Основні застосування
Глибоке навчання та нейронні мережі	Використовують складні структури, модельовані на зразок людського мозку, для інтерпретації візуальних даних.	Розпізнавання зображень, детекція об'єктів, розуміння сцен
3D-зір	Включає використання сенсорів, таких як LiDAR та стереокамери, для сприйняття глибини.	Автономна навігація, маніпуляція об'єктами
Обробка зображень та відео в реальному часі	Зосереджена на негайному аналізі візуальних даних для своєчасної відповіді.	Автономні транспортні засоби, спостереження
Доповнена реальність (AR) та віртуальна реальність (VR)	Інтегрують цифрові елементи у фізичний світ (AR) або створюють занурювальні віртуальні середовища (VR), покладаючись на комп'ютерний зір для відстеження та взаємодії.	Покращені користувацькі досвіди, навчання, розваги
Периферійні обчислення (Edge Computing)	Включає обробку даних безпосередньо на пристрої для зниження затримок і використання пропускної здатності, критично для часочутливих застосунків.	Всі застосунки КФС, що вимагають негайної обробки даних

Таблиця 1.2 - Сфери застосування комп’ютерного зору

Галузь застосування	Опис	Використовувані технології
Автономні транспортні засоби	Транспортні засоби, які орієнтуються в просторі і працюють без людського втручання, розуміючи оточення.	Глибоке навчання, 3D-зір, обробка в реальному часі
Виробництво та Робототехніка	Автоматизація промислових процесів та завдань з точністю та ефективністю.	Глибоке навчання, обробка в реальному часі, 3D-зір
Охорона здоров'я	Підтримка в діагностиці, хірургічних процедурах та моніторингу пацієнтів за допомогою візуального аналізу.	Глибоке навчання, периферійні обчислення
Сільське господарство	Оптимізація сільськогосподарських практик через моніторинг, детекцію захворювань і автоматизацію.	Глибоке навчання, 3D-зір
Спостереження та безпека	Посилення безпеки через неперервне спостереження та виявлення загроз.	Глибоке навчання, обробка в реальному часі, периферійні обчислення
Розумні міста	Покращення урбаністичного життя через управління трафіком, обслуговування інфраструктури та безпеку.	Глибоке навчання, обробка в реальному часі, периферійні обчислення
Торгівля	Трансформація досвіду покупок через управління запасами та взаємодію з покупцями.	Глибоке навчання, периферійні обчислення
Моніторинг навколишнього середовища	Зусилля з охорони природи та управління катастрофами через моніторинг дикої природи, лісів тощо.	Глибоке навчання, 3D-зір, периферійні обчислення

Останнім часом комп'ютерний зір широко досліджується в області виявлення об'єктів для промислової автоматизації, побутової електроніки, медичної візуалізації, військового та відеоспостереження. Світовий ринок комп'ютерного зору сягнув межі $50 млрд ще наприкінці 2020 року [3].

Для розпізнавання об'єктів, необроблені вхідні дані представляються у формі пікселів матриці, де перший шар репрезентації абстрагує пікселі та кодує краї, наступний шар компонує та кодує розташування країв, наступний шар кодує очі та ніс, а останній шар розпізнає обличчя, присутнє на зображенні. Як правило, процес глибокого навчання оптимально класифікує риси обличчя за відповідними рівнями без нагляду.

Завдяки функції автоматичної екстракції ознак моделі глибокого навчання в задачах в комп'ютерного зору стали високоточними. Архітектура Deep CNN передбачає складні моделі. Вони вимагають великих наборів даних зображень для більш високої точності. CNN потребують великих наборів даних з мітками для виконання суміжних задач у сфері комп'ютерного зору, таких як класифікація об'єктів, виявлення, відстеження об'єктів та розпізнавання.

Для виконання як навчання, так і тестування, глибоке навчання вимагає потужних обчислювальних ресурсів, в тому числі потужних графічних процесорів (GPU), і більших наборів даних. У комп'ютерному зорі класифікація зображень є найбільш широко дослідженою областю, і вона досягла приголомшливих результатів завдяки методам глибокого навчання.

У загальному виявленні об'єктів основна мета полягає в тому, щоб визначити, чи є на зображенні екземпляри об'єктів із зазначених різновидів (наприклад, тварини, транспортні засоби та пішоходи), і якщо вони є, то визначити просторове розташування та протяжність окремого об'єкта. Виявлення об'єктів стало основою для вирішення більш складних завдань, пов'язаних із комп'ютерним зором: розуміння сцени, підписування зображень, сегментація екземплярів, семантична сегментація, розпізнавання об'єктів та відстеження. Застосування виявлення об'єктів охоплює такі області, як Інтернет речей (IoT) і штучний інтелект, який включає інтелектуальні військові системи спостереження,

безпеку, безпілотні автомобілі, зір роботів, взаємодію людини з комп'ютером, і побутову електроніку.

Методи глибокого навчання досягли великого прогресу у виявленні об'єктів, проблемі, яка привернула увагу багатьох дослідників у цьому десятилітті. Відеоспостереження є одним з найскладніших і фундаментальних напрямків в системах безпеки, оскільки воно повністю залежить від виявлення та відстеження об'єктів. Він відстежує поведінку людей у громадських місцях, щоб виявити будь-яку підозрілу поведінку.

Еволюція систем виявлення об’єктів у кіберфізичних системах комп'ютерного зору (КФСКЗ) представлена на рисунку 1.2. Заявлені цілі виявлення об'єктів полягають у досягненні як високої точності, так і високої ефективності шляхом розробки надійних алгоритмів виявлення об'єктів.

Досягнення високої точності виявлення пов'язане з наступними викликами:

- варіативністю всередині класів: варіації реальних об'єктів включають варіації кольору, розміру, форми, матеріалу та пози;
- умовами зображення і необмежені середовища: такі фактори, як освітлення, погодні умови, оклюзія, фізичне розташування об'єкта, точка зору, завади, тіні, розмиття і рух;
- шумами зображень: такі фактори, як зображення з низькою роздільною здатністю, шум стиснення, спотворення фільтрів;
- різноманітністю структурованих і неструктурованих категорій об'єктів реального світу, які повинен розрізняти детектор.

Досягнення високої ефективності пов'язане з певними викликами:

- мобільні пристрої низького класу мають обмежений обсяг пам'яті, низьку швидкість і низькі обчислювальні можливості;
- необхідно розрізняти величезне різноманіття класів об'єктів відкритого світу;
- наявність зображень і відеоданих великого розміру;
- неможливість обробляти об'єкти, яких раніше ніколи не бачили.

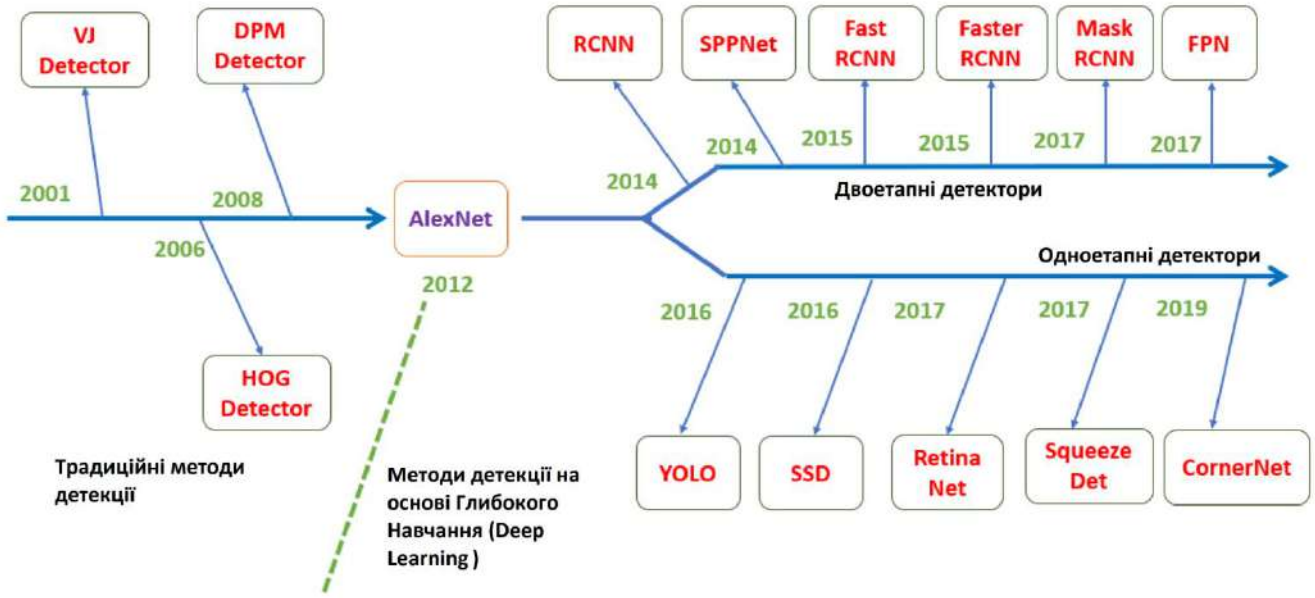


Рисунок 1.2 - Еволюція систем виявлення об’єктів у КФС [3]

Напрями наукових досліджень. Незважаючи на значний прогрес, досягнутий у галузі виявлення об'єктів, технологія все ще залишається значно далекою від людського зору, вирішуючи реальні проблеми, такі як: виявлення об'єктів в обмежених умовах, робота у відкритому світі та інші способи. Виходячи з цих викликів, напрямки подальших досліджень мають бути зосереджені в наступних напрямках :

1. Більш ефективні фреймворки виявлення: Основна причина успіху виявлення об'єктів пов'язана з розробкою високоякісних фреймворків виявлення, як у двоступеневих, так і в одноетапних детекторах (RCNN, Fast/Faster/Mask RCNN, YOLO та SSD). Двоступеневі детектори демонструють високу точність, тоді як одноступеневі детектори простіші та швидші. Детектори об'єктів багато в чому залежать від базових системоутворюючих моделей, і більшість з них оптимізовані для класифікації зображень, що, можливо, спричиняє зміщення навчання; І може бути корисно розробити нові детектори об'єктів, навчаючись з нуля.

2. Компактні та ефективні функції CNN: шари CNN збільшені в глибину від декількох шарів (AlexNet) до сотень шарів (ResNet, ResNext, CentreNet, DenseNet). Всі ці мережі вимагають багато даних і високопродуктивних графічних

процесорів для навчання, оскільки вони мають мільярди параметрів. Таким чином, щоб ще більше зменшити надмірність мережі, дослідники повинні проявити інтерес до проектування легких і компактних мереж.

3. Слабоконтрольоване виявлення: В даний час всі сучасні детектори використовують тільки мічені дані з масками сегментації об'єктів або обмежувальними рамками на повністю контрольованих моделях. Але за відсутності мічених навчальних даних повністю контрольоване навчання не є масштабованим, тому важливо розробити модель, де доступні лише частково мічені дані.

4. Ефективна магістральна архітектура для виявлення об'єктів: Робиться шляхом прийняття ваг попередньо навчених моделей класифікації, оскільки вони навчаються на великомасштабних наборах даних для завдань виявлення об'єктів. Таким чином, прийняття попередньо навченої моделі може не привести до оптимального рішення через конфлікти між завданнями класифікації зображень і виявлення об'єктів. В даний час більшість детекторів об'єктів засновані на магістралях класифікації, і лише деякі використовують різні магістральні моделі (наприклад, SqueezeDet на основі SqueezeNet). Таким чином, існує потреба в розробці моделі легкої магістралі з урахуванням виявлення об'єктів у реальному часі.

5. Виявлення об'єктів в інших модальностях: В даний час більшість детекторів об'єктів працюють тільки з 2D-зображеннями, але виявлення в інших модальностях - 3D, LIDAR і т.д. - буде дуже актуальним в областях застосування, таких як безпілотні автомобілі, дрони і роботи. Однак, знову ж таки, виявлення 3D-об'єктів може викликати нові проблеми з використанням відео, глибини та хмарних точок.

6. Оптимізація мережі: Вибір оптимальної мережі виявлення забезпечує ідеальний баланс між швидкістю, пам'яттю та точністю для конкретної програми та на вбудованому обладнанні. Незважаючи на те, що точність виявлення знижується, краще навчати компактним моделям з невеликою кількістю

параметрів, і цю ситуацію можна подолати, запровадивши навчання підказками, дистиляцію знань і кращі схеми попереднього навчання.

7. Адаптація масштабів: Це більш очевидно при виявленні облич і натовпу; Об'єкти, як правило, існують в різних масштабах. Для того, щоб підвищити стійкість до вивчення просторових перетворень, необхідно навчати спроектовані детектори масштабно-інваріантним, мульти-масштабним або масштабно-адаптивним способами:

а) для масштабно-адаптивних детекторів необхідно створити механізми уваги, сформувати каскадну мережу та масштабувати оцінку розподілу для адаптивного виявлення об'єктів;

б) для мульти-масштабних детекторів, таких як GAN (generative adversarial network), або FPN (feature pyramid network) генерують мульти-масштабну карту ознак;

в) для масштабно-інваріантних детекторів рекомендованими будуть магістральні моделі, такі як AlexNet та ResNet.

8. Навчання крос-наборів даних: Навчання крос-наборів даних для виявлення об'єктів має на меті виявити об'єднання всіх класів у різних існуючих наборах даних за допомогою єдиної моделі та без додаткового маркування, що, у свою чергу, вирішує ресурсоємну проблему маркування нових класів на всіх існуючих наборах даних. Використовуючи навчання крос-наборів даних, потрібно лише позначити нові класи в новому наборі даних. Такий підхід широко використовується в промислових додатках, які зазвичай стикаються з проблемою збільшення класів.

Дослідження в багатьох областях ще далекі від завершення, зокрема, в частині використання розподілених і периферійних обчислень в системах відеоспостережень, в тому числі з використанням спеціалізованих мікропроцесорів обробки візуальної інформації (VPU -vision processing unit) [5].

1.3 Аналіз похибок, які виникають у кіберфізичних системах комп’ютерного зору

Кіберфізичні системи комп'ютерного зору (КФСКЗ) схильні до різних типів похибок, які можуть виникати на різних етапах обробки інформації. Перелік найбільш поширених з них, причини та способи їх мінімізації наведно в таблицях 1.3 – 1.4.

Таблиця 1.3 - Типи помилок КФСКЗ та їх ознаки

Тип помилки	Опис
Помилки датчиків	Помилки, що виникають через неточні або зашумлені дані, що надійшли з датчиків.
Помилки алгоритмів	Помилки в алгоритмах обробки та інтерпретації зображень, що призводять до неправильних результатів.
Похибки обробки даних	Проблеми, пов'язані з обробкою даних (наприклад, затримки обробки, втрата або пошкодження даних).
Помилки інтеграції	Помилки можуть виникнути через інтеграцію систем комп'ютерного зору з іншими компонентами кіберфізичної системи.
Помилки виконавчих пристроїв (приводів)	Помилки, що виникають на етапі виконання, наприклад неправильний рух або інші дії виконавчих пристроїв.

На рисунку 1.3 наведено приклади помилок розпізнавання образів у КФСКЗ: (а) помилкове розпізнавання вантажівки та людини за кермом бульдозера; б) помилкове розпізнавання рефрижератора та вантажівки на місці вантажних контейнерів; в) помилкове впізнання вантажного автомобіля та людини на місці вантажного контейнера та розмежувального стовпчика; г)

помилкове розпізнавання вантажних автомобілів замість мікроавтобусів та людини замість обмежувальної перегородки).

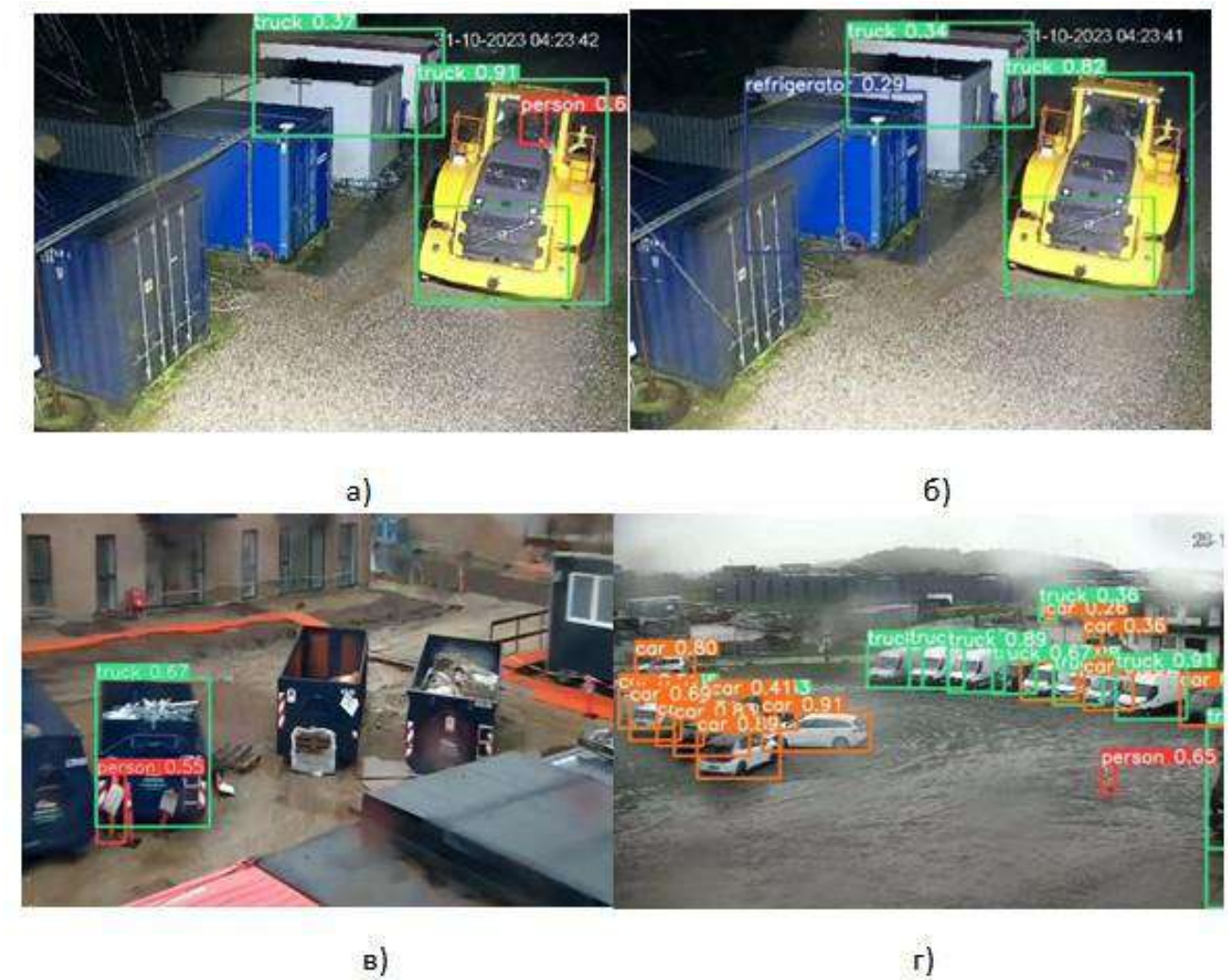


Рисунок 1.3 - Приклади помилок розпізнавання образів у КФСКЗ

Найбільш розповсюджені причини помилок що виникають у КФСКЗ наведено в таблиці 1.4

Формули для кількісної оцінки найбільш поширених типів похибок у кіберфізичних системах комп'ютерного зору (КФСКЗ) наведені нижче.

1. Похибки датчиків.

А) Шум. Це тип похибки сенсора, який виникає через випадкові флуктуації в сигналі. Характеризується середньоквадратичним відхиленням і співвідношенням сигнал-шум (Signal-to-noise ratio, SNR).

Середньоквадратичне відхилення розраховується як

$$\sigma = \sqrt{(\Sigma(x\_i - \mu)\text{^}2 / N)}, \quad (1.1)$$

де x_i - значення і-го вимірювання;

μ - середнє значення вимірювань;

N- кількість вимірювань.

Таблиця 1.4 - Причини помилок КФСКЗ

Причина помилки	Опис
Змінність навколишнього середовища	Зміни в освітленні, погодні умови або присутність неочікуваних об'єктів тощо можуть вплинути на точність датчиків.
Обмеження апаратного забезпечення	Обмеження в роздільній здатності датчиків, обчислювальна потужність або пам'ять можуть обмежити продуктивність системи.
Помилки програмного забезпечення	Недоліки в коді можуть призвести до неочікуваної поведінки або збоїв системи.
Причина помилки	Опис
Обмеження проектування	Внутрішні обмеження алгоритмів або моделей, як-от нездатність узагальнювати навчальні дані.
Людський фактор	Помилки в дизайні системи, конфігурації або експлуатації через людську помилку.

Співвідношення сигнал-шум розраховується як

$$SNR = 10 \cdot \log_{10}(P\_signal / P\_noise), \quad (1.2)$$

де P_signal – потужність сигналу;
P_noise – потужність шуму.

Б) Спотворення. Це тип похибки сенсора, який виникає, коли виміряне значення систематично відрізняється від істинного значення. Спрощено, спотворення можна розділити на лінійні та нелінійні.

Лінійні спотворення. Виміряне значення пропорційне істинному значенню, але з додаванням константної похибки. Розраховується як

$$f(x) = ax + b, \quad (1.3)$$

де a, b - коефіцієнти лінійного спотворення;
f(x) – виміряне значення;
x – істинне значення.

Нелінійні спотворення. Розповсюдженим видом нелінійних спотворень є, зокрема квадратичні спотворення, коли виміряне значення пропорційне квадрату істинного значення. В цьому випадку нелінійне спотворення розраховується як

$$f(x) = ax^2 + bx + c, \quad (1.4)$$

де a, b, c - коефіцієнти нелінійного спотворення;
f(x) – виміряне значення;
x – істинне значення.

2. Похибки алгоритмів комп'ютерного зору. Характеризуються такими параметрами, як Точність (Precision), Акуратність (Accuracy), Відгук (Recall), F1-оцінка (F1 score).

Точність (Precision). Показує, скільки з позитивних прогнозів виявилося правильними.

$$Precision = TP / (TP + FP) , \quad (1.5)$$

де TP - кількість правильно класифікованих позитивних прикладів;
TN - к ількість правильно класифікованих негативних прикладів.

Акуратність (Accuracy). Вимірює, скільки з усіх прогнозів, зроблених моделлю, є правильними.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN), \quad (1.6)$$

де TP - кількість правильно класифікованих позитивних прикладів;
TN - к ількість правильно класифікованих негативних прикладів;
FP -кількість помилково класифікованих позитивних прикладів (хибні спрацьовування);
FN - к ількість помилково класифікованих негативних прикладів (пропуски).

Відгук (Recall). Це частка позитивних випадків, які модель правильно визначила.

$$Recall = TP / (TP + FN) , \quad (1.7)$$

де TP - кількість правильно класифікованих позитивних прикладів;
TN - к ількість правильно класифікованих негативних прикладів.

F1-оцінка (F1 score). Є метрикою для вимірювання продуктивності моделі у задачах класифікації. Вона поєднує точність і відгук в один показник, щоб

забезпечити збалансовану оцінку точності моделі. Особливо актуальним є використання цього показника в наборах даних, розбалансованих за класами [38].

$$F1 = 2 * (Precision * Recall) / (Precision + Recall) , \quad (1.8)$$

де Recall – відгук (1.7);
Precision – точність (1.5).

Цей же показник F1 може бути розрахований без необхідності розрахунку Precision та Recall:

$$F1=TP / (TP + (FP + FN) / 2) \quad (1.9)$$

1.4 Постановка задачі та вибір технологій для реалізації

У даній роботі планується розробити спосіб зменшення похибок системи комплексного відеонагляду. В якості прикладу з дозволу власників було вибрано таку систему, що містить понад 1200 камер спостереження, і належить одній з європейських компаній. Щодня її оператори отримують сигнали про порушення з понад 400 камер. Середньодобова кількість сигналів становить майже 3 тис. (понад 86 тис. за минулий місяць). Слід зазначити, що більшість камер розташовані ззовні і на них суттєво впливають різноманітні фактори, що викликають помилкові спрацьовування. Ці фактори можна розділити на категорії: 1) опади (дощ, сніг); 2) вітер (розгойдує предмети в полі зору камери, або саму камеру); 3) світлові ефекти (мерехтіння світла на території або за її межами, фари автомобіля вночі, відблиски сонця або просто тінь від хмари вдень). Усі зазначені фактори, а частіше – їх комбінації – складають значну частину всіх активацій камери. Так, за згаданий минулий місяць кількість хибних тривог сягнула 44 тисяч, тобто понад 51% від усіх тривожних сигналів. Враховуючи, що кожен сигнал тривоги повинен бути перевірений людиною – оператором, - помилкові

спрацювання завдають значних збитків власнику, змушуючи тримати на роботі збільшену кількість операторів.

Слід також зазначити, що згадана система характеризується різноманіттям:

- виробників та моделей камер;
- типів сенсора камери (ч/б, кольоровий, термо);
- розмірів зображення (шириною від 640 до 1920 з різним співвідношенням сторін);
- підсистем детекції об’єктів (від різних виробників і лізингових сервісів);
- умов експлуатації (для внутрішнього / зовнішнього використання);
- типів медіафайлів і їх форматів тощо (відео чи зображення).

Як один із варіантів вирішення поставленої задачі розглядався варіант детекції об’єктів у відеокліпах периферійними апаратно-програмними комплексами. Але внаслідок гетерогенності структури системи, та значних вимог щодо деталізації вхідних відеосигналів було прийнято рішення використати варіант з централізованою пост-обробкою відео. Імплементація подібного рішення на апаратній базі працюючої системи створило б потенційні ризики непланових пікових навантажень, аж до відмови працездатності всієї системи.

Тому було прийняте рішення створення кіберфізичної системи комп'ютерного зору (КФСКЗ) як окремого апаратно-програмного комплексу. Схему функціонування комплексу наведено на рисунку 1.4.

Завдання, які необхідно вирішити у даній магістерській роботі:

1. Підготовка датасету для навчання нейронної мережі.
2. Навчання нейронної мережі, валідація та тестування детекції об’єктів.
3. Аналіз отриманих даних.
4. Організація обміну даними між створеною системою і існуючою системою управління відео-даними.

Для реалізації поставлених завдань будуть використані наступні технології:

- нейромережа для детекції об’єктів на відео: YOLOv8;
- середовище для роботи з нейромережею: Python 3.9+;

- обмін даними між підсистемами здійснюється через Інтернет за допомогою технології webhooks. Для цього задіяні бібліотеки Flask у Python на стороні КСКЗ, PHP + CURL на стороні існуючої системи управління відео-даними.

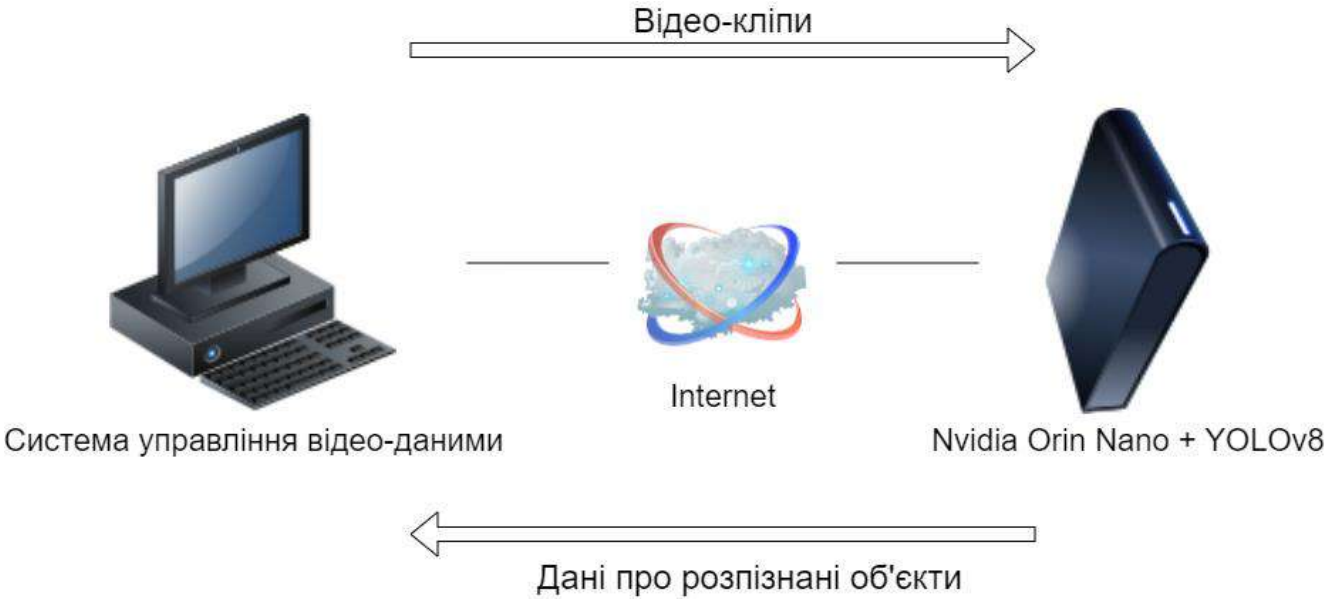


Рисунок 1.4 - Схема функціонування КСКЗ

1.5 Висновки

В даному розділі було проаналізовано сучасний стан галузі кіберфізичних систем (КФС), технології та сфери застосування комп’ютерного зору (КЗ) у КФС. Крім цього, було проведено аналіз похибок, що виникають у КС КФС, їх причини та стратегії мінімізації.

Було проведено аналіз наукових публікації на схожу тематику; також були розглянуті варіанти можливої реалізації вирішення подібних проблем.

Аналіз предметної галузі показав, що з огляду на останні досягнення в алгоритмах детекції об’єктів у зображеннях, і зважаючи на необхідні апаратні ресурси, найкращим варіантом буде використання нейромережі YOLOv8.

2 ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ОБРОБКИ ПОХИБОК У КІБЕРФІЗИЧНИХ СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ

2.1 Методи обробки похибок у кіберфізичних системах комп'ютерного зору

Для усвідомлення можливих похибок у КФСКЗ слід розуміти принаймні узагальнено схему їх функціонування. Приклад такої схеми наведено нижче на рисунку 2.1.

Рисунок 2.1 - Приклад функціонування КФСКЗ

Похибки комп'ютерного зору можна згрупувати в кілька основних категорій, які охоплюють різні сторони роботи СКЗ та їх взаємодію з навколишнім середовищем:

- помилки сприйняття;
- помилки обробки даних;
- системні помилки.

Розглянемо кожну з цих категорій детальніше.

Помилки сприйняття. Це, по-перше, помилки через обмеженість навчальних даних: Наприклад, самокерований автомобіль може не розпізнавати рідкісні дорожні знаки через недостатню кількість даних в навчальних наборах даних. Також до цієї категорії слід віднести помилки попередньої обробки даних. Наприклад, неправильна нормалізація або фільтрація даних може спотворювати об'єкти на зображеннях. Крім того, до помилок сприйняття відносяться помилки оклюзії та помилки зміни освітлення. Прикладом тут може бути ситуація, коли

об'єктив камери вкритий брудом, що призводить до неправильного сприйняття оточення.

Помилки обробки даних. Насамперед, сюди слід віднести алгоритмічні помилки. Наприклад, розроблений алгоритм може не підходити для певного типу даних, наприклад, розпізнавання облич може працювати неправильно на різних етнічних групах. Або обраний алгоритм детекції неправильно оцінює розмір об'єкта. Крім того, до цієї категорії відносяться помилки, пов'язані з апаратним забезпеченням. Зокрема, це можуть бути помилки, що виникають через апаратні обмеження. Тобто, наприклад, моделі можуть бути обмежені лімітами пам'яті апаратного забезпечення. Або помилки, викликані нештатною роботою обладнання, наприклад, перегрівом процесора.

Системні помилки. Це можуть бути помилки, викликані порушенням зв’язку. Пакети даних можуть втрачатися під час передачі (наприклад, через зіткнення з перешкодою), або приходити з затримками. Це все може призвести до виникнення помилок у КФС. До цієї ж категорії слід віднести й помилки синхронізації даних з різних датчиків або компонентів системи.

Розширена схема класифікації похибок комп’ютерного зору у КФС, із вказанням можливих методів їх вирішення наведено в Додатку Г. Крім цього, при проектуванні подібних систем рекомендується дотримуватися певних стратегій, перелік яких наведено нижче в таблиці 2.1.

Таблиця 2.1- Стратегії мінімізації помилок КФСКЗ.

Стратегія мінімізації	Опис
Резервування	Використання кількох датчиків або алгоритмів для перехресної перевірки даних та рішень.
Надійний дизайн	Розробка систем, які можуть толерувати або адаптуватися до змін та невизначеностей у своєму середовищі.

Кінець таблиці 2.1- Стратегії мінімізації помилок КФСКЗ.

Стратегія мінімізації	Опис
Неперервне навчання	Впровадження механізмів для неперервного вдосконалення системи на основі нових даних та досвіду.
Тестування та валідація	Проведення ретельного тестування в широкому діапазоні умов для виявлення потенційних помилок.
Обробка помилок та відновлення	Розробка систем зі здатністю виявляти помилки та коригувати їх або безпечно вимикатися.
Навчання користувачів	Забезпечення того, щоб користувачі розуміли можливості та обмеження системи для уникнення неправильного використання.

2.2 Опис моделей нейронних мереж, які використовуються у роботі

У ході дослідження проведено аналіз останніх наукових публікацій у галузі розпізнавання образів за допомогою штучних нейронних мереж. Наукові публікації [1-21] присвячені застосуванню моделей на основі штучних нейронних мереж, таких як Google Cloud Vision API, Pytorch Faster R-CNN, OpenCV+CNN та бібліотеки YOLO для різних галузей, таких як біологія, медицина, розумні міста та кібернетика. -переглянуто фізичні системи, розпізнавання жестів та міміки. Результати аналізу наукових публікацій представлено в таблиці 2.2.

На основі наведеного нижче аналізу існуючих рішень [51-70] було вирішено зібрати набір даних із 12 тестових зображень і провести порівняльний аналіз якості розпізнавання образів.

Для тестування були обрані Google Cloud Vision API [48], Pytorch FasterR-CNN [49] і нейронна мережа YOLOv8 [7]. Набір даних, який був зібраний і підготовлений для тестування моделей, представлений на рисунку 2.2. Він містить зображення в інфрачервоному світлі, у відтінках сірого, на яких є

зображення різних об’єктів на будівельному майданчику, автомобілів, людей тощо

Таблиця 2.2 - Аналіз готових існуючих підходів комп'ютерного зору для розпізнавання образів

Джерело	Рік	Алгоритм/ Модель	Сфера застосування	Короткий опис підходу
[51]	2021	CNN	Система для розумного паркування	Дослідження в цілому спрямоване на розпізнавання зображень для інтелектуального паркування на основі камери за допомогою згорткової нейронної мережі (CNN).
[52]	2020	Google Cloud AutoML Vision	Медична галузь. Рання діагностика карциноми	У статті оцінюється придатність AutoML для ідентифікації інвазивної протокової карциноми (IDC) на повних зображеннях слайдів (WSI). Експериментальна модель ідентифікації IDC створена за допомогою Google Cloud AutoML Vision.
[53]	2020	Google Cloud Vision (GCV) APIs	Галузь кібербезпеки	Було досліджено можливість застосування методу для широкого спектру реальних завдань комп'ютерного зору, включаючи класифікацію зображень, виявлення об'єктів, семантичну сегментацію, виявлення тексту тощо.

Продовження таблиці 2.2 - Аналіз готових існуючих підходів комп'ютерного зору для розпізнавання образів

Джерело	Рік	Алгоритм/ Модель	Сфера застосування	Короткий опис підходу
[55]	2022	pre-trained CNN model with score-level fusion technique	Розпізнавання жестової мови	Було розроблено та протестовано систему розпізнавання американської жестової мови (ASL) в реальному часі з використанням запропонованої методики.
[56]	2021	multiple-fine-tuned CNNs	Медична галузь. Діагностика хвороби Паркінсона	Використання навченої згорткової нейронної мережі для діагностики хвороби Паркінсона через дослідження голосу пацієнта.
[57]	2020	Google Cloud Vision, OpenCV	Розпізнавання паркомісць	Використання технології Google Cloud Vision і попередньо навченої згорткової мережі у створенні кіберфізичної системи розумної парковки.
[58]	2020	OpenCV	Розпізнавання обличчя	Система обліку відвідувачів на основі технології розпізнавання облич з використанням бібліотеки OpenCV.
[67]	2024	YOLOv8 and CNN	Розпізнавання номерного знаку автомобіля	Розпізнавання номерних знаків на розумних парковках, з використанням нейронної мережі YOLOv8.

Кінець таблиці 2.2 - Аналіз готових існуючих підходів комп'ютерного зору для розпізнавання образів

[70]	2023	YOLOv5-v1	Розпізнавання яблук	Модифікація нейронної мережі YOLOv5 для побудови системи візуального оцінювання врожаю яблук.

Рисунок 2.2 - Набір даних цільових зображень для тестування нейронних мереж

Оскільки цільовими об’єктами для розпізнавання на зображеннях є люди, у центрі уваги експерименту було розпізнавання зображень людей і встановлення присутності людей. За результатами тестування моделей [48, 49, 7] складено таблицю з ефективністю розпізнавання образів людей та наявними перевагами та недоліками кожної з моделей. Результати тестувань представлені в таблиці 2.3.

Таблиця 2.3 - Ефективність розпізнавання нейронними мережами образів людей

Критерії	Google Cloud Vision API	YoloV8	PyTorch Faster R-CNN
Випадків, коли кількість людей була розпізнана правильно	1	8	8

Кінець таблиці 2.3 - Ефективність розпізнавання нейронними мережами образів людей

Критерії	Google Cloud Vision API	YoloV8	PyTorch Faster R-CNN
Випадків, коли нейронна мережа розпізнала на зображенні більшу кількість людей, ніж там було насправді	2	1	2
Випадків, коли нейронна мережа розпізнала на зображенні меншу кількість людей, ніж там було насправді	9	3	2
% правильних відповідей	8.3	66.7	66.7
Переваги	Хороша інтеграція з іншими сервісами Google	Висока точність розпізнавання, зручний API	Висока точність розпізнавання

2.3 Підготовка даних до навчання нейронної мережі

Найкращі у світі команди машинного навчання витрачають понад 80% свого часу на вдосконалення навчальних даних [6].

Існують три основні фактори, які безпосередньо впливають на якість навчальних даних. Люди, процеси і інструменти (People, Process, and Tools, PPT) — це три життєво важливі компоненти будь-якого бізнес-процесу. Розглянемо кожну з них в контексті підготовки даних для навчання нейронної мережі.

Люди. Кваліфікація людей, що здійснюють підготовку даних, суттєво впливає на ефективність роботи та кінцеві результати.

Процеси. Це набір дій, який повинен бути виконаний для збору даних, контролю їх якості та маркування.

Інструменти. Це, зокрема, набір програмних засобів, що використовується для автоматизації отримання даних, їх маркування тощо.

Підготовка даних є одним із найважливіших етапів у процесі навчання нейронної мережі. Від якості даних залежить точність та ефективність роботи моделі. Зокрема, на етапі збору даних слід звернути увагу на:

- обсяг даних. Зазвичай, чим більше даних використовується, тим краще може бути навчена модель;
- якість даних. Дані повинні бути точними, повними та репрезентативними для задачі, яка буде вирішуватися;
- різноманіття даних. Важливо, щоб набір даних містив приклади з усіх можливих категорій або класів, які повинна буде розпізнавати модель.

Розробники нейронних мереж часто зазначають свої рекомендації щодо підготовки датасету. Так, зокрема, розробник нейромережі YOLO надає наступні рекомендації [7] щодо підготовки даних:

- кількість зображень на клас: >=1500;
- кількість примірників (об’єктів з мітками) на клас: >=10000;
- фонові зображення (Background images): 0-10% від загальної кількості.

Фонові зображення – це зображення без об’єктів, які додаються до набору даних для зменшення помилкових спрацьовувань (FP). Наприклад, широко розповсюджений датасет COCO [8] містить 1% фонових зображень.

Крім кількісно виражених рекомендацій, існують і інші, зокрема:

- різноманітність зображень. Для випадків реального використання рекомендується використовувати зображення отримані в різний час доби, різні пори року, різну погоду, різне освітлення, різні ракурси, з різні джерел (камер) тощо;

- послідовність маркування. Усі екземпляри всіх класів на всіх зображеннях мають бути позначені. Часткове маркування не працюватиме;
- точність маркування. Мітки повинні щільно охоплювати кожен об’єкт. Між об’єктом і його обмежувальною рамкою не повинно бути проміжків.

Підготовлені дані повинні бути розбиті на 3 групи:

А. Навчальний набір. Набір даних, який використовується для навчання моделі.

Б. Перевірочний (валідаційний) набір. Це набір даних, який використовується для оцінки продуктивності моделі під час навчання.

В. Тестовий набір. Набір даних, який використовується для остаточної оцінки продуктивності моделі після завершення навчання.

Співідношення між цими наборами визначається наступним чином : на валідаційний та тестовий набори, як правило, відводиться по 10-20% від загального обсягу підготовлених даних, на навчальний набір, відповідно, 80-60%. Ці параметри можуть варіюватися, навіть виходити за згадані межі, в залежності від загального розміру підготовлених даних, складності моделі, використаних гіперпараметрів навчання, а також предметної області використання навченої моделі. Так, наприклад, для великих датасетів (що налічують сотні тисяч екземплярів або більше) валідаційна вибірка менш ніж 10% може виявитися досить репрезентативною для підтвердження якісно проведеного навчання. Іншим прикладом може бути валідація даних в предметних областях, де якість навчання моделі є критично важливою (наприклад, в медичних застосунках), такі моделі перевіряються наборами, що сягають 30% загального обсягу даних.

Оцінка датасету - баланс даних, репрезентативність датасету, якість розмітки.

Баланс даних характеризує збалансовану за певними критеріями представленість різних класів об'єктів у датасеті. В даній роботі був тренований всього один клас, тому датасет є збалансованим за визначенням. Але, якщо говорити в цілому, то моделі працюють на основі шаблонів вивчення, і коли класи занадто малі, моделям важко робити прогнози для цих груп. Незбалансовані дані

можна виправити за допомогою різних методів. Зокрема, зменшенням вибірки у великих класах (undersampling), або збільшенням кількості даних у менших класах (oversampling). Є багато способів зробити це, наприклад, використовуючи пакет imbalanced-learn для Python [41].

Якість розмітки. В даній роботі вся розмітка була зроблена (або валідована після розмітки нейромережею) однією людиною (мною). В надточних датасетах, наприклад, робляться паралельні незалежні маркування різними людьми, і потім вони порівнюються [6], але в рамках даної роботи, звичайно, такий підхід не є прийнятним.

Репрезентативність. Цей параметр показує наскільки підготовлений датасет відповідав реальним даним, за яким проводилося подальше тестування. Кроки, вчинені в даній роботі для забезпечення репрезентативності, описані нижче. По-перше, до 40% кліпів з людьми було взято з камер, де попередня наявність людей, виявлених ШІ, не була підтверджена людьми операторами. Ще 40% було взято з інших камер. Крім того, 20% зображень датасету було представлено зображеннями з термокамер (оскільки на них попередньо було зауважено порівняно низький відсоток розпізнавання, було вирішено збільшити їх частку в навчальному датасеті). При відборі кліпи (за згаданими кретеріями) бралися в хронологічному порядку іх появи в системі, з обмеженням кількості використаних кліпів з кожної камери не більше 30. Таким чином, для підготовки датасету з 11152 зображень було використано матеріали з 371 відеокамери. Співставляючи це значення із загальною кількістю камер (1200), і средньою кількістю камер, що надсилають сигнали тривоги щодня (366), можна зробити висновок про мінімально достатню репрезентативність підготовленого датасету.

Досить часто буває, що по мірі того, як нейромережа розвивається або з'являються нові дані, виникає необхідність переглянути та вдосконалити датасет для навчання.

Досі питання підготовки даних для навчання розглядалося в контексті повного циклу створення власного датасету. Насправді, на даний час у відкритому доступі є багато ресурсів з найрізноманітнішими датасетами. В контексті,

наприклад, датасетів із зображеннями людей можна використовувати [8, 45, 46, 47].

Нижче в таблиці 2.4 наведено порівняння переваг та недоліків у використанні відкритих набрів даних, та створення власного набору даних. Найкращий підхід залежить від конкретних потреб і ресурсів проекту.

Таблиця 2.4 - Переваги та недоліки використання відкритих наборів даних порівняно зі створенням власного набору даних

Фактор	Використання відкритих наборів даних	Створення власного набору даних
Час та ресурси	Заощаджує час і ресурси	Вимагає значних витрат часу та ресурсів
Різноманітність і якість	Зазвичай пропонує ширший спектр даних і потенційно вищу якість	Забезпечує контроль над різноманітністю та якістю даних
Економічна ефективність	Зазвичай безкоштовні або дуже доступні	Може бути дорогим, залежно від масштабу та складності
Відтворюваність	Дозволяє іншим відтворити результати ваших досліджень або проектів	Гарантує відтворюваність результатів
Упередженість даних	Може відображати властиву упередженість, присутню під час збору, що може вплинути на продуктивність вашої моделі, якщо її не вирішити	Дозволяє мінімізувати упередженість даних

Кінець таблиці 2.4 - Переваги та недоліки використання відкритих наборів даних порівняно зі створенням власного набору даних

Фактор	Використання відкритих наборів даних	Створення власного набору даних
Обмежений контроль	Ви не маєте контролю над процесом збору даних або стандартами маркування відкритих наборів даних	Забезпечує повний контроль над процесом збору даних та стандартами маркування
Наявність даних	Знайти набір даних, який ідеально відповідає вашим конкретним потребам, може бути складно	Гарантує наявність даних, що відповідають вашим конкретним потребам
Фактор	Використання відкритих наборів даних	Створення власного набору даних
Проблеми конфіденційності	Деякі відкриті набори даних можуть мати обмеження конфіденційності, що потребує ретельного розгляду етичних наслідків перед використанням	Гарантує відповідність вимогам конфіденційності
Проблеми якості даних	Якість даних може відрізнятися в відкритих наборах даних. Можливо, вам доведеться витратити час на очищення або попередню обробку даних, перш ніж використовувати їх у своєму проекті	Забезпечує високу якість даних

У даному розділі роботи слід згадати також про інструменти, які можуть бути використані для підготовки даних до навчання нейронної мережі. Їх перелік і

сфери застосування наведено нижче в таблиці 2.5. Було б не коректно стверджувати, що для кожного етапу підготовки існує єдиний найкращий інструмент. Підготовка даних часто передбачає поєднання методів та інструментів в залежності від конкретних даних, вибраного фреймворку, типу нейронної мережі тощо.

Таблиця 2.5.- Інструменти для підготовки даних до навчання нейронної мережі

Назва інструменту	Опис
Етап: збір даних	
Beautiful Soup (Python)	Інструмент веб-парсингу для отримання даних з веб-сайтів.
Scrapy (Python)	Фреймворк веб-парсингу.
Системи керування базами даних (СКБД)	СКБД, як MySQL, PostgreSQL або Oracle, для керування даними та доступу до них у реляційних базах даних.
API (застосункові програмні інтерфейси)	Програмні інтерфейси для доступу до даних з різних сервісів.
Етап: очищення даних	
Pandas (Python)	Потужна бібліотека для маніпулювання даними, їх очищення та аналізу. Забезпечує функції для роботи з відсутніми значеннями, дослідження даних тощо.
NumPy (Python)	Фундаментальна бібліотека для числових обчислень. Корисна для завдань маніпулювання даними разом з Pandas.

Продовження таблиці 2.5.- Інструменти для підготовки даних до навчання нейронної мережі

Назва інструменту	Опис
scikit-learn (Python)	Комплексна бібліотека машинного навчання, що містить інструменти для попередньої обробки даних, інженерії ознак та заповнення відсутніх значень.
OpenRefine (з відкритим кодом)	Колишній Google Refine. Окремий інструмент від для очищення та перетворення даних. Корисний для очищення великих наборів даних зі складним форматуванням або невідповідностями.
Етап: перетворення даних	
scikit-learn (Python)	Забезпечує функції для нормалізації, стандартизації та різних методів кодування для категоріальних даних.
Pandas (Python)	Надає функціональні можливості для завдань перетворення даних, таких як перетворення типів даних або створення нових ознак.
Етап: попередня обробка даних	
scikit-learn (Python)	Забезпечує функції для розподілу даних на навчальні, валідаційні та тестові набори.
Pandas (Python)	Надає функції для рандомізації даних.
TensorFlow (Python) або PyTorch (Python)	Фреймворки глибокого навчання, які зазвичай обробляють пакетні дані під час навчання для ефективності використання пам'яті.
Етап: візуалізація даних	
Matplotlib (Python)	Універсальна бібліотека для створення різних діаграм та графіків для дослідження розподілів даних, зв'язків та виявлення закономірностей.

Кінець таблиці 2.5.- Інструменти для підготовки даних до навчання нейронної мережі

Назва інструменту	Опис
Seaborn (Python)	Побудована на Matplotlib, пропонує високорівневі функції для створення інформативних та візуально привабливих візуалізацій для дослідження даних.
TensorBoard (часто використовується з TensorFlow)	Набір інструментів візуалізації для моніторингу ходу навчання, візуалізації показників ефективності моделі та перегляду розподілів ваг всередині нейронної мережі.

2.4 Налаштування моделей нейронних мереж для тренування

Це, як правило, ітеративний процес, спрямований на оптимізацію показників продуктивності машинного навчання, таких як , зокрема, акуратність, точність і відгук. У більшості випадків хороші результати навчання можна отримати без змін у моделях або налаштуваннях, за умови, що набір даних достатньо великий і добре маркований [7]. У будь-якому випадку, перш ніж розглядати будь-які зміни, рекомендовано спочатку провести тренування з усіма налаштуваннями за замовчуванням. Це допомагає визначити базову продуктивність і визначити напрямки для подальшого покращення. Розглянемо нижче деякі важливі фактори у налаштуваннях навчання моделей для, зокрема:

- вибір моделі нейронної мережі;
- кількість епох навчання;
- розмір зображення;
- розмір пакета;
- гіперпараметри.

Нижче розглянемо кожен з цих факторів трохи детальніше.

Вибір моделі нейронної мережі. Лінійка моделей нейронних мереж останньої (8-ї на момент початку роботи над даним проектом) версії сімейства

YOLO, YOLOv8, представлена 5 моделями. Рисунок 2.3 дає уявлення про назви моделей та їх базові параметри. На рисунку використано інформацію від виробника (компанії Ultralytics) щодо попереднього покоління нейронних мереж YOLOv5, що є прямим попередником версії 8 однак, як показали інші дослідження [16], співідношення характеристик для різних варіантів (від Nano до XLarge) залишилися приблизно такими ж, тому можуть бути використані для порівняння.

Зазвичай більші моделі (Large та XLarge) демонструють кращу продуктивність, але при цьому вони є більш вимогливими до обсягів пам`яті CUDA при тренуванні, і працюють повільніше при розпізнаванні об'єктів, у порівнянні з меншими моделями. На рисунку 2.4 проілюстровано співідношення метрики якості мАР@50-95 для результатів тренувань моделей різних поколінь YOLO, в порівнянні із складністю цих моделей. На прикладі навчального набору COCO [8] було наочно продемонстровано, що кращі результати дають моделі із більшою кількістю параметрів.

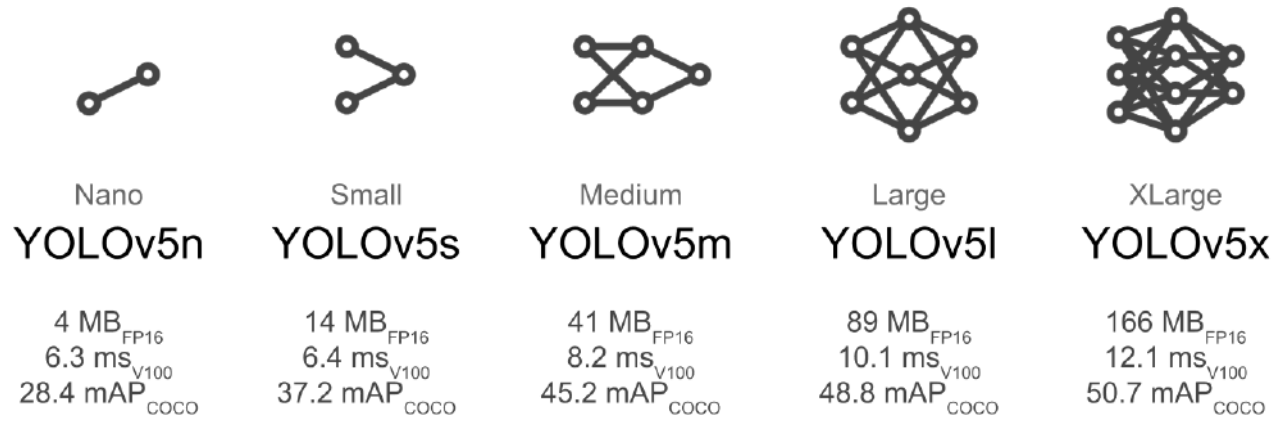


Рисунок 2.3 - Різновиди моделей YOLO.

Кількість епох навчання. Більша кількість епох може призвести до кращої точності, але також може спричинити перенавчання. Розробник YOLOv8 рекомендує починати з 300 епох, і, якщо не спостерігається перенавчання, збільшувати їх кількість до 600, і навіть до 1200 [7]. Слід зазначити, що

функціонал YOLOv8 надає можливість відслідковувати появу перенавчання (зокрема, використовуючи параметр навчання patience) в процесі тренування, і зупиняти тренування достроково в таких випадках.

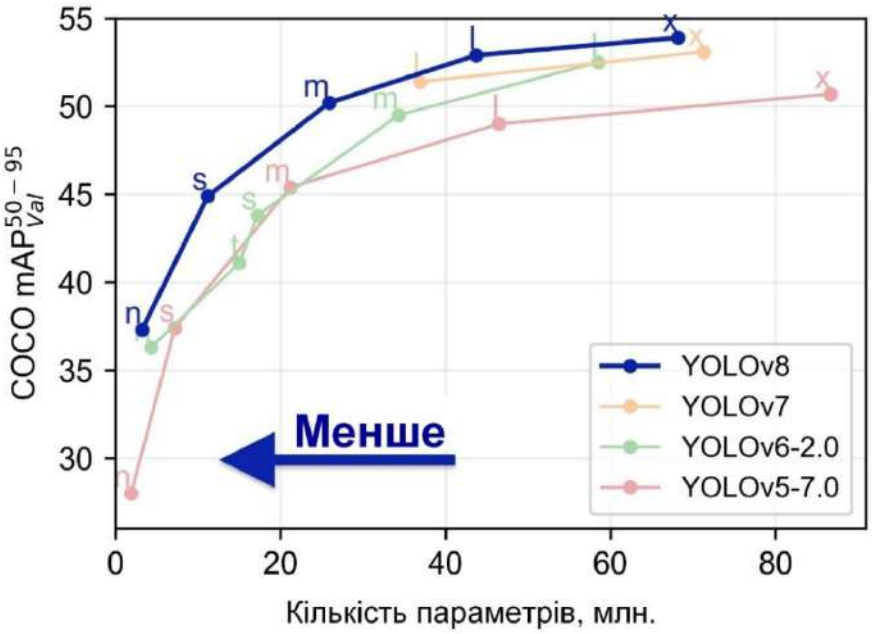


Рисунок 2.4 - Співвідношення метрики якості мАР@50-95 із складністю моделей.

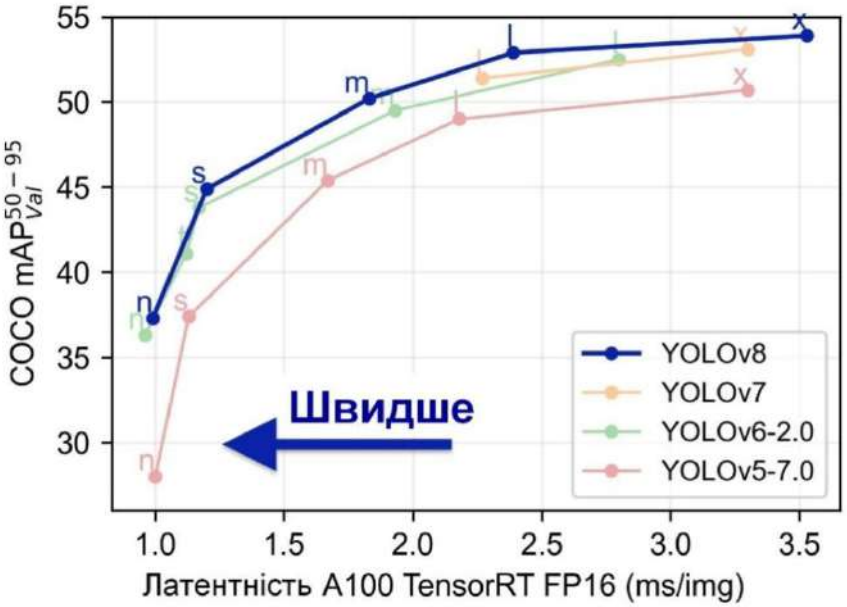


Рисунок 2.5 - Співвідношення метрики якості мАР@50-95 із латентністю.

Розмір зображення. Найкращі результати розпізнавання будуть спостерігатися здебільшого у випадках, коли розміри зображення, що буде розпізнаватися, буде таким же або близьким до розмірів зображень, на яких проводилися тренування. Але, наприклад, якщо зображення для розпізнавання містить багато дрібних деталей, для навчання краще використати зображення більшого розміру. За замовчуванням, готові моделі YOLO пропонуються користувачам натренованими на датасетах COCO розміром 640 пікселів, але налаштування параметрів навчання дозволяють легко визначати розміри зображень для навчання за допомогою опції --img , наприклад, --img 1280.

Розмір пакета. Розмір пакета (batch size) - це кількість зразків даних, які обробляються алгоритмом одночасно під час навчання або тестування. Іншими словами, в контексті навчання моделі це кількість прикладів (зображень в нашому випадку), які обробляються разом перед тим, як модель оновить свої ваги. Розглянемо вплив розміру пакета на навчання.

По перше, розмір пакета впливає на швидкість навчання. Більший розмір пакета може підвищити швидкість навчання, оскільки дозволяє обробляти більше зразків за одну епоху. Однак, занадто великий розмір пакета може призвести до затримки в оновленні ваг моделі, що може зменшити швидкість навчання.

По друге, розмір пакета впливає на градієнти. Більші пакети можуть призвести до більш гладких градієнтів, що може допомогти алгоритму сходитися до оптимального рішення швидше. З іншого боку, занадто великий розмір пакету може призвести до того, що модель фокусується на специфічних особливостях більшої групи зразків, замість того, щоб узагальнювати на основі всієї вибірки, що, в свою чергу може призвести до погіршення здатності моделі до узагальнення нових даних.

По третє, не слід ігнорувати зв’язок розміру пакета з обчислювальними ресурсами. Більший розмір пакетів вимагає більше пам'яті (RAM або GPU-пам'яті) для обробки даних, що може бути обмеженням для великих моделей або обробки великих датасетів. На рисунку 2.6 наведено приклад залежності часу розпізнавання об’єктів від розміру пакетів (для нейромережі YOLOv8m, з

використанням обладнання Nvidia Jetson AGX Xavier) [28]. Із збільшенням розміру пакету спостерігається покращення швидкості розпізнавання до певного порогового значення, що можна пояснити апаратними обмеженнями обладнання.

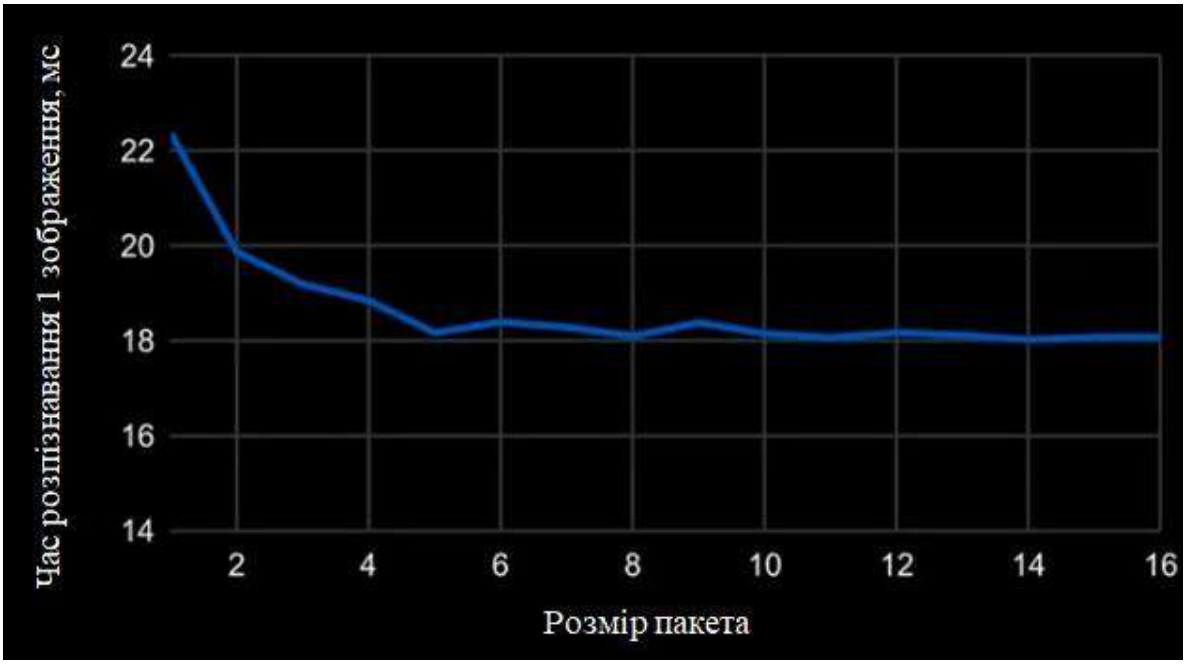


Рисунок 2.6 - Залежність часу розпізнавання об’єктів від розміру пакетів

Важливо зазначити, що не існує універсального оптимального розміру пакета. Найкращий розмір пакета для однієї задачі може відрізнятися від найкращого розміру пакета для іншої задачі. Оптимальним підходом може бути початок навчання з середнього значення; потім можна експериментувати з різними розмірами, при цьому необхідно стежити за стабільністю навчання та продуктивністю.

Гіперпараметри (hyperparameters). Це - високорівневі, структурні налаштування алгоритму. Вони визначаються перед початком навчання і залишаються незмінними під час всього процесу. Прикладами гіперпараметрів можуть бути: швидкість навчання, кількість шарів, тип функції активації, тип оптимізатора тощо.

2.5 Висновки

Нейронні мережі (НМ) стають все більш популярним інструментом для обробки похибок у СКЗ. Їх здатність до навчання на великих наборах даних та виявлення складних закономірностей робить їх ідеально підходящими для вирішення такого роду проблем. Ключовими перевагами використання НМ для обробки похибок у СКЗ, у порівнянні з традиційні методи обробки зображень, є висока точність, гнучкість і широкі можливісті автоматизації. Кілька конкретних прикладів застосування НМ для обробки похибок у КЗ наведено нижче.

Відновлення зображень. НМ можна використовувати для відновлення пошкоджених або нечітких зображень, заповнюючи відсутні або пошкоджені пікселі.

Класифікація зображень. НМ можна використовувати для класифікації зображень навіть за наявності шуму або оклюзії.

Виявлення об'єктів. НМ можна використовувати для виявлення об'єктів на зображеннях, навіть якщо вони частково приховані або знаходяться в складних умовах освітлення.

Сегментація зображень. НМ можна використовувати для сегментації зображень на різні області, навіть якщо межі між цими областями нечіткі або нерівні.

Трекінг об'єктів. НМ можуть використовуватися для відстеження об'єктів у відео. Це може бути корисно для таких завдань, як відстеження трафіку, нагляд за безпекою та автономне водіння.

НМ є потужним інструментом, який може значно покращити продуктивність систем КЗ. Завдяки постійному розвитку вони , ймовірно, відіграватимуть ще більшу роль в обробці похибок в СКЗ в майбутньому.

3 МЕТОД ТА АЛГОРИТМ ОБРОБКИ ПОХИБОК ОТРИМАНИХ ВІДЕОЗОБРАЖЕНЬ НЕЙРОННОЮ МЕРЕЖЕЮ

3.1 Метод навчання нейронної мережі для обробки похибок відеозображень

Процес навчання штучної нейронної мережі — це метод, математична логіка або алгоритм, які покращують продуктивність мережі та/або час навчання. Процес навчання є одним із факторів, який вирішує, наскільки швидко та точно може бути розроблена штучна мережа. Залежно від процесу розробки мережі виділяють три основні парадигми машинного навчання. Їх перелік та можливі сфери застосування представлені на рисунку 3.1.

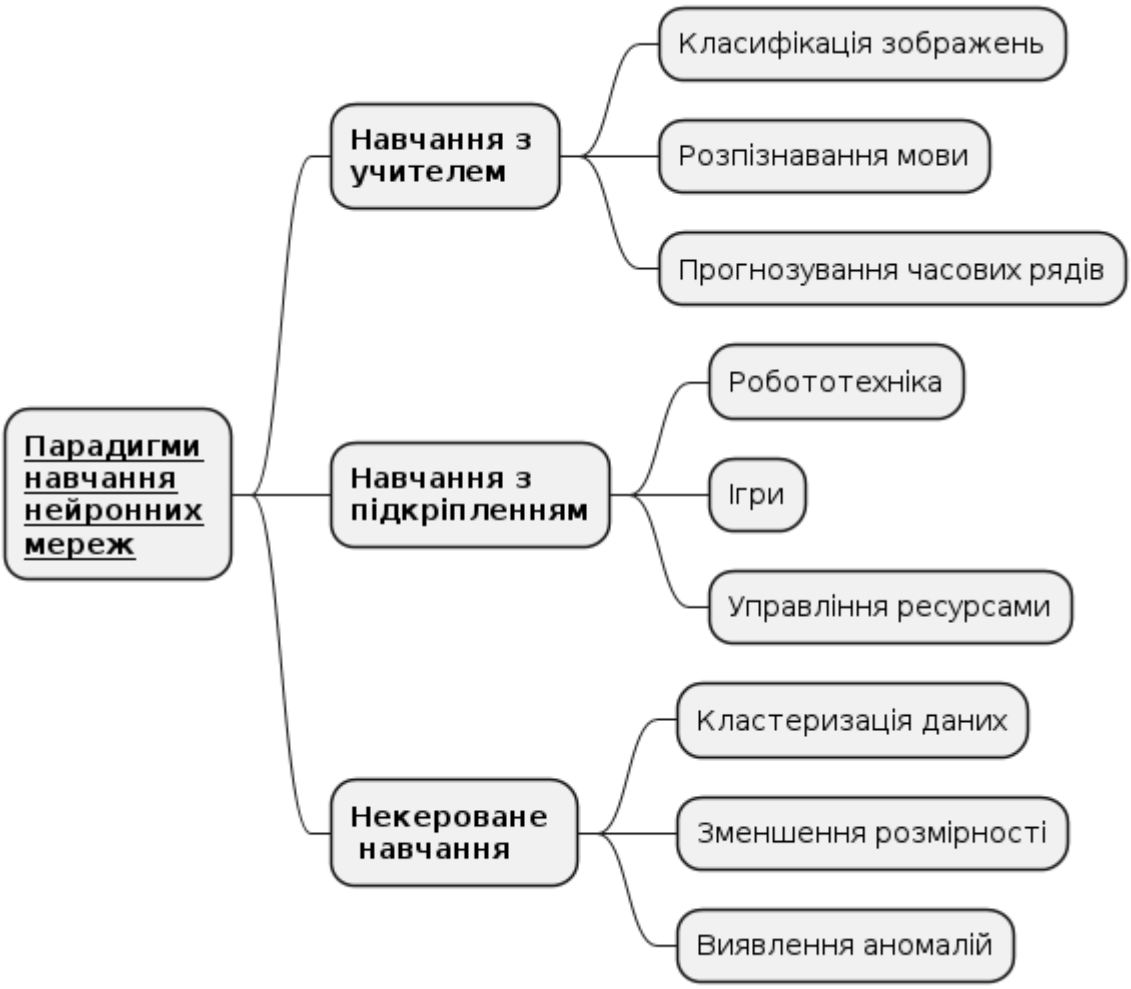


Рисунок 3.1 - Основні парадигми машинного навчання.

Навчання з учителем (Supervised Learning). При даному підході мережа навчається на наборі даних, який містить як вхідні дані, так і бажані вихідні дані. Мета навчання – мінімізувати функцію втрат (loss function) між прогнозованими та фактичними значеннями.

Некероване навчання (Unsupervised Learning). При цьому підході мережа навчається на немаркованих даних, тобто на даних, які не мають чітко визначених міток. Мережа сама знаходить закономірності та структури в даних.

Навчання з підкріпленням (Reinforcement Learning). Є проміжним варіантом двох попередніх парадигм . Замість «вчителя» в схему навчання вводиться блок «критика» , який відслідковує реакцію середовища на вхідний сигнал і , спираючись на неї, визначає евристичну похибку, яку покладено в процес навчання мережі.

Нейронні мережі YOLO використовують підхід «навчання з учителем». Розпізнавання об’єктів здійснювалося покадрово. Діапазон FPS на різних камерах варіювався від 1 до 100 кадрів в секунду. Тобто для 10-секундного відео при fps=100 і часу обробки одного кадру t=170-175 мс час обробки відеопотоку збільшується до 175 секунд. Було очевидно, що для задачі ідентифікації відеокліпу немає потреби проводити розпізнавання всіх підряд фреймів. Експериментальним шляхом було підібрано мінімальне значення fps = 12 у відеопотоці, при якому модель на основі штучної мережі чітко фіксувала об’єкти, зняті тепловізором. Тому за формулою (1) розраховували параметр N. N - інтервал у вхідному відео між кадрами, які слід скопіювати в цільове відео, щоб отримати бажану кількість кадрів в секунду. Означає, наприклад: якщо частота кадрів в секунду у вхідному відео дорівнює 100, а бажана частота кадрів у секунду становить 25 - у цьому випадку кожен 4-й (100/25) кадр вхідного відео повинен бути оброблений і вставлений в цільове відео - 4-й, 8-й, 12-й..., інші кадри слід пропустити.

$$N = round\ (fps(source)/fps(target)), \tag{3.1}$$

де $fps(source)$ - fps вхідного відео;

$fps(target)$ - fps вхідного відео.

Одним з питань, яке необхідно було вирішити під час роботи над КФС, було питання ідентифікації відеокліпів (належності їх до одного з класів об’єктів, які цікавили кінцевого користувача системи). Складність полягала в тому, що нейронна мережа здійснює розпізнавання пофреймово, кожен фрейм міг містити велику кількість об’єктів одного або багатьох класів (а міг не містити жодного). При цьому набір класів нейромережі (попередньо тренованої на наборі COCO) містив 80 класів, набір класів, що цікавили користувача, містив всього 5 класів. Класи об’єктів користувача, їх відповідність класам моделі COCO та пріоритет їх виявлення у відео наведені в таблиці 3.1.

Таблиця 3.1 - Відповідність класів об’єктів користувача класам моделі COCO

Пріоритет	Клас замовника	Класи моделі
1	Person	'person'
2	Vehicle	'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat'
3	Animal	'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe'
4	Light	'traffic light'
5	NIL	жоден з класів наведених вище не був виявлений

Роботу алгоритму в частині виявлення об’єктів класів клієнта в кліпі можна звести до кількох моментів:

1. Здійснювався пошук об’єктів у кожному фреймі відеокліпу. Результати пошуку для кожного кліпу зберігалися, а статистика записувалася в таблицю. Зразок статистики розпізнавання наведено в таблиці 3.2.

2. Віднесення кліпу до певного класу здійснювалося від вищого 1 до найнижчого 5 пріоритету. Тобто, наприклад, якщо в кліпі був виявлений хоча б один об'єкт пріоритету 1 (Людина), кліп належав до цього класу незалежно від кількості об'єктів нижчих класів. Так само, якщо не виявлено об'єктів пріоритету 1, враховується наявність об'єктів пріоритету 2 тощо. Схему алгоритму наведено в додатку Г.

Вибраний спосіб реалізації кіберфізичної системи як окремого (не вбудованого) апаратно-технічного комплексу забезпечив достатню гнучкість у виборі обладнання як безпосередньо для корекції похибок , так і для навчання нейронної мережі.

Таблиця 3.2 Зразок статистики розпізнавання об’єктів у кліпі.

Номер фрейму	Код об’єкту COCO	Назва об’єкту	Confidence
1	0	person	0.5774
1	13	bench	0.4359
2	0	person	0.5949
2	13	bench	0.4844
3	0	person	0.5488
3	13	bench	0.4548
4	13	bench	0.5063
4	0	person	0.4896
4	0	person	0.2570

Даний підхід показав усі свої переваги під час виконання роботи. Так, наприклад, спочатку в якості апаратної платформи для пост-обробки зображень

використовувався пристрій “все в одному” Nvidia Jetson Orin Nano 8Gb, згодом він був замінений на більш потужний сервер з Nvidia RTX 4070 Super 12Gb в якості GPU. Ця зміна сприяла зменшенню часу обробки фрейму із 170 до 14 мс. При цьому заміна обладнання на стороні КФСКЗ не спричинила жодних перерв в роботі основної системи управління відео-даними підприємства.

Однією із складових КФС є апаратна платформа, на якій здійснювалося навчання нейронних мереж. З певних технічних причин (таких, зокрема, як вимогливість нейромережі до ресурсів під час навчання) було вибрано Google Colab. Під час навчання моделі дані завантажувалися з Google Drive. Слід зауважити, що для проведення навчання виявилося недостатньо ресурсів , що надавалися в межах безкоштовного пакету Colab. Для Так, для навчання моделі з вагами yolo8s (120 епох) було використано пакет Colab Pro, а для моделі yolo8m (300 епох) можливостей пакету виявилося недостатньо, довелося підключати пакет Colab Pro+. Ще одна деталь, що може бути корисною : для завантаження підготовленого на локальному комп’ютері датасету на Google Drive краще використати не веб- інтерфейс останнього, а встановити відповідний застосунок від виробника; це забезпечить доступ до Google Drive через файловий провідникяк до докального диску, і збільшить в рази швидкість завантаження файлів (мова йде про сотні тисяч мегабайт, а у нашому випадку – про понад 2 Гб інформіції.

Але навіть в межах пакету Colab Pro+ довелося відчути деякі обмеження . Зокрема, час збереження сесії в режимі простою (idle time) для всіх згаданих пакетів не регулюється, і обмежений 60 хвилинами. Для подібних завдань в майбутньому я рекомендував би використовувати одразу пакет Colab Enterprise , який пропонує більше можливостей в налаштуваннях (наприклад, регулювання idle time в межах 24 годин).

Під час тренування моделі з вагами yolo8m було виявлено, що можливостей відеокарти Nvidia Tesla V100 16Gb, що використовувалася для тренування попередньої моделі (з вагами yolo8s), недостатньо, робоче середовище Colab повідомляло про недостатність відео-пам’яті. Тому довелося задіяти більш

потужну карту, Nvidia Tesla A100 40Gb. Характеристики пристроїв наведено в таблиці 3.3

Таблиця 3.3 - Порівняння пристоїв, що використовувалися в КФС.

Пристрій	Пікова продук-тивність ШІ	К-ть тензорних ядер	Дартість в Україні, тис. грн	Зовнішній вигляд
Nvidia Tesla V100 16Gb	120 TFLOPS	640	120	
Nvidia Tesla A100 40Gb	312 TFLOPS	Не визначено виробником	277	
Nvidia RTX 4070 Super 12Gb	Не визначено виробником	224	28	
Nvidia Jetson Orin Nano 8Gb	40 TOPs	48	28	

3.2 Алгоритм навчання нейронної мережі для обробки похибок відеозображень

Навчання нейронної мережі – як правило, процес ітеративний. Дана робота не є винятком. Загальний алготитм роботи наведено нижче на рисунку 3.3. Одразу

хочу заважити, що значна частина етапів роботи вже згадувалася в попередніх розділах, тому нижче будуть згадані деталі, про які не говорилося.

Підготовка датасету. Етапи підготовки даних для виконання даної роботи наведені на рисунку 3.4. Створення датасету було частково автоматизовано в процесі роботи: була написана програма на Python з використанням існуючої “коробочної” моделі YOLOv8 (yolo8x.pt). Програма розбивала відео з камер спостереження на окремі зображення (по 1 фрейму на секунду), у випадку виявлення на зображеннях людей - створювала текстові файли міток.

Всі отримані зображення переглядалися, при виявленні помилок розпізнавання файли міток створювалися для таких зображень вручну на локальному комп’ютері, за допомогою програми Labelimg. Підготовлені зображення та файли міток були структуровані у відповідності до вимог. Структура підготовлених даних представлена на рисунку 3.6.

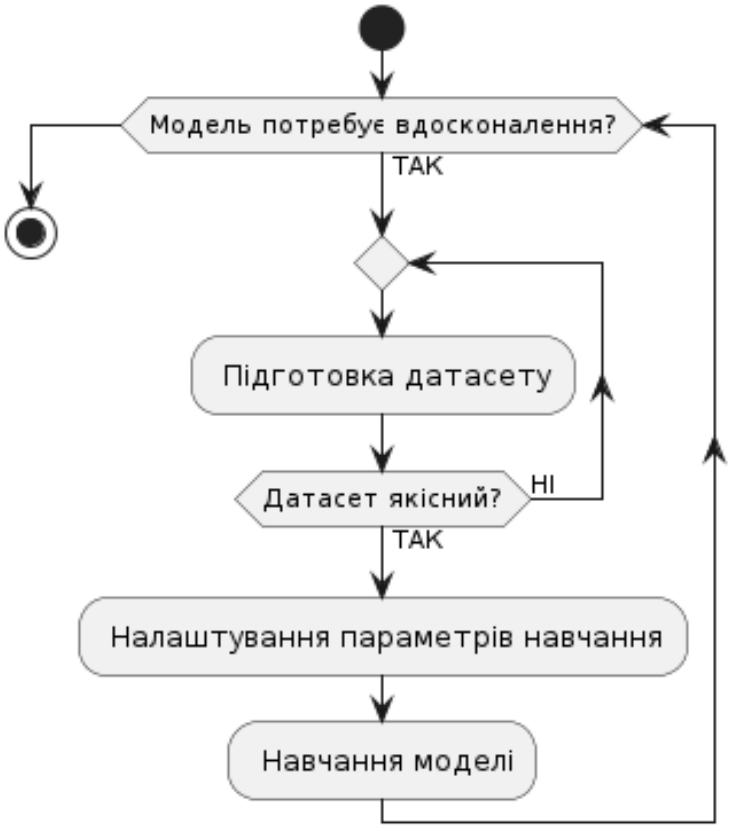


Рисунок 3.3 - Алгоритм навчання нейронної мережі для обробки похибок

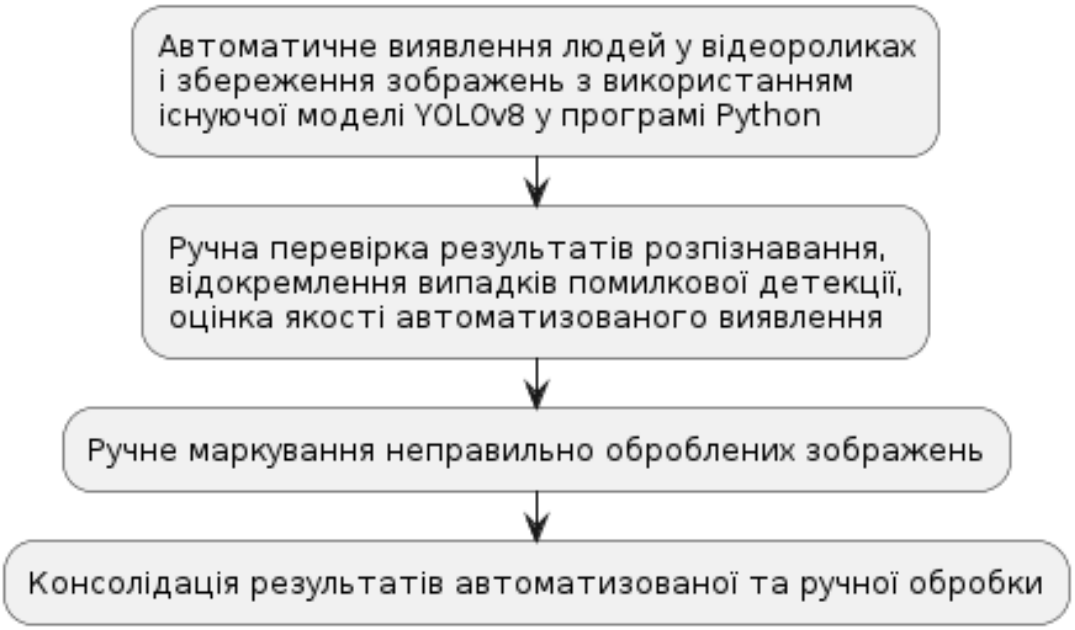


Рисунок 3.4 - Етапи підготовки даних для навчання нейронної мережі

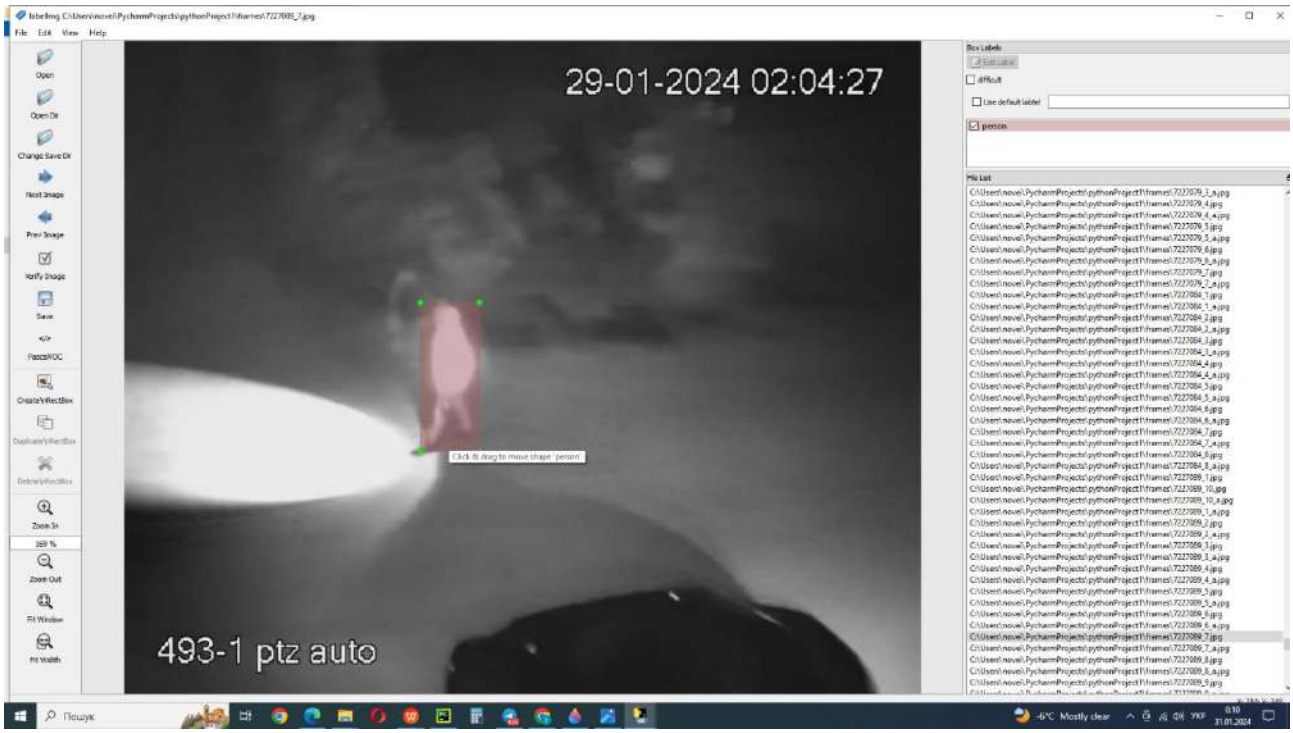


Рисунок 3.5 - Зовнішній вигляд програми Labelimg

Для кожного зображення, що використовувалося для навчання мережі, було створено поставлено у відповідність текстовий файл міток. Приклад змісту

заповненого файлу міток у форматі yolo, що визначають клас об'єктів та їх позицію на зображенні для навчання моделі, наведено на рисунку 3.7

```
root/
├── images/
│   ├── train/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ... (more training images)
│   └── val/
│       ├── image3.jpg
│       ├── image4.jpg
│       └── ... (more validation images)
└── labels/
    ├── train/
    │   ├── image1.txt
    │   ├── image2.txt
    │   └── ... (more training label files)
    └── val/
        ├── image3.txt
        ├── image4.txt
        └── ... (more validation label files)
```

Рисунок 3.6 - Структура датасету зображень для навчання нейромережі YOLOv8

API нейромережі YOLOv8 демонструє простоту і зручність налаштуваннь незалежно від робочого середовища (Windows, Linux, Colab тощо). Приклад конфігураційного файлу для навчання нейромережі в середовищі Google Colab наведено на рисунку 3.8

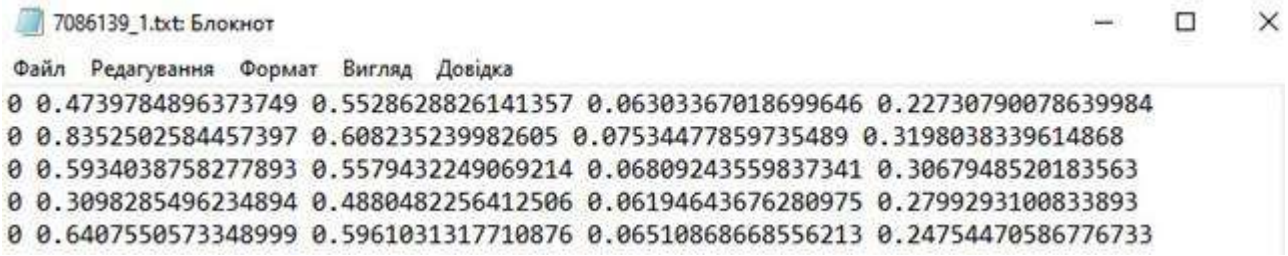


Рисунок 3.7 - Приклад файлу міток у форматі YOLO.

На рисунку 3.9 наведено зразок скрипту на Python, що запускає безпосередньо навчання моделі на 300 епох з вагами yolov8m. Звертаю увагу, що за замовчуванням всі моделі yolov8 навчаються на зображеннях розміром 640 пікселів.

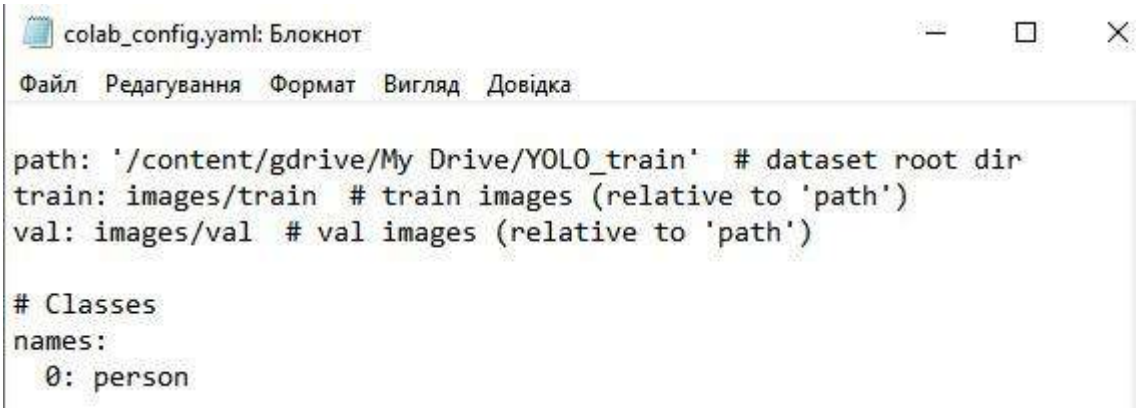


Рисунок 3.8 - Конфігураційний файл для навчання нейромережі yolo.

Але, зважаючи на те, що розмір відео з камер може бути різним,аж до 1920 - було прийнято рішення в параметрах навчання використати розмір 1280 (2*640), для забезпечення кращої деталізації.

```
import os
from ultralytics import YOLO
# Load a model
model = YOLO("yolov8m.yaml")  # build a new model from scratch
# Use the model
results = model.train(data=os.path.join(ROOT_DIR, "colab_config.yaml"), epochs=300, patience=30, imgsz=1280
```

Рисунок 3.9 - Зразок скрипту на Python, що запускає навчання моделі на 300 епох з вагами yolov8m.

3.3 Математична модель

Алгоритм YOLO (You Look Only Once) є одним із найпопулярніших методів виявлення об’єктів у режимі реального часу, оскільки він забезпечує високу точність у більшості завдань обробки в режимі реального часу [36]. Розглянемо детальніше особливості його роботи та математичну модель.

Метою алгоритму YOLO є передбачення класу об’єкта та обмежувальної рамки (bounding box), яка визначає розташування об’єкта на вхідному зображенні. Алогритм розпізнає кожну обмежувальну рамку використовуючи чотири числа:

- центр обмежувальної рамки (b_x, b_y);
- ширина рамки (b_w);
- висота рамки (b_h).

Крім цього, YOLO передбачає відповідне значення C для прогнозованого класу, а також ймовірність цього прогнозу (P_c).

Розглянемо для прикладу зображення з двома обмежувальними рамками, що представляють кота та собаку (рисунок 3.10). Перший крок, який робить алгоритм YOLO - розділення зображення на сітку. Наприклад, на сітку 3×3, як показано нижче.

Завдяки наявності сітки можна виявляти один об’єкт на клітинку сітки замість одного об’єкта на зображення. Для кожної клітинки сітки ми можемо закодувати вектор, який описуватиме її. Наприклад, перша клітинка зліва вгорі не містить жодного об’єкта, і ми описуємо її як:

$$C_{1,1} = (P_c, B_x, B_y, B_w, B_h, C_1, C_2) = (0, ?, ?, ?, ?, ?, ?), \quad (3.1)$$

де (P_c)– ймовірність класу об’єкта;

B_x, B_y – координати центру обмежувальної рамки відносно комірки;

B_w, B_h – ширина та висота обмежувальної рамки відносно всього зображення;

C_1, C_2 - клас який представляє обмежувальна рамка (C_1 для кота і C_2 для собаки відповідно).

Вектор ($C_{1,1}$) складається із символів «?», оскільки якщо перший компонент (P_c) дорівнює нулю, то решта компонентів можуть мати будь-які значення, і вони не беруться до уваги.

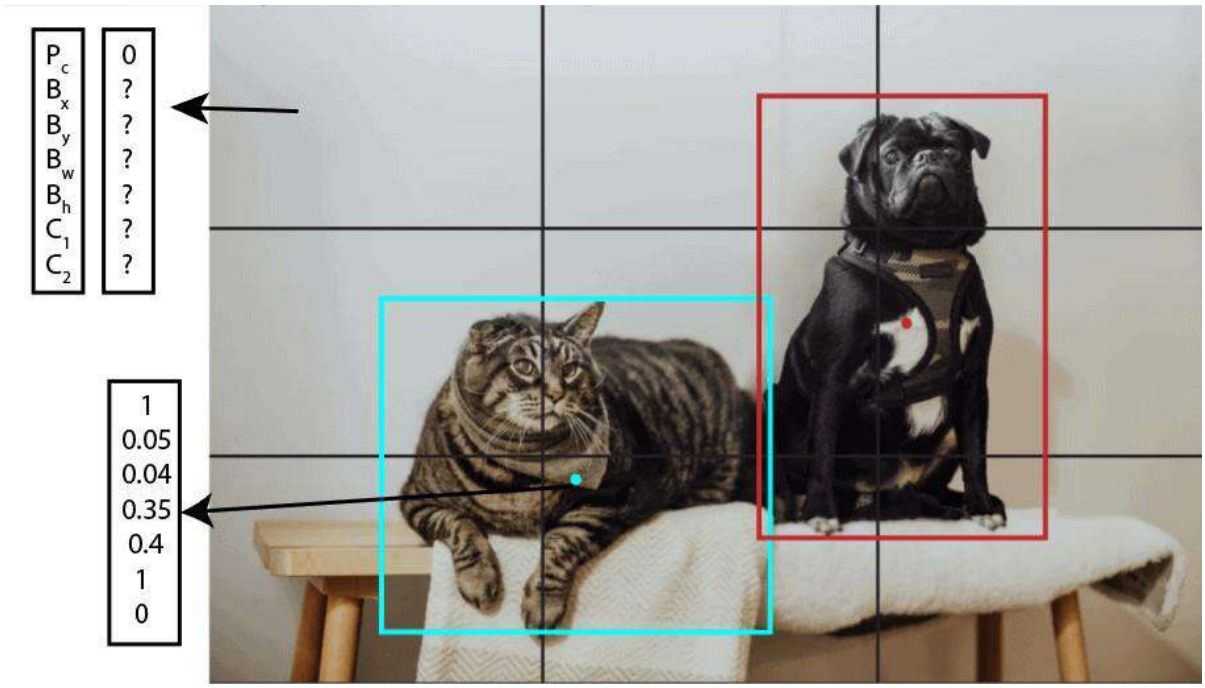


Рисунок 3.10 - Приклад поділення зображення сіткою у алгоритмі YOLO

По аналогії з (3.1), клітинка, що містить центр синьої обмежувальної рамки з котом, може бути представлена вектором

$$C_{3,2}= (1, 0.05, 0.04, 0.35, 0.4, 1, 0). \quad (3.2)$$

Отже, якщо кожна комірка сітки буде визначена одним вектором, все зображення представлятиметься дев’ятьма векторами з розміром 7 , або тензором 3x3x7. Це означає, що в нашому наборі даних кожен зразок зображення позначено одним тензором 3x3x7, і, використовуючи цей набір даних, ми можемо створити навчальний і тестовий набір і навчити згорткову мережу. Використовуючи згорткову мережу, YOLO здатен визначати всі об’єкти за один прохід, і це є причиною його повної назви «You Only Look Once».

Одна з проблем, які можуть трапитися при описаному вище підході, полягає в тому, що згаданий алгоритм передбачає можливість наявності кількох обмежувальних рамок для одного класу. (рисунок 3.11). Можна було б вибрати лише одну обмежувальну рамку на клас – ту, що має найвищу ймовірність, але це не працюватиме, якщо на зображенні є більше об’єктів одного класу (наприклад,

кілька котів). Тому в таких випадках використовується алгоритм немаксимального пригнічення (non-max suppression algorithm, NMS).

Спочатку вибираємо рамку (bounding box) з максимальною ймовірністю для заданого класу. Після цього порівнюємо дану рамку з усіма іншими рамками цього ж класу за допомогою перетину через об'єднання (Intersection Over Union, IoU) . Зазвичай цей показник також відомий як індекс Жаккара (Jaccard index), але у сфері комп'ютерного зору і машинного навчання використовується назва IoU. Нижче наведена його формула.

$$\text{IoU} = \frac{\text{Площа перекриття B1 і B2}}{\text{Площа об'єднання B1 і B2}}, \quad (3.3)$$

де де B_1 , B_2 – дві обмежувальні рамки.

Рисунок 3.11 - Приклад наявність кількох обмежувальних рамок для одного класу

Дуже вдало сутність показника IoU можна продемонструвати графічно (див. нижче рисунок 3.12):

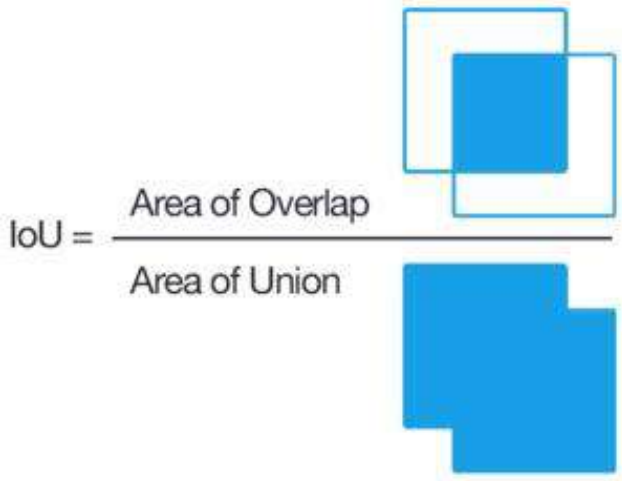


Рисунок 3.12 - Графічне визначення показника IoU

Отже, необхідно розрахувати значення IoU для вибраної рамки (з найвищою ймовірністю для даного класу), з кожною з інших рамок для даного класу. І, якщо IoU вище попередньо визначеного порогового значення (зазвичай,

це 0.5), то така рамка пригнічується (виключається з розгляду). Такий підхід пояснюється тим, що якщо у пари полів високе значення IoU, то, ймовірно, вони вказують на той самий об’єкт, і поле з меншою ймовірністю повинно бути виключено з розгляду. Результат роботи алгоритму показано на рисунку 3.13.

Вище розглядалася ситуація, що лише один об’єкт може бути розпізнаний для кожної клітинки сітки. Насправді, таких об’єктів може бути багато. Так, наприклад, для випадку розпізнавання двох об’єктів (наявності двох рамок) на клітинку сітки вектор для клітинки $C_{1,1}$ матиме вигляд:

$$C_{1,1} = (P_c^1, B_x^1, B_y^1, B_w^1, B_h^1, P_c^2, B_x^2, B_y^2, B_w^2, B_h^2, C_1, C_2), \quad (3.4)$$

де всі визначення аналогічні визначенням у формулі 3.1, а верхній індекс вказує на індекс обмежувальної рамки.

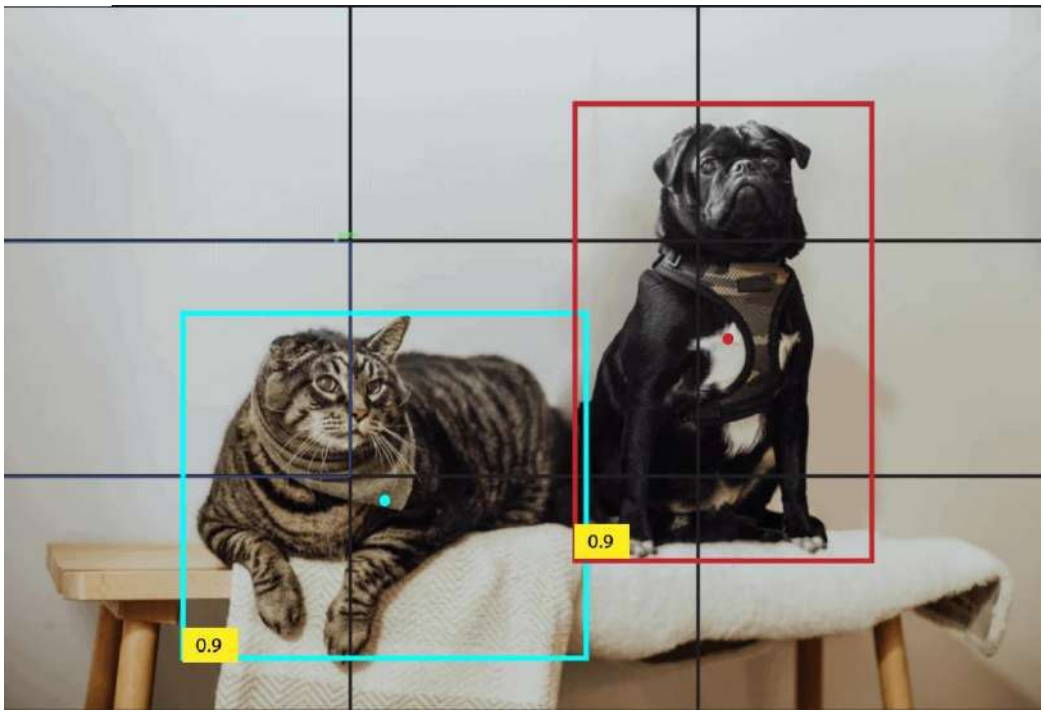


Рисунок 3.13 - Результат роботи NMS алгоритму

Подібним чином, замість прогнозування лише двох класів, можна визначити стільки класів, скільки потрібно. Також можна збільшити розмір сітки. Загалом, можна описати кожну клітинку за допомогою вектора з розмірністю, B×5 + C, де B– кількість обмежувальних рамок, а C – кількість класів об’єктів. Так, якщо зображення поділене сіткою розміром S×S, то його можна представити

за допомогою тензора S×S× (B×5 + C). Опис вище стосувався нейронної мережі YOLO v1. Подальші версії містили багато вдосконалень, які й зробили дану нейронну мережу однією з найпопулярніших в наш час. Хронологію основних версій YOLO представлено на рисунку 3.14.

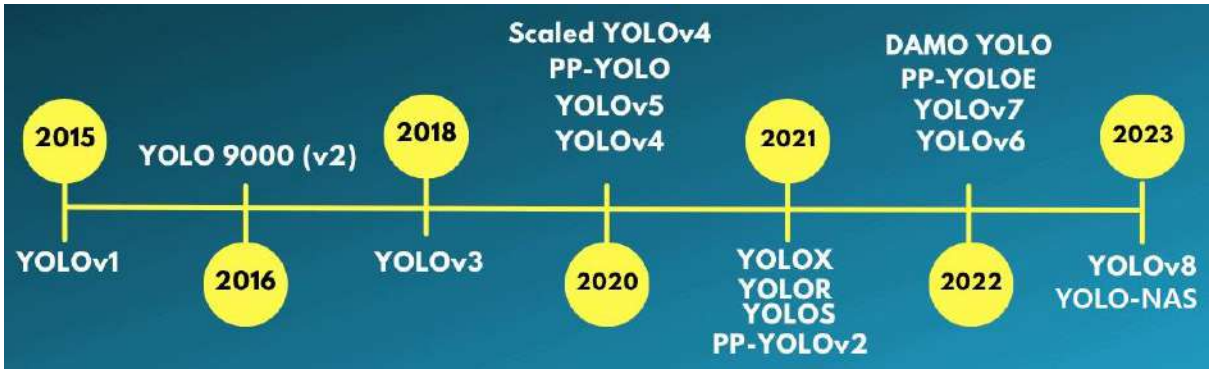


Рисунок 3.14 - Хронологія створення основних версій сімейства штучних нейронних мереж YOLO

Як можна спостерігати, опис математичної моделі YOLO потребував використання термінів вектор та тензор. Нижче в таблиці 3.4 наведено особливості використання цих математичних сутностей в контексті машинного навчання.

Таблиця 3.4 - Порівняння властивостей скаляра, вектора, матриці, тензора

Параметр	Скаляр	Вектор	Матриця	Тензор
Розмірність	0	1	2	≥ 3
Що представляє собою	Одиничне числове значення	Впорядкований масив значень	Двовимірний масив значень	Багато-вимірний масив значень

Продовження таблиці 3.4 - Порівняння властивостей скаляра, вектора, матриці, тензора

Параметр	Скаляр	Вектор	Матриця	Тензор
Використання	Представляють основні величини	Представляють особливості, спостереження	Упорядкування даних у табличному форматі	Робота зі складними структурами даних
Приклади	Метрики помилок, ймовірності	Об'єкти векторів, градієнтів	Матриці даних, вагові матриці	Тензори зображення, тензори послідовностей
Операції	Прості арифметичні дії	Операції лінійної алгебри	Матричні операції, лінійні перетворення	Тензорні операції, операції глибокого навчання
Представлення даних	Точка в просторі	Напрямок і величина величина в просторі	Рядки та стовпці в табличному форматі	Багато-вимірні зв'язки

Кінець таблиці 3.4 - Порівняння властивостей скаляра, вектора, матриці, тензора

Параметр	Скаляр	Вектор	Матриця	Тензор
Застосування	Основні розрахунки, статистичні заходи	Моделі машинного навчання, представлення даних	Маніпулювання даними, статистичний аналіз	Глибоке навчання, обробка природної мови

Розглянемо поняття тензора більш детально. Тензор є n-вимірним масивом, що задовольняє певному закону перетворення.[42] На відміну від матриці, він показує об'єкт, розміщений у певній системі координат. Більшість алгоритмів машинного навчання використовують тензор для виконання обчислень. У системі матриця є просто контейнером для записів, і вона не змінюється, якщо в системі відбуваються будь-які зміни, тоді як тензор — це сутність, яка взаємодіє з іншими сутностями в системі та змінює свої значення при зміні інших значень. Тензор часто розглядають як узагальнену матрицю. Тобто, це може бути 1-D матриця (вектор насправді є таким тензором), 2-D матриця, 3-D матриця (щось на кшталт куба чисел), навіть 0-D матриця (скаляр) або структура вищої розмірності, яку важче візуалізувати. Розмірність тензора називається його рангом. Але в цьому описі упускається найважливіша властивість тензора. Тензор — це математична сутність, яка живе в структурі та взаємодіє з іншими математичними сутностями. Якщо одна з них регулярно перетворює інші сутності в структурі, то тензор повинен підкорятися пов'язаному правилу перетворення. Будь-який тензор 2-го рангу може бути представлений у вигляді матриці, але не кожна матриця насправді є тензором 2-го рангу. Числові значення матричного представлення тензора залежать від того, які правила перетворення застосовані до всієї системи. Ця «динамічна» властивість тензора є ключовою, яка відрізняє його від простої

матриці[43]. Як приклад, розглянемо фрагмент нейронної мережі з 2 шарів (рисунок 3.15).

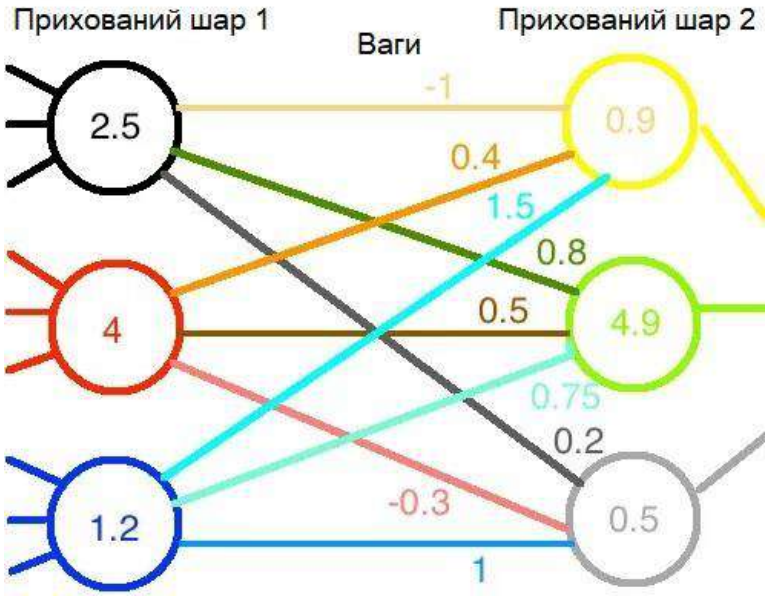


Рисунок 3.15 - Фрагмент нейронної мережі до внесення змін

Припустимо,на вузли прихованого шару 1 прийшли деякі дані, дані пройшли через ReLU функції цих вузлів, і на виході було отримано значення 2.5, 4 і 1.2 відповідно (потік даних проходить зліва направо). Можна представити вихідні дані цих вузлів у вигляді вектора:

$$L1 = \begin{bmatrix} 2.5 \\ 4 \\ 1.2 \end{bmatrix}. \tag{3.5}$$

Наприклад, кожен з 3-х вузлів прихованого шару 1 має вагу, пов'язану з його входом в кожен з вузлів прихованого шару 2. Представимо ці ваги у вигляді матриці:

$$W1 = \begin{bmatrix} -1 & 0.4 & 1.5 \\ 0.8 & 0.5 & 0.75 \\ 0.2 & -0.3 & 1 \end{bmatrix}. \tag{3.6}$$

В нашому випадку, всі ваги з одного рядка надходять до одного вузла в шарі 2, а ваги в певному стовпці надходять від того ж вузла в шарі 2. Наприклад, вага, яку вхідний вузол 1 вносить у вихідний вузол 3, дорівнює 0,2 (рядок 3, стовпець 1, див. рисунок 3.11). Ми можемо обчислити загальні значення, що подаються в вузли шару 2, помноживши матрицю ваги на вхідний вектор:

$$W1L1 = L2 \rightarrow \begin{bmatrix} -1 & 0.4 & 1.5 \\ 0.8 & 0.5 & 0.75 \\ 0.2 & -0.3 & 1 \end{bmatrix} \begin{bmatrix} 2.5 \\ 4 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 4.9 \\ 0.5 \end{bmatrix}. \quad (3.7)$$

Наприклад, виникла потреба використовувати спеціальні функції активації для кожного нейрона в шарі 1 шляхом перемасштабування кожної з функцій ReLU з першого шару окремо. Припустимо, що значення першого вузла було збільшено в 2 рази, для другого вузла значення не змінювалося, а для третього вузла зменшилося в 5 разів. Це еквівалентно множенню L1 на матрицю A:

$$A\ L1 = L3 \rightarrow \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 4 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 0.24 \end{bmatrix}. \quad (3.8)$$

Тепер, якщо ці нові значення подати через існуючу мережу ваг, на вузлах шару 2 отримаємо зовсім інші вихідні значення (рисунок 3.16)

Для того, щоб компенсувати зміни значень на виході шару 1, можна вплинути на ваги. З математичної точки зору, зробити це можна, використавши новий набір ваг, який ми отримуємо, множачи початкову матрицю ваг на матрицю A1, що є оберненою до матриці A:

$$W1A1 = W2 \rightarrow \begin{bmatrix} -1 & 0.4 & 1.5 \\ 0.8 & 0.5 & 0.75 \\ 0.2 & -0.3 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.4 & 7.5 \\ 0.4 & 0.5 & 3.75 \\ 0.1 & -0.3 & 5 \end{bmatrix}. \quad (3.8)$$

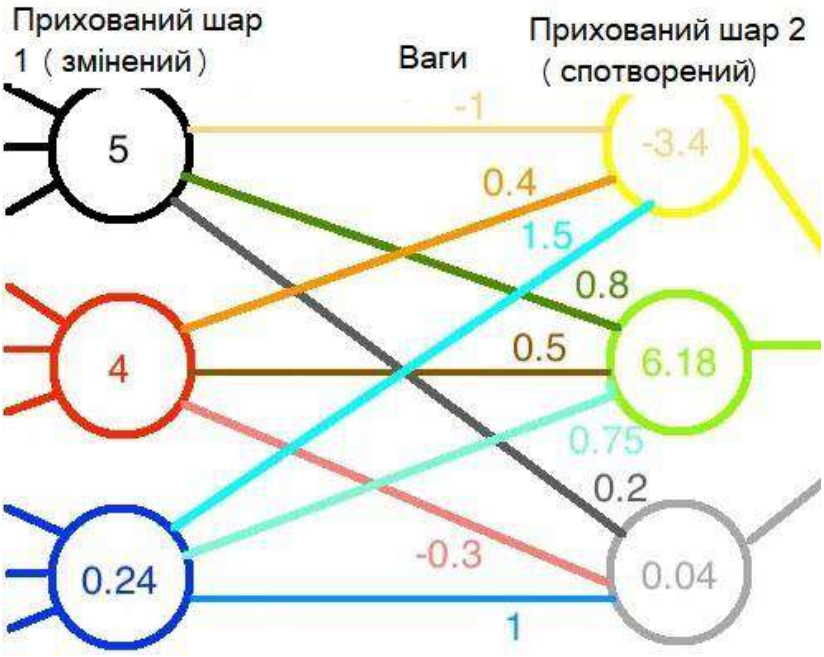


Рисунок 3.16 - Фрагмент нейронної мережі після внесення змін у шар 1

Якщо ми об'єднати модифікований вивід шару 1 із зміненими вагами, на входах шару 2 отримаємо правильні значення (рисунок 3.17) . Можна назвати зміни, внесені у вузли шару 1, коваріантними, а модифікований тензор ваг – контраваріантним тензором.

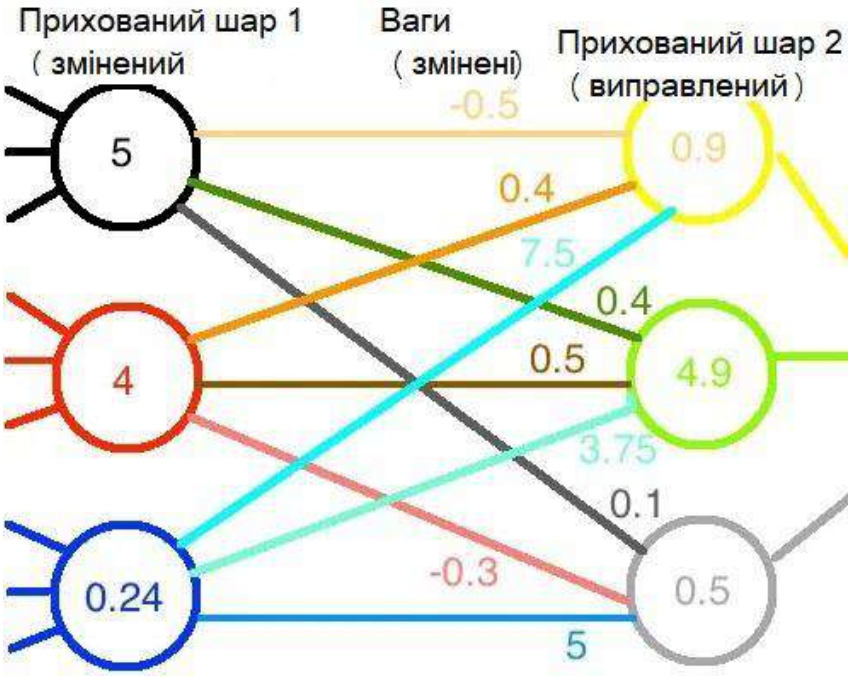


Рисунок 3.17 - Фрагмент нейронної мережі із компенсованими вагами

3.4 Висновки

В даному розділі було запроновано метод та алгоритм обробки похибок отриманих відеозображень нейронною мережею. Також було розглянуто математичну модель роботи нейронної мережі YOLOv1, і роботу тензора ваг у нейронній мережі.

На основі даної інформації буде розроблено програмну реалізацію у вигляді застосунків на мовах Python та PHP.

4 РЕЗУЛЬТАТИ ОБРОБКИ ПОХИБОК ОТРИМАНИХ ВІДЕОЗОБРАЖЕНЬ НЕЙРОННОЮ МЕРЕЖЕЮ

4.1 Проектування архітектури системи для обробки похибок отриманих відеозображень нейронною мережею

Функціонально система складається з 3-х взаємопов’язаних підсистем, в яких можна виділити кілька важливих модулів (рисунок 4.1)

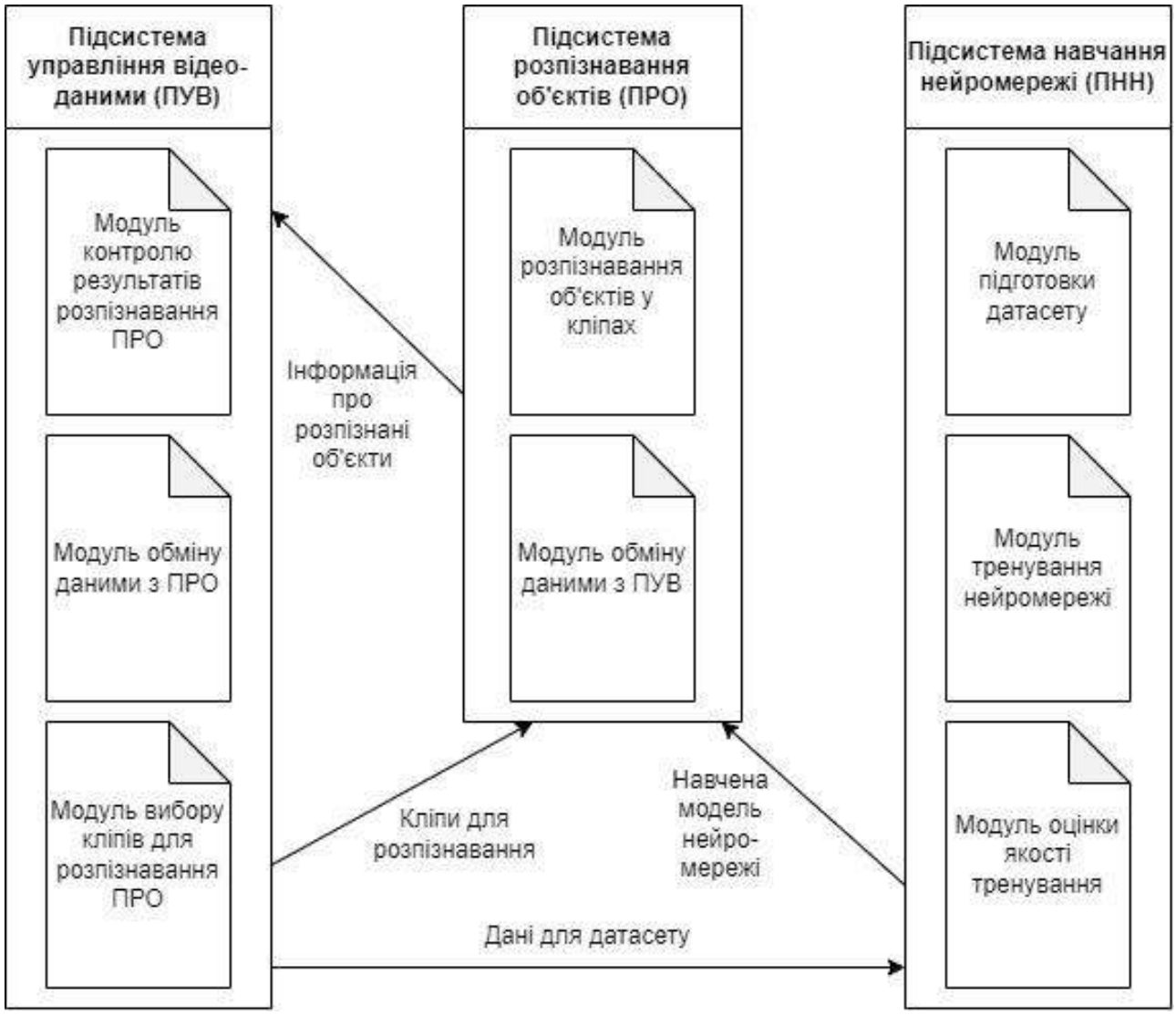


Рисунок 4.1 - Функціональна схема системи для обробки похибок

Підсистема управління відео-даними реалізована мовою PHP, дві інші – мовою Python. Опис модулів всіх підсистем наведено в таблиці 4.1.

Таблиця 4.1 - Опис модулів всіх підсистем

Назва підсистеми	Назва модуля	Короткий опис модуля
Підсистема управління відео-даними (ПУВ)	Модуль вибору кліпів	Згідно вимог замовника, перевірці нейромережею повинні підлягати не всі підряд кліпи, що надходять в систему, а лише деякі, що задовольняють певним критеріям. Крім того, слід врахувати що іноді розпізнавання відбувається повільніше, ніж надходять кліпи, які підлягають розпізнаванню. Тому кліпи ставляться в чергу. Також бувають ситуації, коли кліпи із черги повинні бути повернуті до ручної обробки операторами (наприклад, при втраті зв'язку з підсистемою розпізнавання, чи виході її з ладу, або примусово рішенням користувача системи). Також передбачена опція паузи в обробці нейромережею (у випадку серії невдалих спроб), з автоматичним відновленням такої обробки.
	Модуль обміну даними	Спрощено, містить 2 функції, реалізовані за допомогою бібліотеки CURL: - надсилання інформації у ПРО щодо відео-кліпу для розпізнавання; - приймання від ПРО інформації про оброблений кліп: оброблений кліп з позначеними bounding boxes, текстовий файл з інформацією про виявлені об'єкти, а також висновок про належність кліпу до певного користувацького класу.

Продовження таблиці 4.1 - Опис модулів всіх підсистем

Назва підсистеми	Назва модуля	Короткий опис модуля
	Модуль контролю результа-тів розпізна-вання	Власне, модуль візуалізує для оператора результати розпізнавання у вигляді 2 користувацьких сторінок: статусу кліпу (рисунок 4.2), із вказанням поточного стану кліпу (наприклад, ‘поставлено у чергу’), результати та час розпізнавання тощо, а також сторінки статистики (рисунок 4.3).
Підсистема розпізна-вання об’єктів (ПРО)	Модуль розпізна-вання об’єктів	Пофреймово розпізнає об’єкти на відео, створює кліп з нанесеними bounding boxes, створює текстовий файл з детальною інформацією по виявлених об’єктах тощо. Основну роботу здійснюють функції бібліотек ultralytics (забезпечує роботу нейромережі YOLO у Python) та cv2. Нижче на рисунку 4.4 наведено спрощений алгоритм роботи модуля.
	Модуль обміну даними	Містить функції отримання інформації про кліп від ПУВ, та відправки назад оброблених даних. Реалізовано за допомогою технології webhook. Використано бібліотеки flask та requests.
Підсистема навчання нейромережі (ПНН)	Модуль підготовки датасету	Використовуючи попередньо підготовлений список кліпів (у csv файлі), завантажує з ПУВ відео-файли, і готує набір для навчання нейромережі (зображення та файли міток для них).
	Модуль тренуван-ня нейро-мережі	Складається з набору Python – скриптів та конфігураційних файлів для навчання.

Кінець таблиці 4.1 - Опис модулів всіх підсистем

Назва підсистеми	Назва модуля	Короткий опис модуля
	Модуль оцінки якості трену-вання	На даний момент складається із скриптів, що забезпечують візуалізацію результатів навчання за кількома метриками, для подальшого аналізу людиною. В майбутньому можливе доповнення скриптами, що здійснюють додаткові розрахунки (наприклад, метрики F1 в розрізі епох навчання).

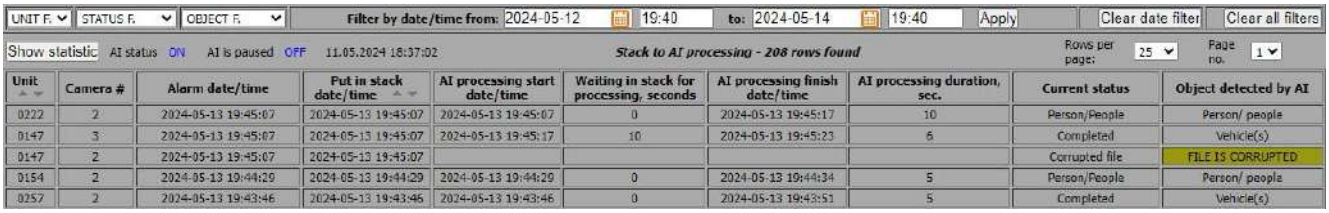


Рисунок 4.2 - Сторінка статусу кліпів

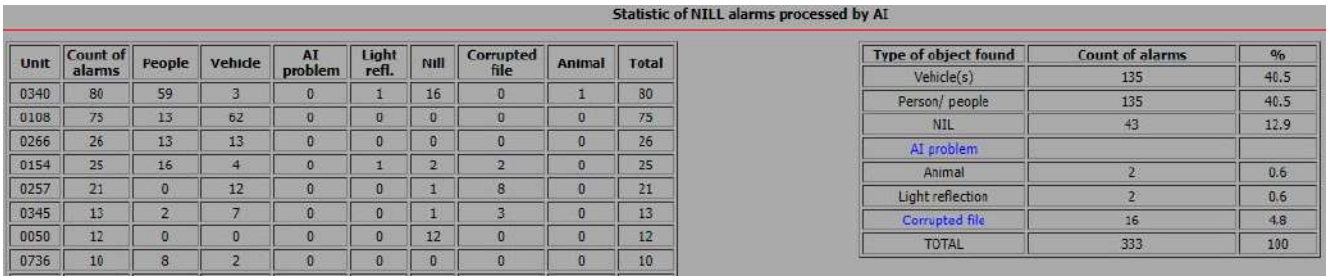


Рисунок 4.3 - Сторінка статистики розпізнавання

Слід зробити деякі зауваження щодо алгоритмів роботи модулів. Так, у модулі розпізнавання об’єктів (рисунок 4.4) після завантаження в пам’ять моделі нейромережі запускається безкінечний цикл опитування стану файлу, з можливістю ручного переривання роботи. Таке рішення виникло внаслідок того, що етап завантаження моделі нейромережі в пам’ять виявився найбільш тривалим (на Jetson Orin продовжувався близько 20 секунд), і варіант кожного разу при появі нового кліпу робити це не був прийнятним. У модулі підготовки датасету (рисунок 4.5) була необхідність збереження кліпів з виявленими людьми двічі (з

bounding boxes та без) для забезпечення як матеріалів для подальшого навчання мережі, так і візуального контролю результатів поточного розпізнавання.

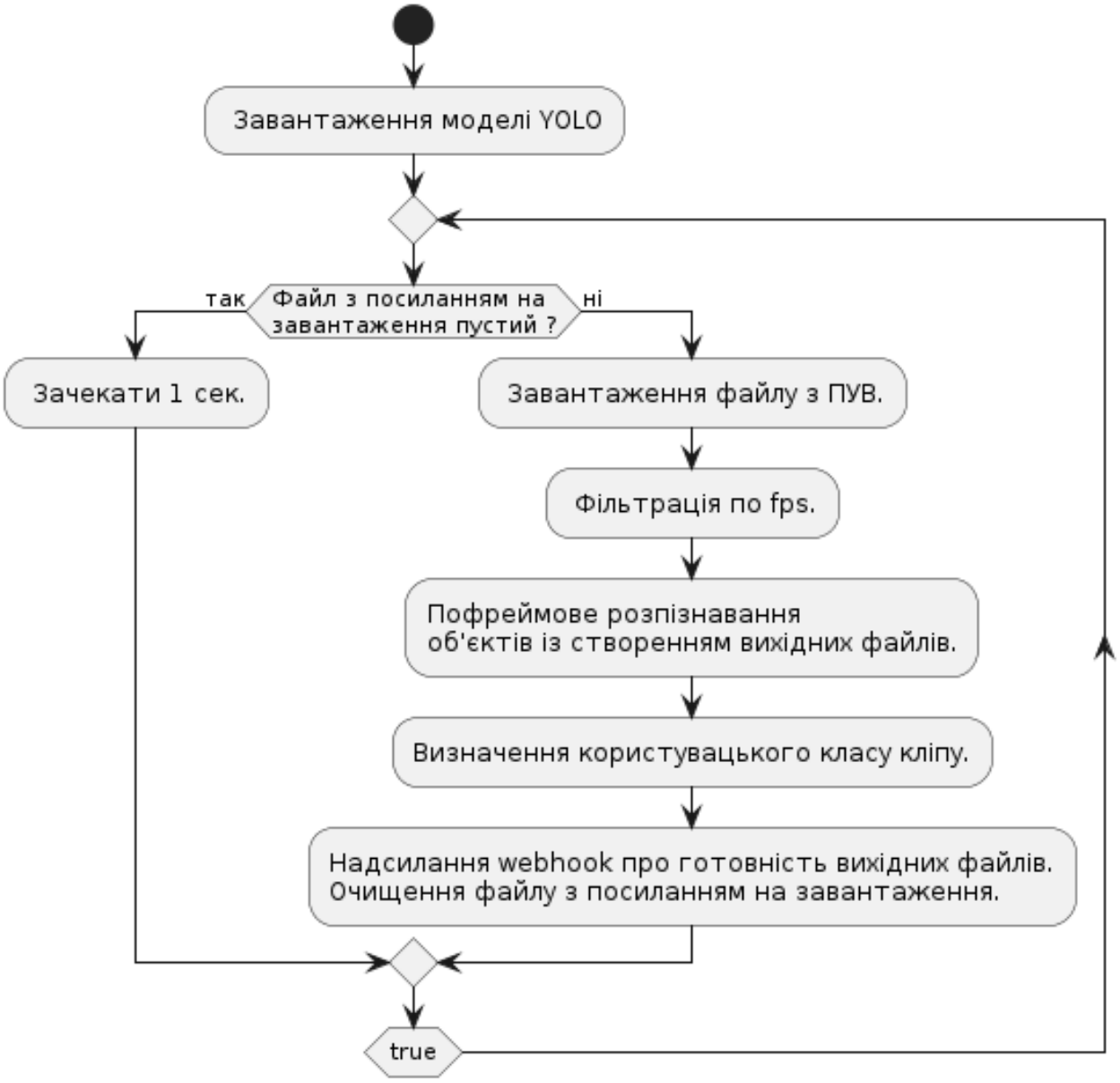


Рисунок 4.4 - Алгоритм роботи модуля розпізнавання об’єктів

4.2 Аналіз результатів експериментів, та оцінка точності обробки похибок отриманих відеозображень нейронною мережею

Експерименти проводилися в кілька етапів, серед них можна виділити 3 основні:

1. Перевірка роботи комплексу на розпізнаванні усіх доступних класів об’єктів з використанням коробочної версії моделі YOLO (yolov8x.pt, найбільша з моделей).

2. Вибір одного класу, і тренування своєї моделі, використовуючи ваги моделі yolov8s.pt (одна з менших моделей).

3. Вдосконалення результатів через покращення навчального датасету, тренування з вагами наступної по розміру моделі, yolov8m.pt.

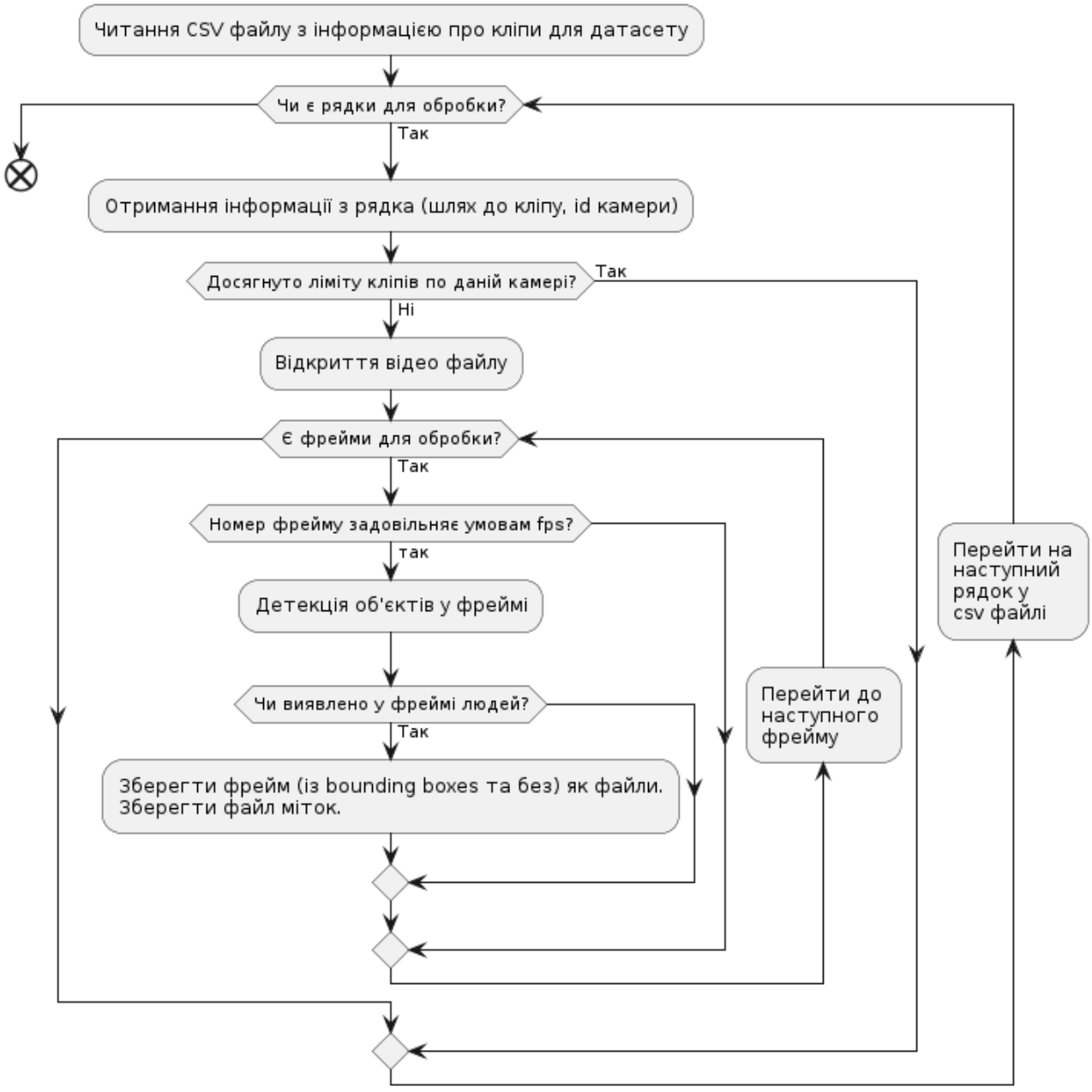


Рисунок 4.5 - Алгоритм модуля автоматизованої підготовки датасету

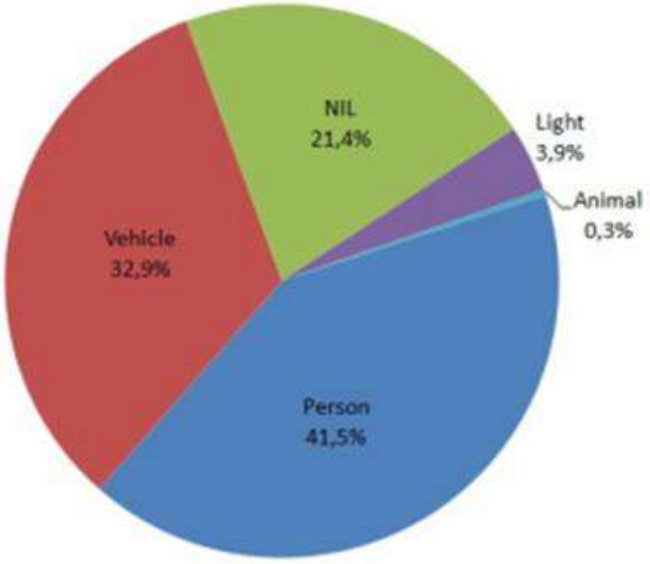
Отже, перший етап робіт підтвердив правильність вибраних методів і алгоритмів, і продемонстрував фактичне співідношення кількості кліпів з різними типами об’єктів (див. рисунок 4.6)

Оскільки основною метою відеоспостереження на об’єктах, де були встановлені камери, з яких бралися кліпи для експериментів, є охорона об’єктів, на цьому етапі було вирішено провести додаткову перевірку результатів, де були виявлені люди (клас Person). Перевірку проводили люди-оператори. Слід зазначити, що кліпи надсилалися на опрацювання системою в моменти, коли, за попередньою оцінкою, на об’єктах не повинно бути людей. Результати перевірки відеозаписів, на яких нейромережа виявила людей (вибірка 4837 роликів за 6 днів), наведені на рисунку 4.7 у вигляді діаграми. Лише 0,7% із загальної кількості кліпів, визначених програмою як Person, були підтверджені операторами. Тобто похибка другого роду склала 99,3%. Кількість об'єктів Person, виявлених нейронною мережею в кожному відеоролику, варіюється від 1 до 373 (слід зазначити, що загальна кількість кадрів у роликах становить від 8 до 900). Частота, з якою об’єкти класу Person зустрічаються у файлах, зображена на рисунку 4.8.

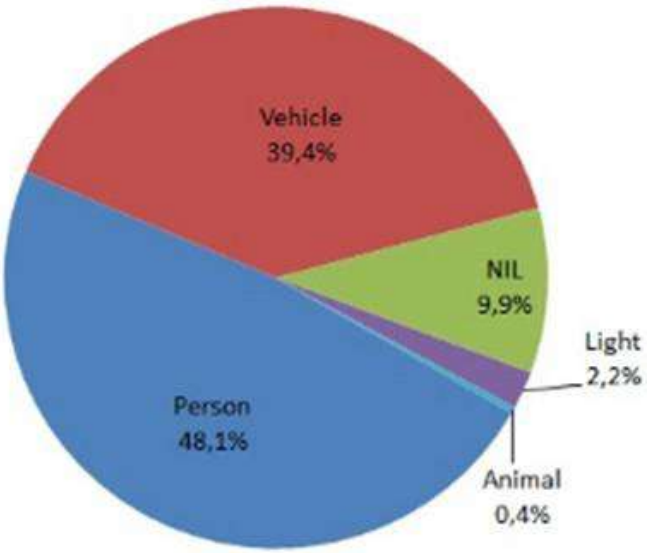
Для підвищення якості розпізнавання було вирішено навчити модель розпізнавання об’єктів класу «людина». Для створення набору даних із класом «людина» було відібрано відеоролики зі 153 камер. Розпізнавання здійснювалося програмою Python, з використанням коробочної (навченої розробником) моделі yolo8x.pt. Результати розпізнавання були перевірені вручну. Відсоток помилок, допущений цією коробочною версією, становив 19.4%.

Рисунок 4.6 - Фактичний розподіл кліпів за класами

Під час підготовки набору даних було отримано 5208 зображень з об’єктами класу «людина». 4908 зображень, що містили 6121 об’єкт класу «людина», було використано для тренування, 300 зображень – для валідації. Частина зображень для валідації становила 5,8% від загального обсягу датасету. Кількість використаних фонових зображень (background images), що не містили жодних об’єктів – 92 (1,8% від обсягу датасету).



a) Добова вибірка (1440 кліпів)



b) Тижнева вибірка (11247 кліпів)

Навчання моделі здійснювалося на Google Colab, дані завантажувалися з Google Drive. Було запущено 120 епох навчання. Навчання було розпочато з пустими вагами для моделі yolo8s. Використовувався GPU Tesla V-100 16Gb, кожен цикл тривав близько 2 хв., загальна тривалість навчання становила 3 год. 40 хв. Інформація про хід експерименту представлялася в робочому вікні в режимі реального часу (див. рисунок 4.9)

Оцінювання результатів навчання здійснювалося 2-ма шляхами: за допомогою отриманих метрик (див. нижче рисунки 4.10-14.13), та вручну (отримання датасету з відео за допомогою новоствореної моделі, ручний пошук помилкових результатів).

Слід зазаначити, що завдяки продуманому API робота по контролю якості проведеного навчання значно спрощується. Так, наприклад, набір графіків, що дозволяють оцінити навчання (рисунок 4.10) створюється бібліотекою ultralytics автоматично, за замовчуванням, не вимагаючи вказання ніяких додаткових опцій. Так само автоматично створюється csv-файл (рисунок 4.11), що містить значення метрик (Train Box Loss, Validation Box Loss та інших, всього 14 метрик) для кожної епохи. Використовуючи дані цього csv-файлу, за допомогою Python-

скриптів були створені графіки для параметрів Train Box Loss і Validation Box Loss (рисунок 4.12) , а також mAP50 (рисунок 4.13).

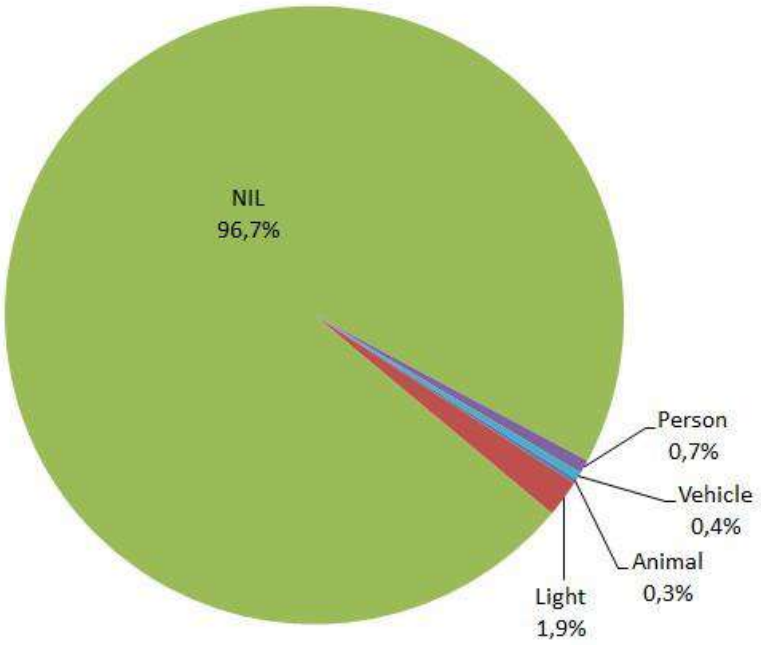


Рисунок 4.7 - Результати верифікації відео кліпів, на яких нейромережа виявила людей

Ручна перевірка показала зменшення долі об’єктів, що були визначені новоствореною моделлю з помилками, з 19,4 до 10,7%. Що стосується отриманих метрик, то вони вказують з одного боку, на успішність навчання : спостерігається покращення значень параметрів Train Box Loss (до 0.5135), mAP50 (до 0.98367) із збільшенням епох. З іншого боку, порівняно стабільне значення Validation Box Loss (0.69291) починаючи з 84 епохи наводить на думку щодо природних флуктуацій продуктивності через занадто малу валідаційну вибірку (6% від навчальної).

Наступним етапом в роботі стало збільшення розміру датасету. Основну роботу по підбору зображень з відеокліпів, визначенню людей на цих зображеннях, і створенню файлів міток для навчання, як і у випадку попереднього набору, було зроблено за допомогою скрипту на Python. Але цього разу для розпізнавання було використано модель, створену під час попереднього навчання

(120 епох на вагах моделі S). Тому є щонайменше 2 причини : по перше, нова модель показала кращі показники розпізнавання, по друге, як, показали досліди, розпізнавання на ній відбувається на 50% швидше, ніж на коробочній yolov8x.

К-ть виявлених об'єктів Person у кліпі	К-ть файлів (кліпів)
1	1248
2	572
3	414
4	278
5	208
6	193
7	169
8	149
9	137
10	101
...	...
230	1
285	1
288	1
372	1
373	1

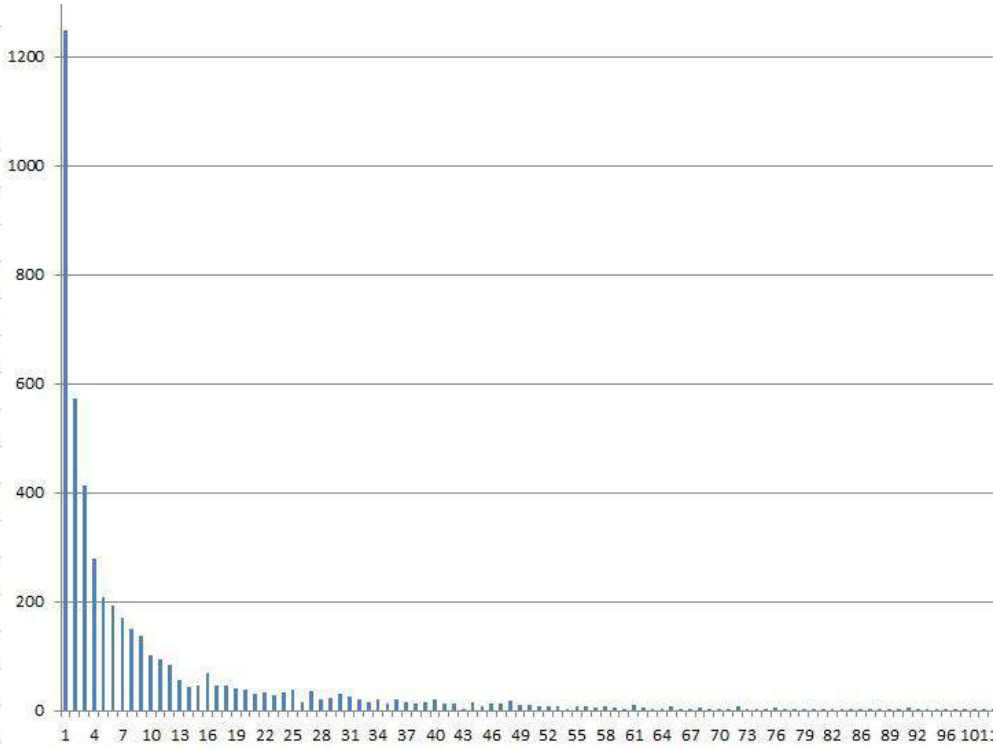


Рисунок 4.8 - Частота, з якою об'єкти класу Person зустрічаються у кліпах

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    119/120      15.5G     0.5156     0.3061     0.9842         12       1280: 100%|██████████| 307/307 [01:39<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 10/10 [00:

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    120/120      15.5G     0.5135     0.3027     0.9805         17       1280: 100%|██████████| 307/307 [01:40<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 10/10 [00:

120 epochs completed in 3.547 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.6MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.6MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.1.7 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla V100-SXM2-16GB, 16151MiB)
YOLOv8s summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 10/10 [00:
                   all        300        383      0.964      0.948      0.983      0.828
Speed: 0.4ms preprocess, 3.9ms inference, 0.0ms loss, 1.4ms postprocess per image
```

Рисунок 4.9 - Робоче вікно Google Colab на момент завершення експерименту з навчанням на 120 епох

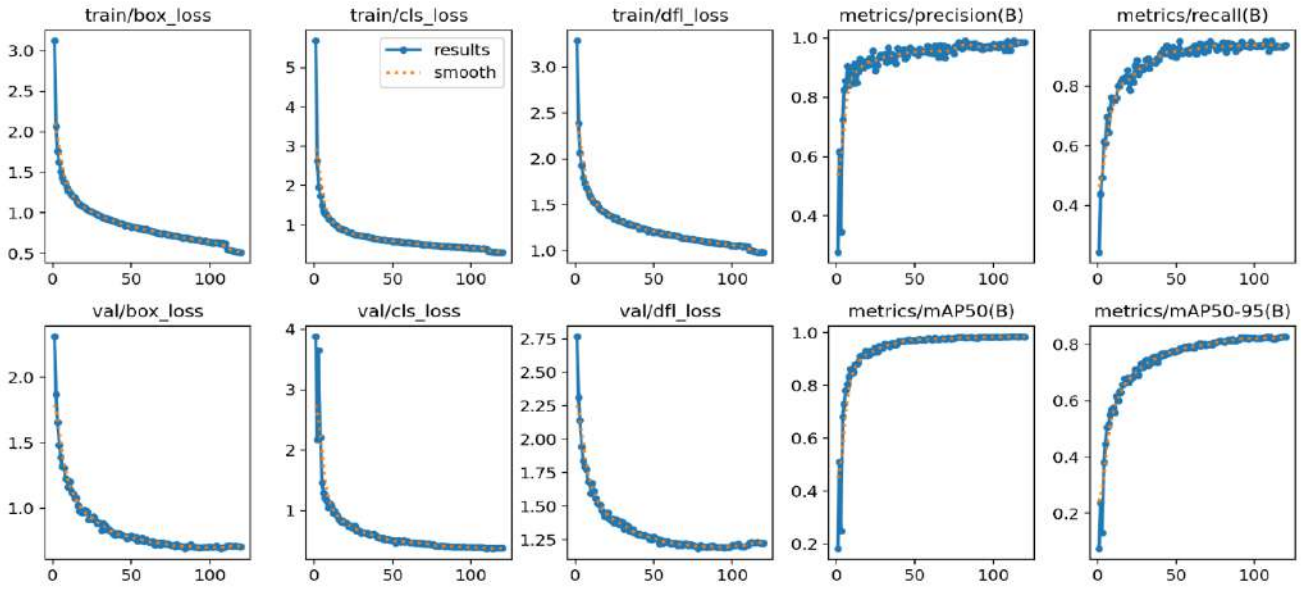


Рисунок 4.10 - Набір метрик для оцінювання результатів навчання на 120 епох

epoch	train/box_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)
112	0.54225	0.98553	0.9295	0.98411
113	0.54041	0.99165	0.93069	0.98389
114	0.53052	0.97817	0.93605	0.98414
115	0.52697	0.98593	0.93211	0.98417
116	0.52797	0.98585	0.93211	0.98339
117	0.51891	0.98619	0.93254	0.98474
118	0.5235	0.98623	0.9295	0.9845
119	0.51555	0.98617	0.93473	0.98439
120	0.5135	0.98625	0.93633	0.98367

Рисунок 4.11 - Фрагмент csv-файлу з набором метрик для 120 епох

Як і в попередньому випадку, всі результати машинного розпізнавання були перевірені вручну. У випадку виявлення помилок розпізнавання такі зображення копіювалися в окрему папку, і для них розмітка потім створювалася вручну.

Отже, для наступного сеансу навчання було підготовлено датасет (зображення та файли міток) , що містив 11152 зображень із 13754 об’єктами класу «людина». Датасет був розділений на 2 частини, тренувальну та валідаційну, у співвідношенні 9486 / 1666 (обсяг валідаційної вибірки становив

14,9 % від загального обсягу). Фонових зображень було застосовано 208 (1,9% від загального обсягу).

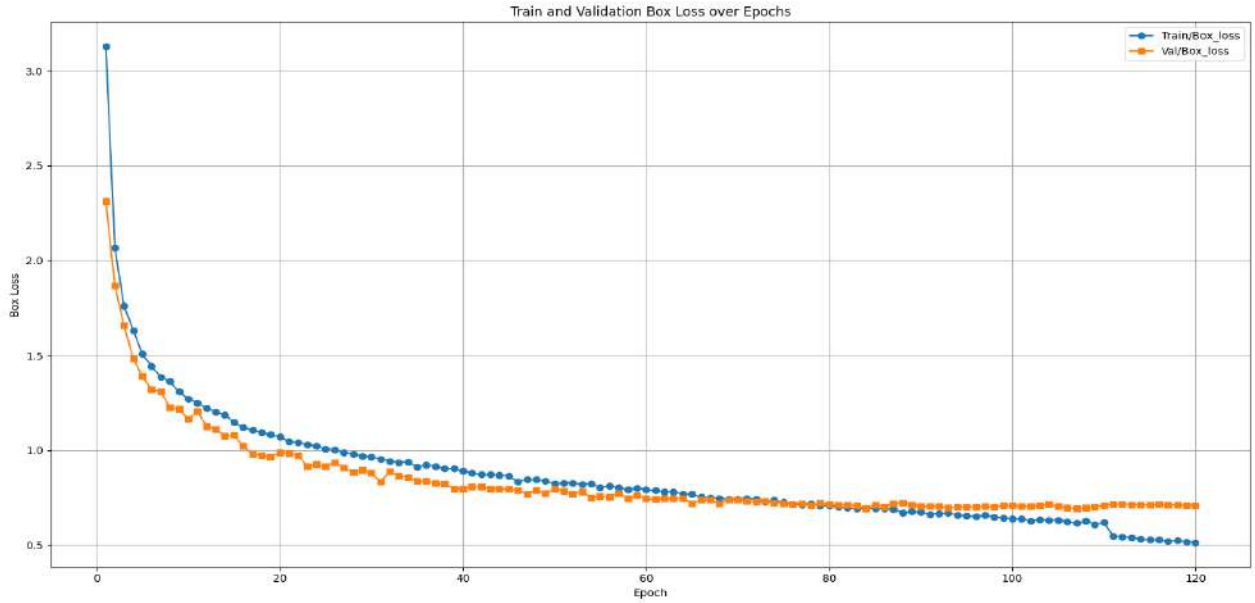


Рисунок 4.12 - Помилки Train Box Loss і Validation Box Loss під час навчання нейронної мережі на 120 епох

Навчання моделі також проходило на Google Colab. Було запущено 300 епох навчання. Навчання було розпочато з пустими вагами для моделі yolo8m. Використовувався GPU Tesla A-100 40Gb, тривалість кожної епохи близько 4 хв., загальна тривалість навчання становила 19 год. 55 хв. Метрики для оцінки результатів навчання наведено нижче на рисунках 4.14-4.16.

По результатах навчання на вагах моделі M спостерігаємо картину, подібну до результатів минулого навчання (на вагах моделі S): Training Box Loss продовжує знижуватися до останніх епох, натомість Validation Box Loss та mAP50 виходять на плато набагато раніше, і з часом навіть крапельку погіршуються (але не значно, на рівні 0.001).

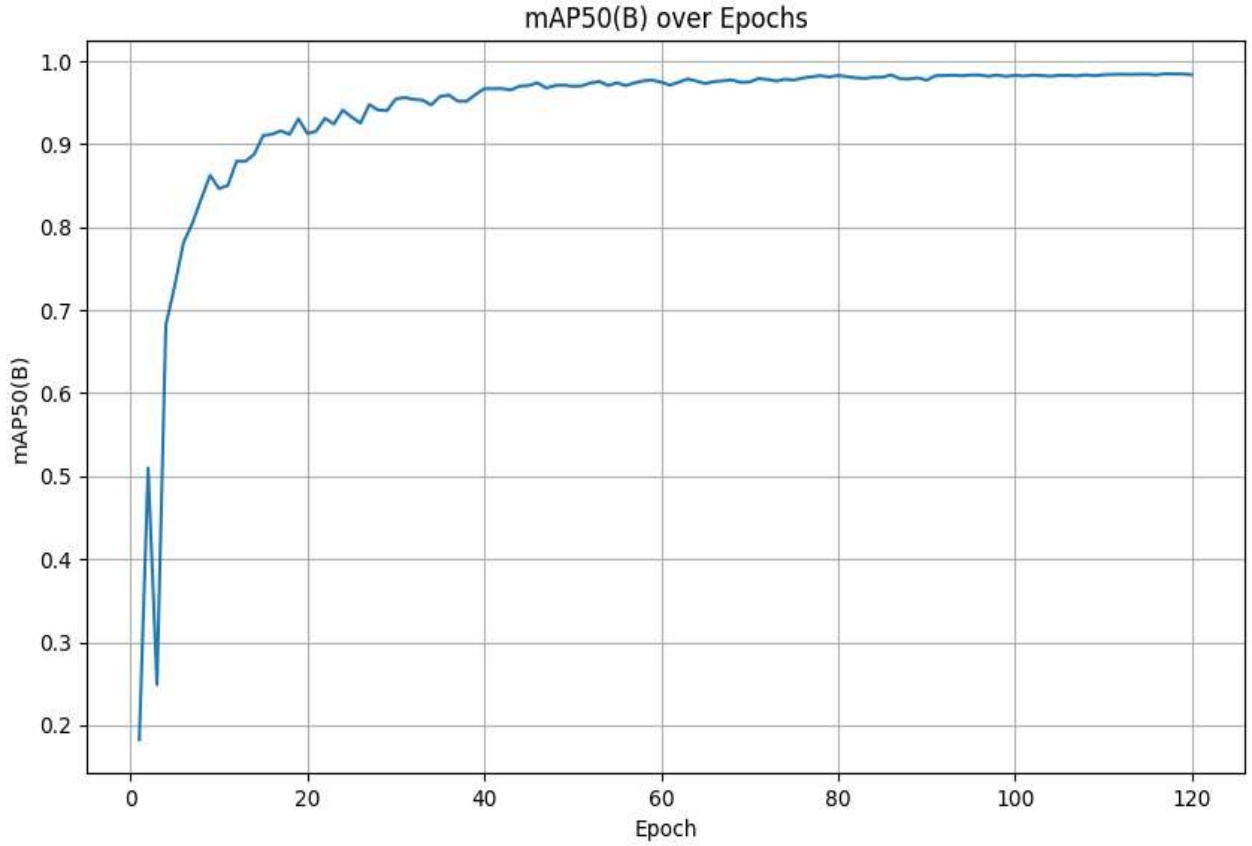


Рисунок 4.13 - Середня точність при пороговому значенні IoU=0.5, протягом 120 епох

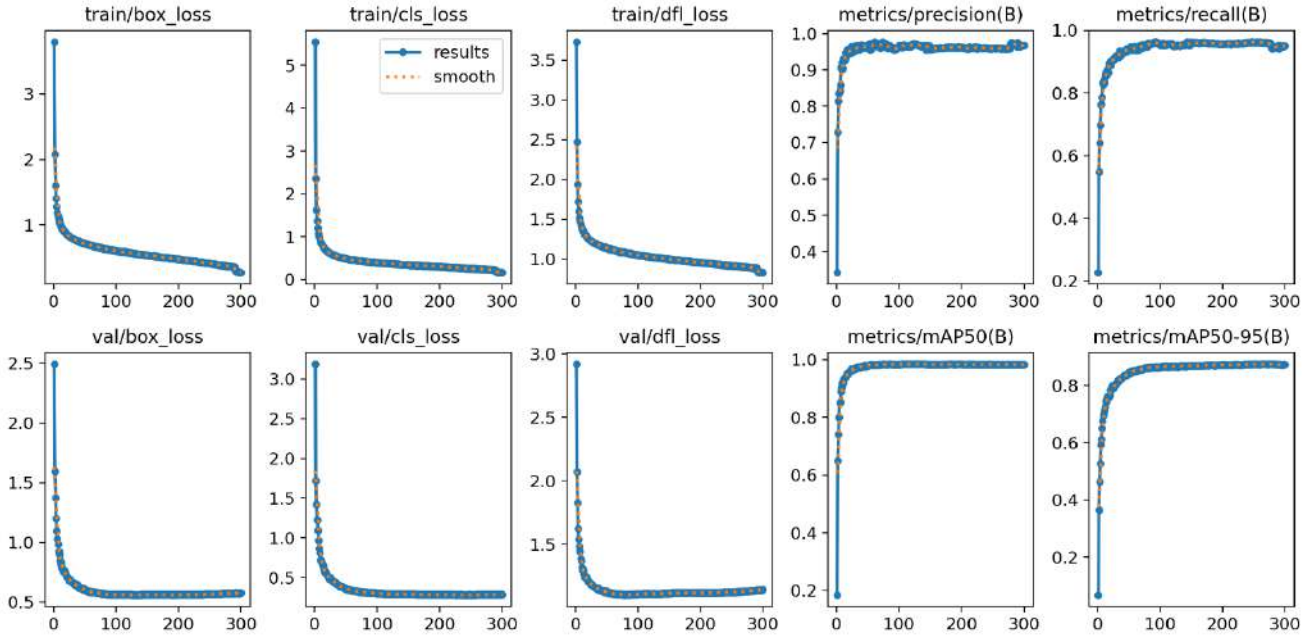


Рисунок 4.14 - Набір метрик для оцінювання результатів навчання на 300 епох

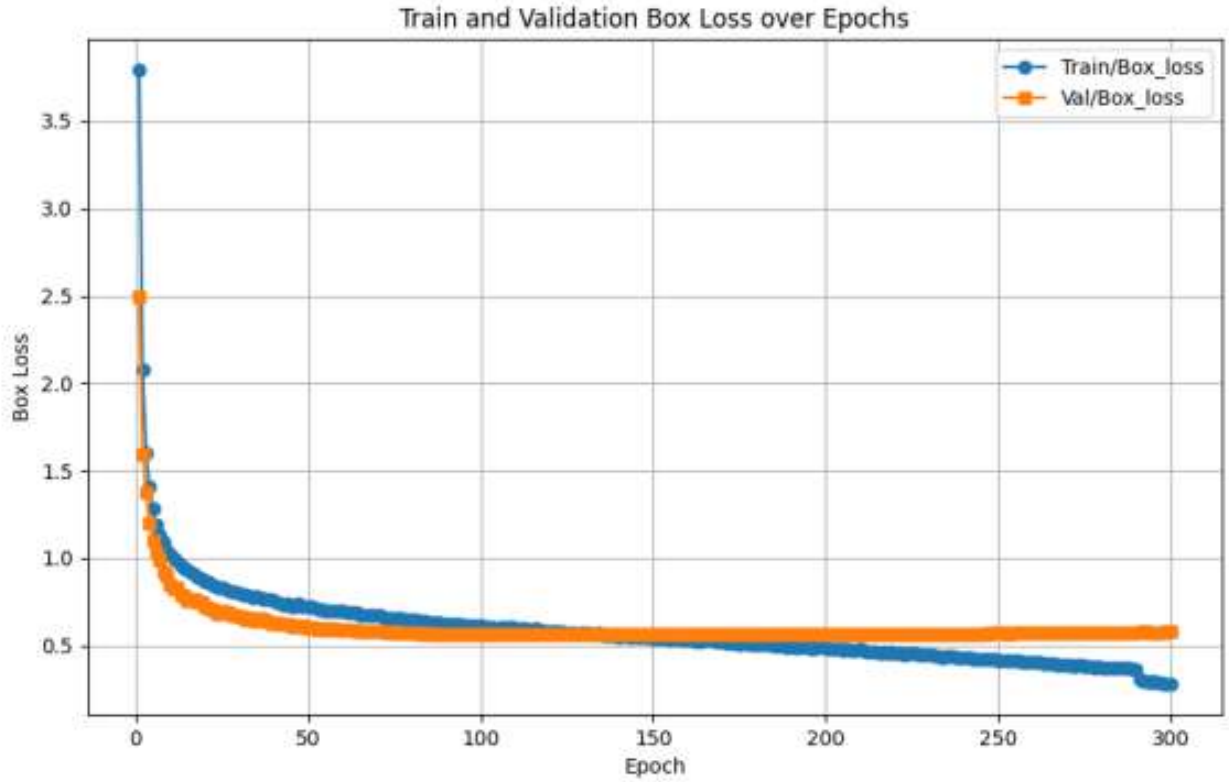


Рисунок 4.15 - Помилки Train Box Loss і Validation Box Loss під час навчання та валідації нейронної мережі на 120 епох

Опція Early Stopping була виставлена при навчанні, тобто сама нейромережа не виявила перенавчання. Порівняно з минулим тренуванням (там де 120 епох) значення Validation Box Loss покращилося : стало 0.5765 , було 0.70797.

Можна зробити наступний висновок: поточне покращення результату відбулося завдяки збільшенню розміру тренувального датасету (з 5208 зображень до 11152) . Також позитивним вважаю рішення використати ваги (weights) від моделі більшого розміру (раніше це була yolo8s, зараз yolo8m).

Для оцінки результатів навчання за допомогою формул (1.5), (1.7), (1.8) розрахуємо значення параметра F1, використовуючи дані матриці невідповідності (див. рисунок 4.17).

Для наочності зведемо вхідні дані і розрахунки в таблицю 4.2.

Для додатової оцінки результатів навчання для двох моделей порівняємо графіки залежності параметра F1 від рівня впевненості (confidence), див. рисунки 4.18, 4.19. Як ми бачимо в розрахунках з таблиці 4.2, в обох випадках значення

F1=0.96 , що є досить високим показником і свідчить про якісно проведене навчання. При цьому графіки з рисунку 4.22, 4.23 показують, що в другому випадку (там, де модель M) крива залежності більш рівномірна. Зважаючи на те, що в першому випадку (модель S) параметр F1 досягає своего максимуму при Confidence = 0.345, в другому – при 0.469, тобто модель S продемонструвала більш консервативну поведінку у детекції об’єктів.

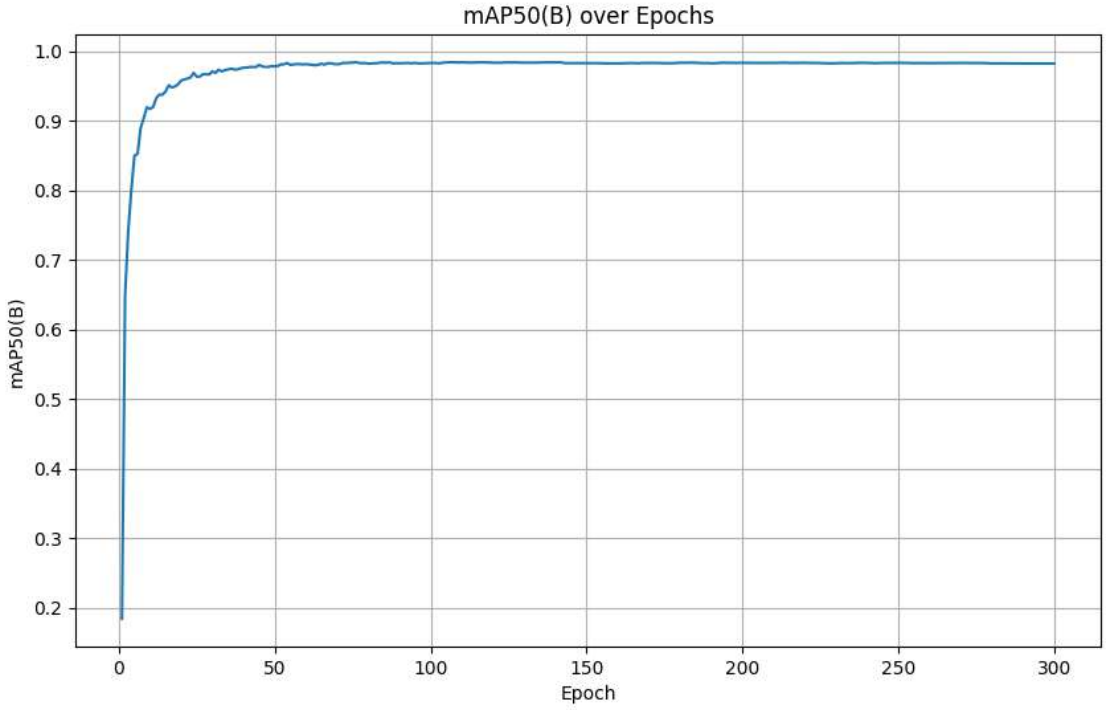


Рисунок 4.16 - Середня точність при пороговому значенні IoU=0.5, протягом 120 епох.

Порівняємо, як змінувалися значення параметра F1 протягом епох для обох моделей (рисунки 4.20 та 4.21). Як видно, на графіку моделі S спостерігається більше флуктуацій ефективності. Проте це також може бути спричинено невеликим розміром валідаційного датасету. Нагадаю, для моделі Sвін становив всього 300 зображень (5,8% від загального обсягу), для другого експерименту (з моделлю M) валідаційний датасет був збільшений як у абсолютному , так і у відносному значенні – до 1666 (14.9%) відповідно.

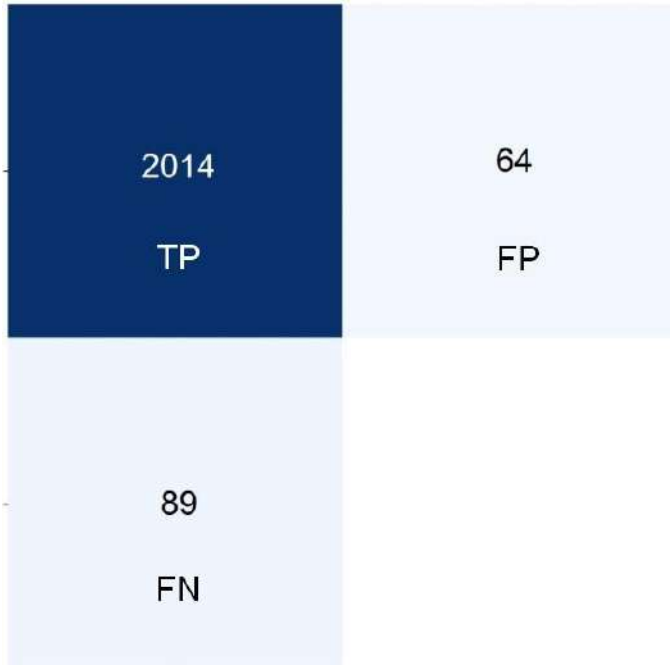


Рисунок 4.17 - Матриця невідповідності з вагами моделі yolov8m

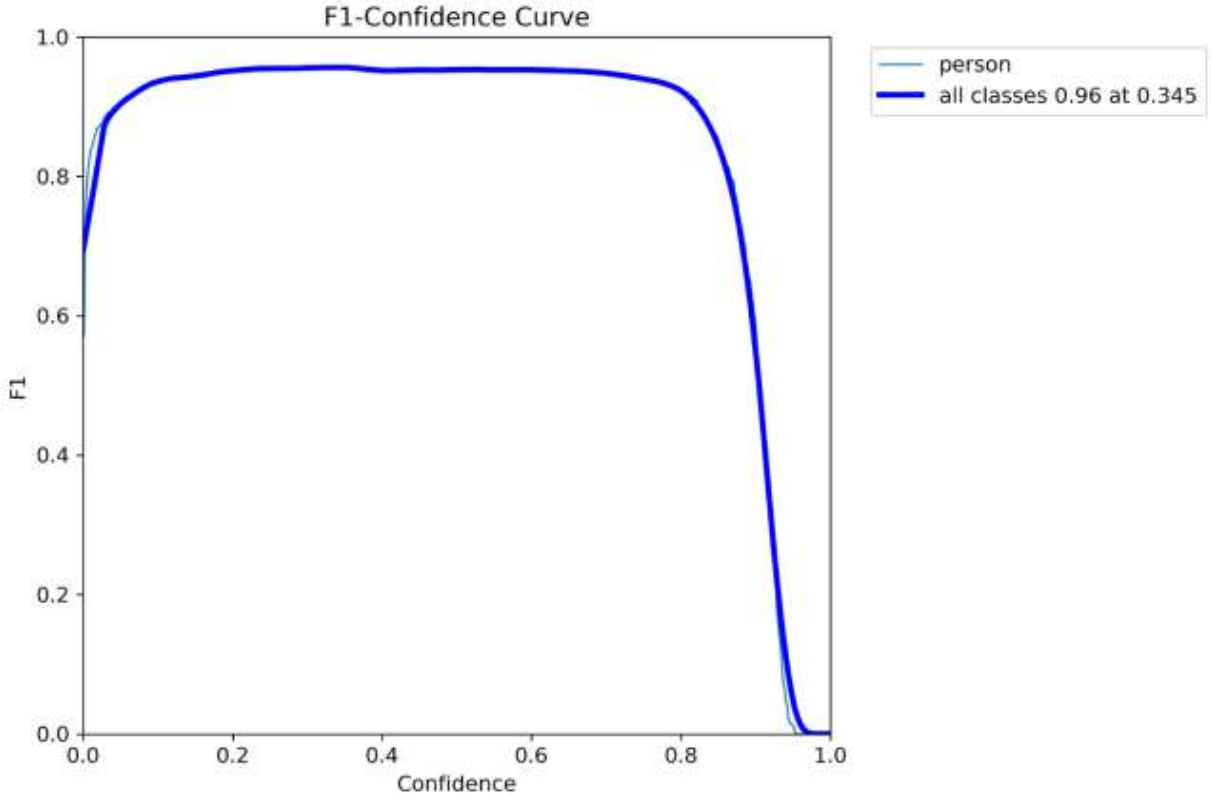


Рисунок 4.18 - Залежність параметра F1 від рівня впевненості від рівня впевненості (confidence) для моделі S (тренування на 120 епох)

Таблиця 4.2 - Розрахунки метрики F1 для двох тренованих моделей.

Модель	S	M
Параметри матриці невідповідності	TP (True Positives) = 366 FP (False Positives) = 16 FN (False Negatives) = 17	TP (True Positives) = 2014 FP (False Positives) = 64 FN (False Negatives) = 89
Розрахунки	Precision = TP / (TP + FP)=366 / (366 + 16) ≈ 0.958 Recall = TP / (TP + FN)=366 / (366+17) ≈ 0.956 F1 Score = 2 * (Precision * Recall) / (Precision + Recall)=2 * (0.958 * 0.956) / (0.958 + 0.956) ≈ 2 * (0.915) / (1.914) ≈ 1.830 / 1.914 ≈ 0.956	Precision = TP / (TP + FP)=2014 / (2014 + 64) ≈ 0.969 Recall = TP / (TP + FN)=2014 / (2014 + 89) ≈ 0.958 F1 Score =2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.969 * 0.958) / (0.969 + 0.958) ≈ 2 * (0.929) / (1.928) ≈ 1.857 / 1.928 ≈ 0.963

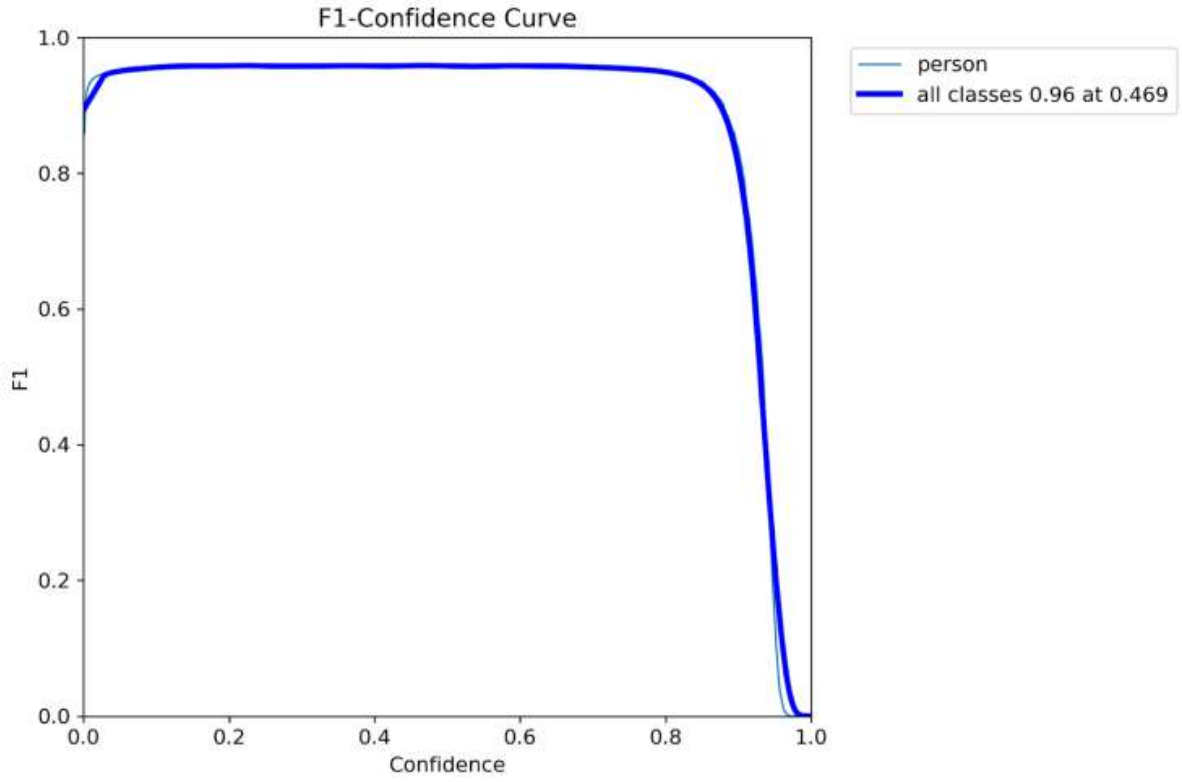


Рисунок 4.19 - Залежність параметра F1 для моделі M (тренування на 300 епох)

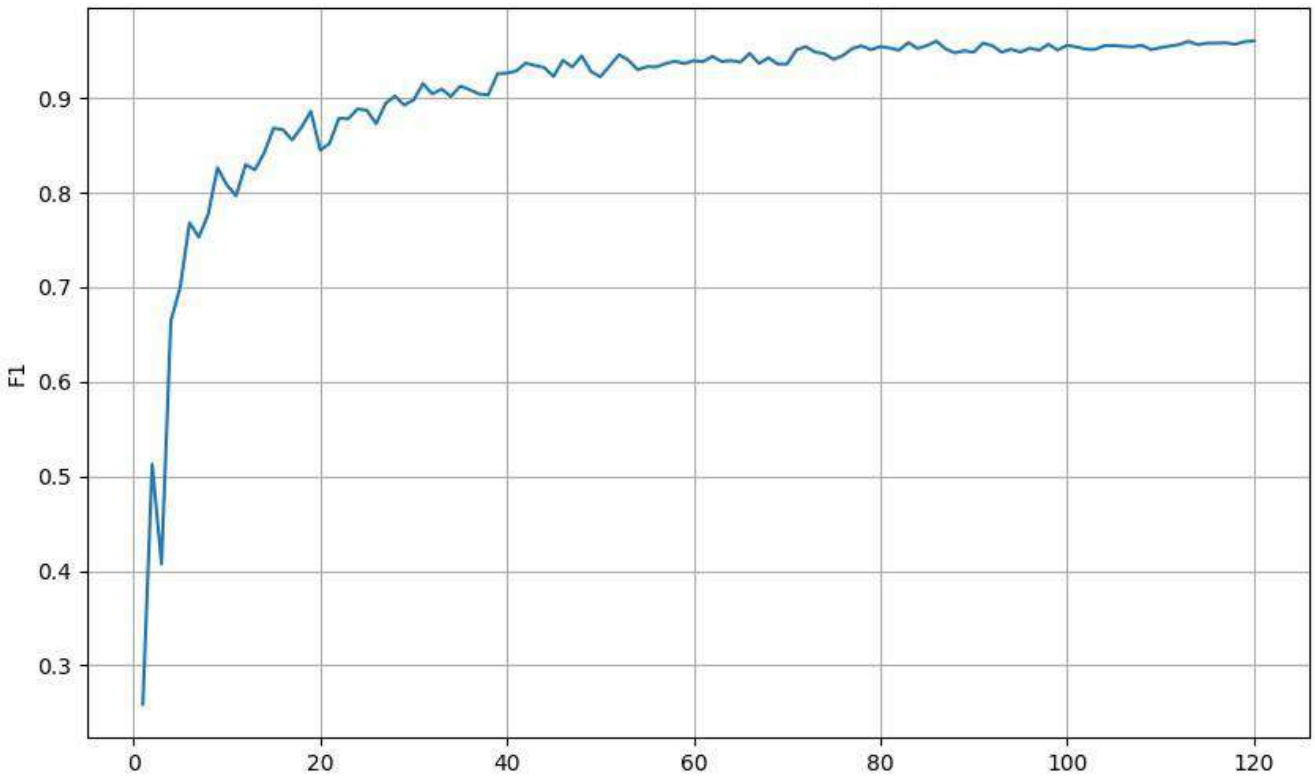


Рисунок 4.20 - Значення F1 протягом епох навчання для моделі S

Додатково були проведені тестування на датасетах з зображеннями з тих камер, зображення з яких не використовувалися в навчанні. Це було зроблено з метою оцінки здатності мереж узагальнювати ознаки при навчанні. Результати тестувань для обох моделей наведено в таблиці 4.3.

Тестування на незнайомому датасеті показало деякі неочікувані результати : модель S, яка мала б демонструвати менш консервативну поведінку щодо позитивних результатів (на валідаційному датасеті значення confidence threshold для неї дорівнювало 0.345 проти 0.469 для моделі M) , натомість виявляла менше позитивних результатів.

Так, зокрема, нею було виявлено на 272 зображення з людьми, ніж модель M, з них було пропущено правильних спрацювань – 182, що становить відповідно 7,2% та 4,8% від розміру датасету відповідно. Також у обох моделей (але більше у моделі M) були помічені систематичні помилкові спрацювання на деякі ознаки: так, наприклад, предмети, схожі на літеру Λ (лямбду) часто визначалися як людина (див рисунок 4.22, випадки неправильної детекції вказані стрілками).

Існує припущення, що це пов’язано з тим, що на багатьох зображеннях з навчального датасету були присутні люди, які в профіль до камери стояли поставивши ноги на ширині плечей, або робили кроки. Інший приклад: предмети якраво – жовтого кольору прямокутної форми також розпізнавалися як люди (рисунок 4.22). Ймовірно, що мережа помилково узагальнила цю ознаку через те, що на багатьох зображеннях навчального датасету були присутні робочі в одязі, що мав яскраво-жовтий, або яскраво-зелений кольори, або на камерах нічного бачення (у відтінках сірого) створюв світлий прямокутний силует через наявність світловідбивних елементів.

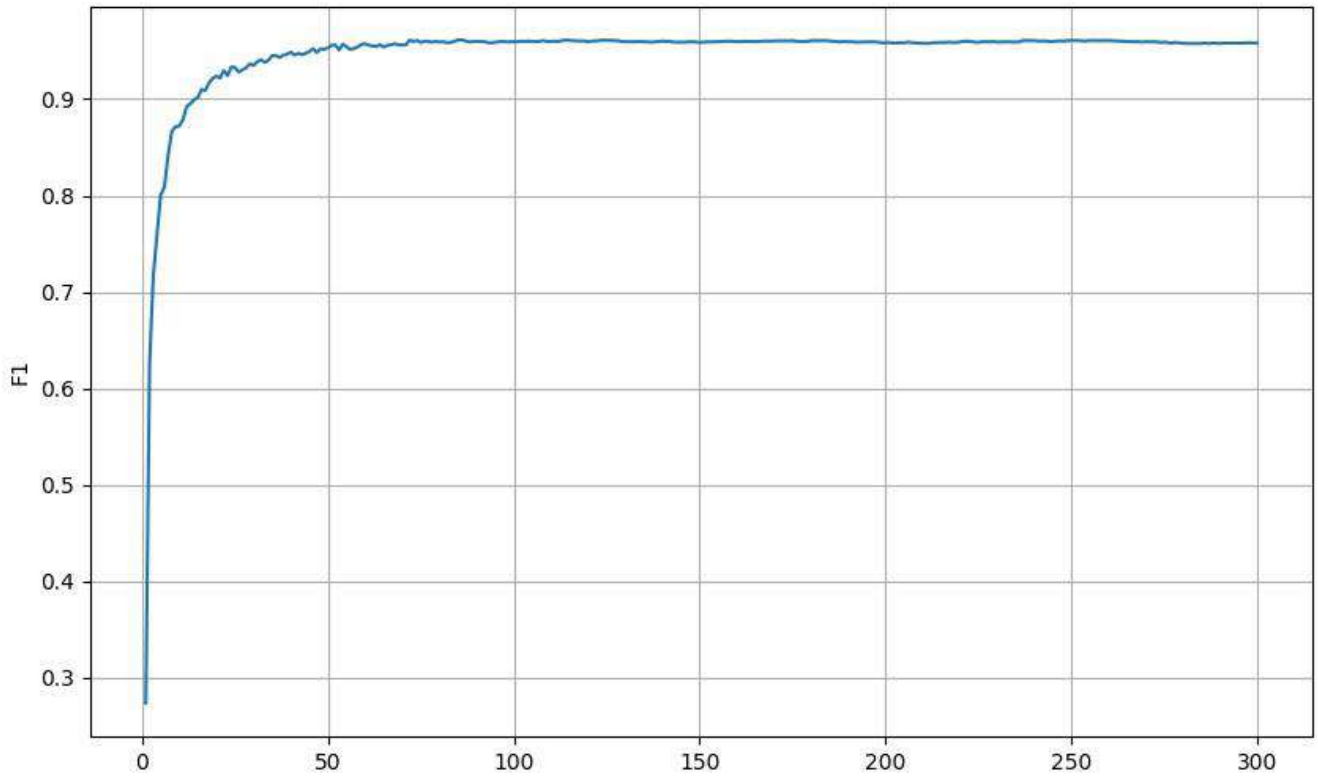


Рисунок 4.21 - Значення F1 протягом епох навчання для моделі M.

Наступними кроками у вдосконаленні моделі можуть бути:

- додавання в навчальний датасет як background – зображень (без жодних об’єктів), щоб довести їх долю в датасеті принаймні до 5%;

- додавання зображень із згаданими в попередньому абзаці ознаками (присутніми не в людей), принаймні 1% від розміру датасету, щоб перевчити нейромережу щодо цих ознак;

- додавання зображень з інших камер, в ідеалі – з усіх доступних. В підготовція навчальному датасеті було використано зображення з 370 камер, при наявному парку понад 1400 камер. Зусилля в даному напрямку слід докласти, тому що показники розпізнавання на незнайомих датасетах вочевидь потребують покращення;

- додаткові експерименти по навчанню нейромережі з вагами yolov8s, yolov8m, використовуючи різну кількість епох та розміру пакета (batch size), а за необхідності – і інші гіперпараметри.

Таблиця 4.3. Результати тестування моделей на незнайомому датасеті.

К-ть зображень в датасеті	К-ть зображень з похибками (Модель S)	К-ть зображень з похибками (Модель M)	% похибок (Модель S)	% похибок (Модель M)
3796	842	613	22,2	16,1

Рисунок 4.22 - Приклади неправильної детекції внаслідок неправильного узагальнення ознак нейронною мережею

4.3 Висновки

В даному розділі було спроєктовано архітектуру системи для обробки похибок отриманих відеозображень нейронною мережею. В процесі проєктування було сформовано діаграми, що описують різні аспекти роботи програмно-технічного засобу, обрано засоби для розгорнення, моніторингу та збереження даних для серверної частини додатку.

Також було проведено аналіз результатів експериментів, та оцінка точності обробки похибок отриманих відеозображень нейронною мережею.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено кіберфізичну систему комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею.

У першому розділі було проведено аналіз існуючих рішень у кіберфізичних системах комп’ютерного зору. Так, зокрема, було розглянуто сучасний стан галузі кіберфізичних систем, технології та сфери застосування комп’ютерного зору у кіберфізичних системах. Крім того, було проведено аналіз похибок, які виникають у кіберфізичних системах комп’ютерного зору. На завершення даного розділу, була здійснена постановка задачі та вибір технологій для реалізації кіберфізичної системи.

У другому розділі були розглянуті методи обробки похибок у кіберфізичних системах комп’ютерного зору. Також там було здійснено опис моделей нейронних мереж, які використовувалися в роботі. Було детально розглянуто підготовку даних до навчання нейронної мережі, а також налаштування моделей нейронних мереж для тренування.

У третьому розділі було детально описано метод та алгоритм навчання нейронної мережі для обробки похибок відеозображень. В процесі роботи над цим розділом також було розглянуто математичну модель роботи нейромереж сімейства YOLO, а також роботу алгоритму немаксимального пригнічення (non-max suppression algorithm, NMS), який використовується і в роботі інших нейромереж.

У четвертому розділі було спроєктовано архітектура системи для обробки похибок отриманих відеозображень нейронною мережею, і розроблено програмно-технічний засіб, базуючись на даній архітектурі. Також в цьому розділі було здійснено аналіз результатів експериментів, та проведено оцінку точності обробки похибок отриманих відеозображень нейронною мережею.

За темою дипломної роботи взято участь:

- у двох Всеукраїнських конференціях (АПКН-2023 у м. Хмельницький в листопаді 2023 року та IT&I у м. Миколаїв у лютому 2024 року) та опубліковано тези за матеріалами цих конференцій;
- у міжнародній конференції IntelITSIS-2024 та опубліковано статтю, яка індексується у наукометричній базі Скопус.

А також подано до друку тези для участі у Всеукраїнській конференції "Ольвійський форум" (22-24 червня м. Миколаїв).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Griffor E. , Greer C. , Wollman D., Burns, M. (2017). Framework for Cyber-Physical Systems: Volume 1, Overview, Special Publication (NIST SP). Gaithersburg, MD: National Institute of Standards and Technology. 66 c. DOI: https://doi.org/10.6028/NIST.SP.1500-201.

2. Greer C. , Burns M. , Wollman D., Griffor E. (2019). Cyber-Physical Systems and Internet of Things, Special Publication (NIST SP). Gaithersburg, MD: National Institute of Standards and Technology. 52 c. DOI: https://doi.org/10.6028/NIST.SP.1900-202.

3. Murthy C., Hashmi M., Bokde N., Geem Z. Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms—A Comprehensive Review. *Appl. Sci.* 2020. Vol. 10. Pp. 3280. DOI: https://doi.org/10.3390/app10093280.

4. Cyber-Physical Systems in the Context of Industry 4.0: A Review, Categorization and Outlook. URL: https://link.springer.com/article/10.1007/s10796-022-10252-x (дата звернення: 12.11.2023).

5. Cob-Parro A., Losada-Gutiérrez C., Marrón-Romera M., Gardel-Vicente A., Bravo-Muñoz I. Smart Video Surveillance System Based on Edge Computing. *Sensors* 2021.Vol. 21. Pp. 2958. DOI: https://doi.org/10.3390/s21092958.

6. Rizzoli A. An Introductory Guide to Quality Training Data for Machine Learning. URL: https://www.v7labs.com/blog/quality-training-data-for-machine-learning-guide (дата звернення: 15.11.2023).

7. Ultralytics YOLOv8 docs. Tips for Best Training Results. URL: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/ (дата звернення: 15.11.2023).

8. Common objects in context. Dataset overview. URL: https://cocodataset.org/#overview (дата звернення: 16.11.2023).

9. Rath S. Train YOLOv8 on Custom Dataset – A Complete Tutorial. URL: https://learnopencv.com/train-yolov8-on-custom-dataset/ (дата звернення: 18.11.2023).

10. Train Yolov8 object detection on a custom dataset . Step by step guide. URL: https://www.youtube.com/watch?v=m9fH9OWn8YM (дата звернення: 17.11.2023).

11. Maharana K., Mondal S., Nemade B. A review: Data pre-processing and data augmentation techniques. International Conference on Intelligent Engineering Approach (ICIEA-2022). Vol. 3. Issue 1. 2022. Pp 91-99. DOI: https://doi.org/10.1016/j.gltp.2022.04.020 (дата звернення: 10.11.2023).

12. Quality Control Data Representation Tools. URL: https://milestonetask.com/quality-control-data-representation-tools/ (дата звернення: 19.11.2023).

13. Terven J., Cordova-Esparza D. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. URL: https://arxiv.org/html/2304.00501v6/#S20.F21 (дата звернення: 19.02.2024).

14. Torres J. YOLOv8 Multi GPU: The Power of Multi-GPU Training. URL: https://yolov8.org/yolov8-multi-gpu/ (дата звернення: 17.03.2024).

15. Ghosh A. YOLOv9: Advancing the YOLO Legacy. URL: https://learnopencv.com/yolov9-advancing-the-yolo-legacy/#aioseo-what-is-yolov9 (дата звернення: 17.03.2024).

16. YOLOv8. Performance Metrics. 2024. URL: https://docs.ultralytics.com/models/yolov8/#performance-metrics (дата звернення: 19.04.2024).

17. Peiris Shashini. How can you evaluate data quality for larger datasets? 2024. URL: https://medium.com/@MoonlightO2/how-can-you-evaluate-data-quality-for-larger-datasets-65c74834a48e (дата звернення: 20.03.2024).

18. Wang C., Dong Q., Wang X., Wang H., Sui Z.. Statistical Dataset Evaluation: Reliability, Difficulty, and Validity. URL: https://arxiv.org/pdf/2212.09272 (дата звернення: 19.03.2024).

19. Dawn K. Comparing KerasCV YOLOv8 Models on the Global Wheat Data 2020. URL: https://learnopencv.com/comparing-kerascv-yolov8-models/ (дата звернення: 19.12.2023).

20. Samaha B. Measuring Agreement with Cohen's Kappa Statistic. URL: https://towardsdatascience.com/measuring-agreement-with-cohens-kappa-statistic-9930e90386aa (дата звернення: 20.03.2024).

21. YOLOv8. Understanding Settings. 2024. URL: https://docs.ultralytics.com/quickstart/#understanding-settings (дата звернення: 19.04.2024).

22. Mastering All YOLO Models from YOLOv1 to YOLO-NAS: Papers Explained. 2023. URL: https://learnopencv.com/mastering-all-yolo-models/ (дата звернення: 20.01.2024).

23. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition. Released October 2023 Publisher(s). O'Reilly Media, Inc. 861 pages. ISBN: 9781098125974.

24. Quesada A. 5 algorithms to train a neural network. 2023. URL: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network/ (дата звернення: 20.10.2023).

25. Karpathy A. A Recipe for Training Neural Networks. 2019. URL: https://karpathy.github.io/2019/04/25/recipe/ (дата звернення: 29.10.2023).

26. Zvornicanin E, Piwowarek G. Relation Between Learning Rate and Batch Size. 2024. URL: https://www.baeldung.com/cs/learning-rate-batch-size (дата звернення: 30.03.2024).

27. Jahangeer G., Rajkumar D. Early detection of breast cancer using hybrid of series network and VGG-16. 2021. URL: 10.1007/s11042-020-09914-2 (дата звернення: 03.11.2023).

28. Batch inference implementation using tensorrt #2 — converting to Batch model engine. 2023. URL: https://medium.com/@DeeperAndCheaper/yolov8-batch-inference-implementation-using-tensorrt-2-converting-to-batch-model-engine-e02dc203fc8b (дата звернення: 20.02.2024).

29. Maximizing Deep Learning Efficiency: The Ultimate Guide to Choosing the Perfect Batch Size. 2023. URL: https://metaprompting.de/en/ai/what-should-be-the-batch-size-in-deep-learning/ (дата звернення: 20.01.2024).

30. Adding background images to a classification dataset. 2023. URL: https://github.com/ultralytics/ultralytics/issues/4479 (дата звернення: 03.01.2024).

31. Şengönül E., Samet R., Abu Al-Haija Q., Alqahtani A., Alturki B., Alsulami A. An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey. *Appl. Sci.* 2023. Vol. 13. Pp 4956. DOI: https://doi.org/10.3390/app13084956

32. Технологія комп'ютерного зору: типові помилки під час розробки та впровадження. 2021. URL: https://brainberry.ua/uk/newsroom/blog/computer-vision-technology-common-mistakes (дата звернення: 20.09.2023).

33. AI-Based Visual Inspection For Defect Detection. 2020. URL: https://mobidev-biz.medium.com/ai-based-visual-inspection-for-defect-detection-ec11e91ea404 (дата звернення: 25.03.2024).

34. Martin Anderson. 10 Best Machine Learning Algorithms. 2022 . URL: https://www.unite.ai/ten-best-machine-learning-algorithms (дата звернення: 10.04.2024).

35. Kononova K. Машинне навчання: Методи та моделі. 2020. URL: https://www.researchgate.net/publication/345765254_MASINNE_NAVCANNA_METODI_TA_MODELI .(дата звернення: 20.01.2024).

36. Zvornicanin E., Piwowarek G. What Is YOLO Algorithm? 2024. URL: https://www.baeldung.com/cs/yolo-algorithm (дата звернення: 10.04.2024).

37. Aishwarya Singh. Selecting the Right Bounding Box Using Non-Max Suppression (with implementation). 2024. URL: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/ (дата звернення: 11.04.2024).

38. Rohit Kundu. F1 Score in Machine Learning: Intro & Calculation. 2022. URL: https://www.v7labs.com/blog/f1-score-guide (дата звернення: 12.04.2024).

39. Milos Simic. Intersection Over Union for Object Detection. 2024. URL: https://www.baeldung.com/cs/object-detection-intersection-vs-union (дата звернення: 15.04.2024).

40. Confusion matrix of YOLOv8. 2023. URL: https://medium.com/@a0922/confusion-matrix-of-yolov8-97fd7ff0074e (дата звернення: 05.03.2024).

41. Jodie Burchell. How to Prepare Your Dataset for Machine Learning and Analysis. 2022. URL: https://blog.jetbrains.com/datalore/2022/11/08/how-to-prepare-your-dataset-for-machine-learning-and-analysis/ (дата звернення: 01.02.2024).

42. Difference Between Scalar, Vector, Matrix and Tensor. 2024. URL: https://www.geeksforgeeks.org/difference-between-scalar-vector-matrix-and-tensor/ (дата звернення: 14.04.2024).

43. Steven Steinke . What's the difference between a matrix and a tensor? 2017. URL: https://medium.com/@quantumsteinke/whats-the-difference-between-a-matrix-and-a-tensor-4505fbdc576c (дата звернення: 02.02.2024).

44. Prabhakar Krishnamurthy. Understanding Data Bias. 2019. URL: https://towardsdatascience.com/survey-d4f168791e57 (дата звернення: 30.01.2024).

45. MPII Human Pose Dataset. URL: http://human-pose.mpi-inf.mpg.de/ (дата звернення: 10.02.2024).

46. TensorFlow datasets. URL: https://www.tensorflow.org/datasets/catalog/overview (дата звернення: 11.02.2024).

47. Google's Open Images. https://storage.googleapis.com/openimages/web/download.html (дата звернення: 11.02.2024).

48. Vision AI. Google Cloud. URL: https://cloud.google.com/vision (дата звернення: 16.02.2024).

49. PyTorch Faster R-CNN. URL: https://pytorch.org/vision/stable/models/faster_rcnn.html (дата звернення: 21.02.2024).

50. Melnychenko O., Savenko O. A Self-Organized Automated System to Control Unmanned Aerial Vehicles for Object Detection. The 4th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2023): CEUR-Workshop Proceedings. 3373, 2023. Pp.589-600.

51. Pavlova O., Kovalenko V., Hovorushchenko T. Neural network based image recognition method for smart parking. *Comput. Syst. Inf. Technol*. 2021. Vol.1. Pp. 49–55.

52. Zeng Y., Zhang, J. A machine learning model for detecting invasive ductal carcinoma with Google Cloud AutoML Vision. Computers in biology and medicine.2020. Vol. 122. 103861. DOI: https://doi.org/10.1016/j.compbiomed.2020.103861.

53. Lu Y., Jia Y., Wang J., Li B., Chai W., Carin L., Velipasalar S.. Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. CVF conference on Computer Vision and Pattern Recognition. 2020. Pp. 940-949.

54. Radiuk P., Pavlova O., Hrypynska N. An Ensemble Machine Learning Approach for Twitter Sentiment Analysis. CEUR Workshop Proceedings. 2022. Vol 3171. Pp. 387–397.

55. Sahoo J. P., Prakash A. J., Pławiak P., Samantray S. Real-time hand gesture recognition using fine-tuned convolutional neural network. *Sensors*. 2022. Vol.22. Issue3. https://www.mdpi.com/1424-8220/22/3/706.

56. Gazda M., Hireš M., Drotár P. Multiple-fine-tuned convolutional neural networks for Parkinson's disease diagnosis from offline handwriting. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2022 Vol.52. Part 1. 78-89.

57. Radiuk P., Pavlova O., El Bouhissi H., Avsiyevych V., Kovalenko V. Convolutional Neural Network for Parking Slots Detection. CEUR Workshop Proceedings. 2022. Vol. 3156. Pp. 284–293.

58. Bussa S., Mani A., Bharuka S.,Kaushik S. Smart attendance system using OPENCV based on facial recognition. *Int. J. Eng. Res. Technol*. 2020.Vol. 9. Issue 3. Pp. 54-59.

59. Ismail A., Abd Aziz F., Kasim N., Daud K. Hand gesture recognition on python and opencv. *IOP Conference Series: Materials Science and Engineering*. 2021. Vol. 1045. P. 012043.

60. Kushal M., Pappa M. ID Card Detection with Facial Recgnition using Tensorflow and OpenCV. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA).* 2020. Pp. 742-746.

61. Sharma A., Shah K., Verma S. Face recognition using haar cascade and local binary pattern histogram in opencv. *2021 Sixth International Conference on Image Information Processing (ICIIP)*. Vol. 6. Pp. 298-303.

62. Tu-Liang L., Hong-Yi C., Kai-Hong C. The pest and disease identification in the growth of sweet peppers using faster R-CNN and mask R-CNN. *Journal of Internet Technology*. 2020. Vol.21. Issue 2. Pp. 605-614.

63. Mahmood T., Arsalan M., Owais M., Lee M., Park K. Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster R-CNN and deep CNNs. *Journal of clinical medicine*. 2020. Vol. 9. Issue 2. P.749.

64. Palanivel N. Automatic number plate detection in vehicles using faster R-CNN. *2020 International conference on system, computation, automation and networking (ICSCAN)*. Pondicherry, India, 2020. Pp. 1-6.

65. Wan S.; Goudos S. Faster R-CNN for multi-class fruit detection using a robotic vision system. *Computer Networks*. 2020, Vol.168. P.107036.

66. Dhufr A., and Seydi K. Facial Features Recognition Based on Their Shape and Color Using YOLOv8.*7th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. Ankara, 2023. Pp.1-6.

67. Safran M., Alajmi A., Alfarhood S. Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems. *Journal of Sensors*. 2024. DOI: https://doi.org/10.1155/2024/4917097.

68. Xiao B., Nguyen M, Yan W. Fruit ripeness identification using YOLOv8 model. *Multimedia Tools and Applications*. 2023. Vol.83. Pp. 28039–28056. DOI: http://dx.doi.org/10.1007/s11042-023-16570-9.

69. Chabi A., Sanda M., Gouton P., Tossa, J. Automatic localization of five relevant dermoscopic structures based on YOLOv8 for diagnosis improvement. *Journal of Imaging*. 2023. Vol. 9. P.148. DOI: https://www.mdpi.com/2313-433X/9/7/148.

70. Melnychenko O., Savenko O., Radiuk P. Apple Detection with Occlusions Using Modified YOLOv5-v1. *12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Dortmund, Germany, 2023, pp. 107-112. DOI: https://doi.org/10.1109/IDAACS58523.2023.10348779.

71. Авсієвич В., Коваленко В. Аналіз інформаційних технологій для розумної парковки на основі штучних нейронних мереж. *Актуальні Проблеми Комп'ютерних Наук (АПКН-2021)*. (ХНУ, 15-16 жовтня 2021). Хмельницький,2021. С. 12-14.

72. Кузьмін А., Павлова О. Застосування комп'ютерного зору для кіберфізичної системи розумної парковки. *Інформаційні технології та інженерія:* тези доп. всеукр. наук.-практ. конф.(ЧНУ імені Петра Могили, 7–10 лютого 2023 р.). Миколаїв, 2023. С.45-46.

73. Pavlova O., Rudyk I, Bouhissi H. Post-processing of video surveillance systems alarm signals using the YOLOv8 neural network. IntelITSIS-2024: CEUR-Workshop Proceedings. 2024. Vol. 3675. Pp.196-207.

ДОДАТОК А
(обов'язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лістинг файлу приймання вебхуків модулю обміну файлами з ПУВ підсистеми розпізнавання об'єктів.

```
import os
from flask import Flask, request, jsonify, send_from_directory  #
Add 'send_from_directory'

app = Flask(__name__)

@app.route('/get_hook', methods=['POST'])
def get_hook():
    # Get the incoming text variable from the webhook request
    webhook_data = request.get_json()
    received_text = webhook_data.get('text')  # Adjust the key based
on the actual data structure

    # Clean and update the text file
    with open('source.txt', 'w') as file:
        file.write(received_text)

    return jsonify({'message': 'Webhook processed, text updated, and
other script started'}), 200

@app.route('/video/<filename>')
def serve_video(filename):
    video_directory = os.path.join(app.root_path, 'videos')  #
'videos' folder in the same directory as the script
    return send_from_directory(video_directory, filename)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Лістинг модулю підготовки датасету.

```
import cv2
import time
import csv
from collections import defaultdict
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('best_m.pt')
```

```
def perform_object_detection(source_video, starting_value,
target_fps, unit_name):
    cap = cv2.VideoCapture(source_video)
    source_fps = int(cap.get(cv2.CAP_PROP_FPS))
    target_fps = min(target_fps, source_fps)
    if target_fps == 0:
        target_fps = 1

    frame_interval = max(1, int(source_fps / target_fps))

    frame_count = 0
    frame_count1 = 0
    total_frames_analyzed = 0
    frames_with_person = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        if frame_count1 % frame_interval == 0:
            results = model(frame)
            total_frames_analyzed += 1

            persons_info = []

            for idx, prediction in
enumerate(results[0].boxes.xywhn):
                cls = int(results[0].boxes.cls[idx].item())
                class_name = model.names[cls]

                if class_name == 'person':
                    persons_info.append(
                        (cls, prediction[0].item(),
prediction[1].item(), prediction[2].item(), prediction[3].item()))

            if persons_info:
                frames_with_person += 1
                frame_count += 1
                save_path_frame =
f"frames/{starting_value}_{frame_count}.jpg"
                annotated_frame = results[0].plot()
                cv2.imwrite(save_path_frame, frame)
                save_path_frame1 =
f"frames/{starting_value}_{frame_count}_a.jpg"
                cv2.imwrite(save_path_frame1, annotated_frame)
                save_path_label =
f"labels/{starting_value}_{frame_count}.txt"
                with open(save_path_label, 'w') as label_file:
                    for person_info in persons_info:
                        label_file.write(
                            f"{person_info[0]} {person_info[1]}
{person_info[2]} {person_info[3]} {person_info[4]}\n")
```

```
        frame_count1 += 1

    cap.release()

    if total_frames_analyzed > 0:
        percentage_with_person = (frames_with_person /
total_frames_analyzed) * 100
    else:
        percentage_with_person = 0

    # Log video processing details, now including unit name
    with open('video_processing_log.txt', 'a') as log_file:
        log_file.write(
            f"Video File: {source_video}, Total Fr. Analyzed:
{total_frames_analyzed}, Fr. with 'Person': {frames_with_person}, %
of 'Person': {percentage_with_person:.2f}%, Unit Name: {unit_name},
Video_ID: {starting_value}\n")


if __name__ == "__main__":
    starting_row_index = 2
    max_rows = 55
    max_rows_per_unit = 2

    unit_counts = defaultdict(int)

    with open("permissions.csv", "r", encoding="utf-8-sig") as
csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')

        for _ in range(starting_row_index - 1):
            next(csv_reader, None)

        for row in csv_reader:
            if len(row) != 3:
                print("Invalid row format. Skipping.")
                continue

            unit_name = row[2].strip()
            if unit_counts[unit_name] >= max_rows_per_unit:
                print(f"Row skipped due to limit for unit
{unit_name}")
                continue

            unit_counts[unit_name] += 1

            source_content =
f"https://custom.mysource.dk{row[1].strip()}"
            print(f"source_content = {source_content}")
            starting_value = row[0].strip()
            print(f"starting_value = {starting_value}")
```

```
            if source_content:
                target_fps = 1
                perform_object_detection(source_content,
starting_value, target_fps, unit_name)
                max_rows -= 1
                if max_rows <= 0:
                    break

    print(f"The task {starting_value} is finished")
    time.sleep(0)
```

Лістинг файлу відправки вебхуків модулю обміну файлами з ПРО підсистеми управління відеоданими.

```
<?php
error_reporting(4096);
ini_set('max_execution_time', 15);
     define("_PATH_TO_CONFIG", 'configuration');//PATH:
configuration dir with out end slash
     $err = "";

     if (!isset($_pathPrefix)) $_pathPrefix='../../../'; // PATH
PREFFIX FOR SUBDOMAINS AND SCRIPTS
     require_once $_pathPrefix._PATH_TO_CONFIG."/kernal.php"; //
INCLUDE KERNAL
     require_once $_pathPrefix._PATH_TO_CONFIG."/engine.php"; //
INCLUDE ENGINE
     require_once $_pathPrefix._PATH_TO_INCLUDES."/functions.inc";

     {
          global $base_URL;
     }

     include "module.nfo";
     date_default_timezone_set('Europe/Berlin');

echo ('its working<br>');

$fe=get_one("select id from ai_stack where proc_now=1");
if(intval($fe)>0) {
     echo ('Processing cycle is busy. program is terminated. ');
     exit(); //means, jetson is busy now
}
else {
     $fa = get_row("select id, path, video_id from `ai_stack` where
id=(select min(id) from `ai_stack` where done=0)");

     if(!isset($fa)) exit(); //if no acceptable rows - terminate
execution
     $file = '/home/sitesecurity/public_html/alarmcentral'
.$fa['path'];
```

```
     $sz = filesize($file);



     if($sz<=20000 || !file_exists($file)){//probably, file is
corrupted, stop this
          $query="INSERT INTO `aiproc_log` (event_id, is_cor) VALUES
(".$fa['video_id']."", 1 )";
          DB_query($query);
          $query="UPDATE `video` SET ai_red=0 WHERE
id=".$fa['video_id']; //return to processing by operator
          DB_query($query);
          $query="UPDATE `ai_stack` SET done=1 WHERE id=".$fa['id'];
          DB_query($query);
          $command = "php calljet5.php > /dev/null 2>&1 &";
          $output = shell_exec($command);
          exit();
     }

     $textVariable =
"https://alarmcentral.sitesecurity.dk".$fa['path'];
     $data = array(
          'text' => $textVariable
     );
     $flaskServerUrl = 'http://192.19.123.124:5000/get_hook'; //
Update with your Flask server URL
     $ch = curl_init($flaskServerUrl);
     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
     curl_setopt($ch, CURLOPT_POST, true);
     curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));
     curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/json'));

     $response = curl_exec($ch);
                         $log = date('Y-m-d H:i:s').'
response='.$response.chr(10);
                         file_put_contents(__DIR__ .
'/response_log.txt', $log . PHP_EOL, FILE_APPEND);    //logging

     if ($response === false) {
          echo 'Error sending request: ' . curl_error($ch);
          $query="UPDATE `video` SET ai_red=0 WHERE
id=".$fa['video_id']; //return to processing by operator
          DB_query($query);
          $query="UPDATE `ai_stack` SET done=1 WHERE id=".$fa['id'];
          DB_query($query);
          $query="INSERT INTO `aiproc_log` (event_id, sent_tm,
is_cor) VALUES (".$fa['video_id']."", ".time()."", 3 )";
          DB_query($query);
     } else {
          echo 'Request sent successfully! Response: ' . $response;
          $query="UPDATE `ai_stack` SET proc_now=1 WHERE
id=".$fa['id'];
```

```
          DB_query($query);

          $query="INSERT INTO `aiproc_log` (event_id, sent_tm) VALUES
(".$fa['video_id']."", ".time()." )";
          DB_query($query);

          $query="UPDATE `video` SET dt_proc_started=Now() WHERE
id=".$fa['video_id'];
          DB_query($query);

     }

     curl_close($ch);
}
?>
```

.....

ДОДАТОК Б
КОПІЇ ТЕЗ ДОПОВІДІ НА КОНФЕРЕНЦІЯХ ТА ПУБЛІКАЦІЇ В ФАХОВОМУ НАУКОВОМУ ВИДАННІ

Тези доповіді на Всеукраїнській науковій конференції "Актуальні Проблеми Комп'ютерних Наук" АПКН-2023

УДК 004.4

Павлова О.О., Рудик І.В.

Хмельницький національний університет

ПОСТ-ОБРОБКА СИГНАЛІВ ТРИВОГИ СИСТЕМ ВІДЕОСПОСТЕРЕЖЕННЯ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ YOLOV8

Розглянуто проблему помилок розпізнавання об'єктів на зображеннях з камер зовнішнього спостереження. Запропоновано рішення після-обробки сигналів тривоги систем відеоспостереження за допомогою нейромережі YOLOv8.

The problem of object recognition errors in images from outdoor surveillance cameras is considered. A solution for post-processing alarm signals of video surveillance systems using the YOLOv8 neural network is proposed.

Наразі застосування камер зовнішнього спостереження для забезпечення безпеки на території певних об'єктів, підприємств та організацій є досить поширеною практикою. Проте, якщо територія огляду є великою, людина фізично не може безперервно стежити за усіма моніторами. Для цього використовують штучну нейронну мережу для автоматизованого розпізнавання об'єктів на відео з камери.

Проте, таке розпізнавання не завжди дає точний результат. Розглянемо таку ситуацію на прикладі фірми, яка спеціалізується на наданні комплексних послуг з відеоспостереження. Щодня їхні оператори отримують сигнали про порушення з декількох десятків камер. Середньодобова кількість сигналів становить майже 3000 (Більше 86 тис. за жовтень 2023 року). Слід зазначити, що переважна кількість камер знаходиться зовні, і зазнають значного впливу різних чинників, що спричиняють фальшиві спрацювання. Ці чинники можна об'єднати в 3 категорії : опади (дощ, сніг), вітер (розхитує об'єкти в полі зору камери, або саму камеру), світлові впливи (мерехтіння світла на території або за її межами, світло фар авто – в нічний час, відблиски від сонця, або просто тінь від хмари – в денний). Всі згадані чинники, а частіше – їх комбінації – мають значний вплив на фальшиві спрацювання камер. Так, за згаданий місяць кількість фальшивих спрацювань сягнула 44 тис, тобто понад 51% від усіх сигналів тривоги. На рисунку 1 (а, б, в) представлено зображення з камер з помилками розпізнавання об'єктів. Зважаючи на те, що кожен сигнал тривоги має бути перевірений людиною – фальшиві сигнали тривоги спричиняють значні збитки компаній, змушуючи утримувати на роботі збільшену кількість операторів.

Щодо рішень, які вже були запропоновані для вирішення цієї проблеми – це збільшення кількості камер, введення в експлуатацію новітніх систем аналізу потокового відео. Проте наразі це не дало дієвого результату.

а) б)

в)

Рисунок 1 – Помилки розпізнавання об'єктів на зображеннях з камер зовнішнього спостереження

Як можна побачити з рисунку 1, у більшості випадків нейромережа помилково розпізнає наявність людини на зображенні, приймаючи тонкий вертикальний об'єкт за особу.

Тому було прийнято рішення застосувати найновішу версію нейронної мережі YOLOv8 [1,2] для того, щоб навчити нейромережу на реальних зображеннях з камер зовнішнього спостереження з метою підвищення точності розпізнавання об'єктів у полі зору камер.

Таким чином, алгоритм дій буде наступним:

1. Зібрати реальні датасети з камер зовнішнього спостереження для навчання нейромережі на реальних об'єктах, які знаходитимуться в полі зору даних

камер (об'єкти, які найчастіше є на підприємстві – автомобіль, бульдозер, вантажівка, вантажний контейнер тощо);

2. Зібрати реальні датасети з камер зовнішнього спостереження для навчання нейромережі на предмет «людина – не людина», а саме зображення, де дійсно присутні особи та де нейромережа помилково розпізнала певні предмети як людину (рисунок 1 а, б, в);

3. Провести навчання нейромережі та експерименти щодо успішності чи неуспішності навчання та чи є підвищення якості розпізнавання після перенавчання нейромережі, у порівнянні із початковими результатами.

Для навчання було обрано нейромережу YOLOv8[1, 2]. На це будуть спрямовані зусилля авторів у подальших дослідженнях на тему пост-обробки сигналів тривоги систем відеоспостереження.

Перелік посилань

1. Ultralytics YOLOv8 Docs URL: https://docs.ultralytics.com/ (Доступ 2.10.2023)
2. Ultralytics on GitHub.com URL: https://github.com/ultralytics/ultralytics/blob/main/docs/models/yolov8.md (Доступ 2.10.2023)
3. Radiuk, P., Pavlova, O., El Bouhissi, H., Avsiyevych, V., Kovalenko, V. Convolutional Neural Network for Parking Slots Detection. CEUR Workshop Proceedings 2022, 3156, pp. 284–293

Тези доповіді на Всеукраїнська науково-практична конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія» IT&I-2024

УДК 004.4

Павлова О. О., Рудик І. В.
Хмельницький національний університет,
м. Хмельницький, Україна

МЕТОД ОБРОБКИ ПОХИБОК РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ З КАМЕР ЗОВНІШНЬОГО СПОСТЕРЕЖЕННЯ МЕРЕЖЕЮ YOLOV8

В [1] було розглянуто проблему виникнення похибок розпізнавання об'єктів на зображеннях з камер зовнішнього спостереження.

У даному дослідженні наводяться результати експериментів щодо розпізнавання об'єктів у відеофрагментах, що надсилалися камерами спостереження після спрацювання. Було використано нейронну мережа YOLO-8 і модель YOLOV8X.pt (модель була попередньо тренована на наборі COCO виробником мережі).

Класи об'єктів та їх відповідність класам моделі COCO, і пріоритетність їх виявлення наведені в табл. 1.

Таблиця 1 – Класи об'єктів, які потрібно розпізнати на зображеннях з камер зовнішнього спостереження

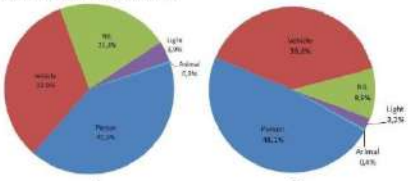
Пріоритет	Клас об'єктів	Класи моделі
1	Person	'person'
2	Vehicle	'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat'
3	Animal	'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe'
4	Light	'traffic light'
5	NIL	жоден з класів, наведених вище, не був виявлений

Роботу алгоритму в частині виявлення в кліпі об'єктів класів можна звести до кількох пунктів:

1) здійснювався пошук об'єктів у кожному фреймі відеокліпа;
2) віднесення кліпу до певного класу здійснювалося від

найвищого (1) до найнижчого (5) пріоритету. Тобто, наприклад, якщо в кліпі було виявлено принаймні один об'єкт пріоритету 1 (Person) – кліп відносився до цього класу незалежно від кількості об'єктів нижчих класів. Аналогічно, якщо не було виявлено об'єктів пріоритету 1, тоді розглядалася наявність об'єктів пріоритету 2 тощо.

Фактичний розподіл кліпів по класах не відповідає встановленій пріоритетності. Розподіл по класах на прикладі денної та тижневої вибірки наведено на рис. 1, а, б.



а) б)

Рисунок 1 – Розподіл по класах: а – денна вибірка (1440 кліпів); б – тижнева вибірка (11247 кліпів)

Оскільки основною метою відеоспостереження, що проводить фірма замовник, є охорона об'єктів, то на даному етапі було прийняте рішення додатково верифікувати результати, де були виявлені люди (клас Person). Верифікація здійснюватиметься операторами-людьми. На це будуть спрямовані подальші дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Павлова О. О., Рудик І. В. Пост-обробка сигналів тривоги систем відеоспостереження за допомогою нейромережі YOLOv8. *XV-та Всеукраїнська наук.-практ. конф. «Актуальні проблеми комп'ютерних наук АПКН-2023»* : зб. наук. праць, Хмельницький, 2023. С. 242–245.

Копія англомовної статті по матеріалах міжнародної конференції IntelITSIS-2024

Post-processing of video surveillance systems alarm signals using the YOLOv8 neural network

Olga Pavlova[1,*,†], Ivan Rudyk[2,†] and Houda El Bouhissi[2]

[1] Khmelnytskyi National University, Instytutska str., 11, Khmelnytskyi, 29016, Ukraine
[2] LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, 06000, Bejaia, Algeria

Abstract

The method of solving the security problem of a warehouse equipped with external surveillance cameras by processing the video stream using artificial neural networks is considered. Experiments were conducted to test already existing pre-trained models and the model that gave the highest recognition quality - YOLOv8 was determined. A dataset of images taken from outdoor surveillance cameras was also compiled for training, validation and experiments. It was proven that the YOLOv8 neural network model currently copied best with the given task, so it will be used for further experiments. However, upon manual verification, it was observed that the proportion of objects identified with errors by the newly developed model decreased to 10.7%. The obtained metrics reflect the success of the training process, as evidenced by improvements in parameters such as Train Box Loss (reduced to 0.5135) and mAP50 (increased to 0.98367) with each successive epoch. However, the relatively stable Validation Box Loss value (0.69291) from epoch 84 onward suggests inherent performance fluctuations possibly due to the limited size of the validation sample (6% of the training sample).

Keywords
video-image processing, pattern recognition, objects detection, neural networks, YOLOv8

1. Introduction

Companies that specialize in providing comprehensive video surveillance services often face the problem of pattern recognition quality. Every day, their operators receive signals about violations from more than 400 cameras. The average daily number of signals is almost 3,000 (more than 86,000 for the past month). It should be noted that the majority of cameras are located outside and are significantly affected by various factors that cause false alarms. These factors can be divided into categories:

IntelITSIS'2024: 5th International Workshop on Intelligent Information Technologies and Systems of Information Security, March 20, 2024, Khmelnytskyi, Ukraine
* Corresponding author.
† These authors contributed equally.
pavlovaod@khmnu.edu.ua (O. Pavlova); novelmyst@gmail.com (I. Rudyk); houda.albouhissi@gmail.com (H. El Bouhissi)
0000-0003-2905-0215 (O. Pavlova); 0009-0009-8355-3471 (I. Rudyk); 0000-0003-3239-8255 (H. El Bouhissi)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1) precipitation (rain, snow);
2) wind (shakes objects in the field of view of the camera, or the camera itself);
3) light effects (flickering of light on the territory or outside it, car headlights at night, glare from the sun, or just a shadow from a cloud during the day).

All the mentioned factors, and more often – their combinations – make up a significant part of all camera activations. So, for the mentioned last month, the number of false alarms reached 44 thousand, that is, more than 51% of all alarm signals. Considering that each alarm must be verified by a person - false alarms cause significant losses to the owner, forcing to keep an increased number of operators on the job. Figure 1 shows examples of errors in pattern recognition on images from outdoor surveillance cameras (a) false recognition of a truck and a person driving a bulldozer; b) false recognition of a refrigerator and a truck at the place of cargo containers; c) false identification of a truck and a person at the place of the cargo container and demarcation column; d) false recognition of trucks in the place of minibuses and a person in the place of a limiting partition).

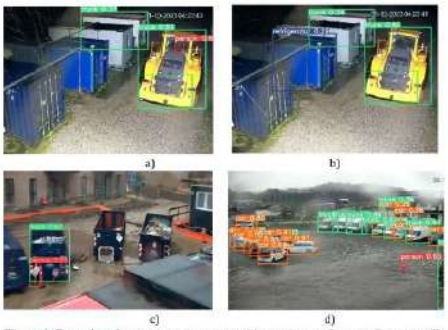


Figure 1: Examples of errors in pattern recognition on images from outdoor surveillance cameras.

Since external surveillance cameras are intended mainly for enterprise security purposes, namely, to prevent illegal entry into the territory of outsiders and vehicles, *the problem of high-quality recognition of patterns on the video stream from the cameras is quite important* for the automation of the operator's work.

2. Related works

In the course of the study, an analysis of the latest scientific publications in the field of pattern recognition using artificial neural networks was carried out. Scientific publications devoted to the application of models based on artificial neural networks such as Google Cloud Vision API, Pytorch Faster R-CNN, OpenCV+CNN and YOLO libraries for various fields such as biology, medicine, smart cities and cyber-physical systems, recognition of gestures and facial expressions were reviewed. The results of scientific publications analysis are presented in Table 1.

Table 1
Analysis of ready existing computer vision approaches for pattern recognition

Reference	Year	Algorithm / Model	Scope of application	Brief Description of the approach
1	2	3	4	5
Pavlova O. et al. [1]	2021	CNN	Smart parking system	The research in general is aimed at image recognition for camera-based smart parking using convolutional neural network (CNN).
Zeng Y. et al. [2]	2020	Google Cloud AutoML Vision	Medicine. Early diagnosis of carcinoma.	The paper assesses the feasibility of an AutoML approach for the identification of invasive ductal carcinoma (IDC) in whole slide images (WSI). An experimental IDC identification model is built with Google Cloud AutoML Vision.
Liu Y. et al. [3]	2020	Google Cloud Vision (GCV) APIs	Cybersecurity	The transferability of adversarial examples across a wide range of real-world computer vision tasks, including image classification, object detection, semantic segmentation, explicit content detection, and text detection were investigated.
Radiuk P. et al. [4]	2022	An Ensemble of Fine-tuned NNs	statistical analysis	The classification task of emotional expressions was performed according to several machine learning algorithms: raw random forest, gradient boosting random forest, support vector machine, multilayer perceptron, recurrent neural network, and convolutional neural network
Saleem J. et al. [5]	2022	pre-trained CNN model with score-level fusion technique	Hand gesture recognition	A real-time American sign language (ASL) recognition system is developed and tested using the proposed technique.
Gazda M. et al. [6]	2021	multiple-fine-tuned CNNs	Medicine. Parkinson's Disease Diagnosis	Multiple-fine-tuned convolutional neural networks for Parkinson's disease diagnosis from offline handwriting
Radiuk P. et al. [7]	2022	Google Cloud Vision OpenCV	Parking slots detection	Google Cloud Vision technology as parking slots detector and a pre-trained convolutional neural network as a feature extractor and a classifier were selected to develop a cyber-physical system for smart parking.
Bussa S.et al. [8]	2020	OpenCV	Face recognition	Smart attendance system using OpenCV based on facial recognition.
Safran M. et al. [17]	2024	YOLOv8 and CNN	License plate recognition	Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems
Melnychenko O. et al. [20]	2023	YOLOv5-v1	Apples recognition	A novel architecture to identify apples in images with occlusions was created.

As can be seen from Table 1, a deep learning approach, particularly CNNs, has been most frequently used over the past five years and has shown the most robust recognition of individual objects, parking slots, faces, emotions, gestures, medical patterns etc. Therefore, considering the abovementioned analysis, it was decided to apply the pre-trained model for objects captured by the surveillance cameras recognition.

3. Dataset preparation and model selection for the implementation

The search for a solution has been ongoing for almost 10 years. The number of cameras is increasing, the latest streaming video analysis systems are being put into operation. However, the quality of automated image recognition that could satisfy the company's security parameters and eliminate the human operator from the process of monitoring the cameras was not achieved.

Based on the above analysis of existing solutions, it was decided to collect a dataset of 12 test images and conduct a comparative analysis on the quality of pattern recognition. Google Cloud Vision API [22], Pytorch FasterR-CNN [23] and YOLOv8 neural network [27] were

chosen for testing. The dataset that was collected and prepared for models testing is presented in Figure 2. It contains images in infrared light, in shades of gray, on which there are images of various objects on a construction site, cars, people.

Figure 2: A dataset of target images for models testing.

Since the target objects for recognition in the images are people, the focus of the experiment was on recognizing the images of people and establishing the presence of people. According to the results of testing the models [22, 23, 27], a table was compiled with the effectiveness of recognizing people's images and with the available advantages and disadvantages of each of the models. The test results are presented in Table 3.

Table 3
The results of the neural networks models testing using the pre-prepared dataset

Criterion	Google Cloud Vision API	YoloV8	PyTorch Faster R-CNN
Cases of count people matched	1	8	8
Found bigger count of people than were presented really	2	1	2
Found fewer count of people than were presented really	9	3	2
% of matched	8,3	66,7	66,7
Advantages	Good integration with other Google services	User-friendly API, high accuracy	High accuracy
Disadvantages	Low accuracy	-	Complicated API (compared to others)

As can be seen from Table 3, the result of the analysis showed that the YoloV8 model coped best with pattern recognition on the test dataset, so it was decided to build further work on its basis.

4. Image recognition using the YOLOv8 neural network

Below is the experience of object recognition in video clips sent by surveillance cameras after they were triggered. It should be noted that cameras were chosen for analysis, where, according to expectations, there should be no people at the time of observation.

By conducting the experiment, the minimum value of fps = 12 in the video stream was selected, at which the model based on the artificial network clearly captured the objects captured by the thermal cameras. On thermal cameras, it is especially difficult to recognize silhouettes of people, because the range on cameras is from 1 to 100 fps, and videos are processed frame by frame. That is, for a 10-second video at fps=100 and the processing time of one frame t=170-175 ms, the processing time of the video stream increases to 175 seconds. Therefore, by the formula (1) parameter N was calculated. N - interval in source video between frames that should be copied to target video, to get desired fps. Means, for example: if fps in source video is 100, and desired fps is 25 - in this case each 4-th (100/25) frame from source video to target video - 4th, 8th, 12th..., should be taken, others frames should be skipped.

$$N = round\,(fps(source)/fps(target)). \quad (1)$$

A Python program was created that uses the YOLOv8 neural network and the yolov8x.pt model (the model was pre-trained on the COCO set by the network manufacturer) [24]. The classes of objects of interest, their correspondence to the classes of the COCO model, and the priority of their detection are shown in Table 4.

Table 4
The correspondence of objects of interest to the classes of the COCO model

Priority	Custom class	Classes of COCO model
1	Person	'person'
2	Vehicle	'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat'
3	Animal	'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe'
4	Light	'traffic light'
5	NIL	none of the above classes were detected

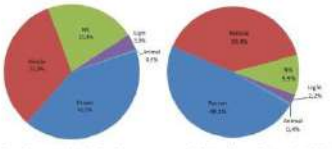
The operation of the algorithm in the part of detecting objects of the customer's classes in the clip can be reduced to several points:

1. Objects were searched for in each frame of the video clip. Search results for each clip were saved and statistics were recorded in a table. A sample of training statistics is presented in Table 5.
2. Assigning a clip to a certain class was carried out from the highest -1 to the lowest - 5 priority. That is, for example, if at least one object of priority 1 (Person) was detected in the clip, the clip belonged to this class regardless of the number of objects of lower classes. Similarly, if no objects of priority 1 were detected, the presence of objects of priority 2 was considered, etc.

Table 5
A sample of training statistics

Frame Number	Object COCO code	Object Name	Confidence
1	0	person	0.5774214267730713
1	13	bench	0.4358726441860199
2	0	person	0.5948778986930847
2	13	bench	0.4844166067660025024
3	0	person	0.5487728118896484
3	13	bench	0.454770326614379
4	13	bench	0.5063151121139526
4	0	person	0.4895791709423065
4	0	person	0.25702717900276184
5	13	bench	0.49720075726909094

The model was trained according to the classes listed in Table 1. The actual distribution of clips by classes on the example of a daily (a) and weekly (b) selection is shown in the diagrams in Figure 3.



a) Daily selection (1440 clips) b) Weekly selection (11247 clips)

Figure 3: The actual distribution of clips by classes on the example of a daily (a) and weekly (b) selection.

In order to improve the quality of recognition, it was decided to train the model for the "person" class. To create a dataset with the "human" class, the authors selected video clips from 153 cameras. Object recognition in video clips was used to mark objects in images, where the presence of people was determined by a human operator beforehand. The largest of the ready-made models from the YOLOv8 neural network developer of the Ultralitics company was used as a model: yolo8x.pt [26,27]. During the preparation of the dataset, 4908 images containing 6121 objects of the "person" class were obtained, all of which were manually checked for possible neural network errors before the start of training. The number of background images (which did not contain objects of the "person" class) - 92.

All received images were reviewed, when recognition errors were detected, label files were created for such images manually (on a local computer, using the Labelimg program). For the training dataset, 4908 images containing 6121 objects of the "person" class were obtained. The number of background images (which did not contain any objects) is 92. The percentage of detected neural network errors is 19.4%. The number of prepared images for verification is 300. Training was carried out on Google Collab data was downloaded from Google Drive. 120 learning epochs were launched. Training was started with empty weights for the yolo8s model. Tesla V-100 GPU was used, each cycle lasted about 2 min. Considering that the version "from the box" was trained on 640px images, and the size of the video from the cameras can also be 704, 1920 - the decision was made in the training parameters to use a size of 1280, to provide better detail.

5. Experiments and Results

Since the main purpose of the video surveillance conducted by the client company is the protection of objects, at this stage it was decided to additionally verify the results where people were detected (Person class). Verification was carried out by human operators. The results of videos verification in which the neural network detected people (a sample of 4837 clips over 6 days) are shown in Figure 4 in the form of a diagram. Only 0.7% of the total number of clips identified by the program as Person were confirmed by the operators. That is, the error of the second kind was 99.3%. The number of Person objects detected by the neural network in each video clip varies from 1 to 373 (it should be noted that the total number of frames in the clips is from 8 to 900). The frequency with which objects of the Person class occur in files is plotted in Figure 5.

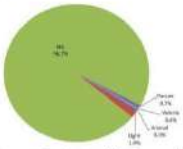


Figure 4: The results of videos verification in which the neural network detected people.

Count of Person objects detected in clips	Count of the files (video-clips)
1	1248
2	972
3	414
4	273
5	203
6	133
7	109
8	109
9	127
10	101
...	...
230	1
385	1
280	1
372	1
372	1

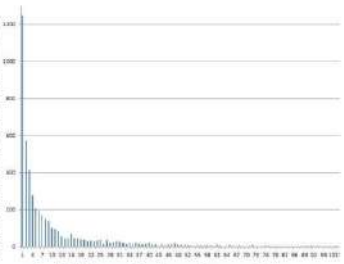


Figure 5: The frequency with which objects of the Person class occur in files.

Evaluation of training results was carried out in 2 ways: using the obtained metrics (Figures 6-8) and manually (saving data from the video using the newly created model, manually searching for erroneous results).

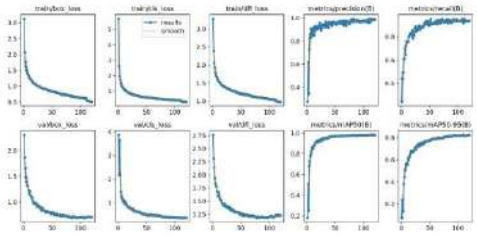


Figure 6: Metrics for training results evaluation.

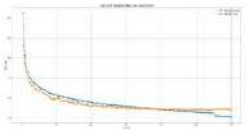


Figure 7: Train and Validation loss over epochs.

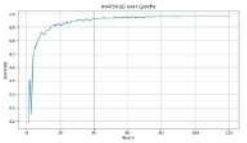


Figure 8: Mean average precision calculated at an intersection over union (IoU) threshold of 0.50.

Manual verification showed a decrease in the proportion of objects identified by the newly created model with errors to 10.7%.

As for the obtained metrics, they indicate, on the one hand, the success of training: there is an improvement in the values of the parameters Train Box Loss (up to 0.5135), mAP50 (up to 0.98367) with an increase in epochs. On the other hand, the relatively stable value of Validation Box Loss (0.69291) starting from epoch 84 suggests natural performance fluctuations due to too small a validation sample (6% of the training sample).

Further efforts will be aimed at increasing the number of objects for training to the recommended value of 10,000, the validation sample to 10-20% (1000-2000 images), the number of epochs - 300. Authors also believe that the dataset should include images from a larger number of cameras.

6. Conclusions

Therefore, in the course of work on the problem of increasing the security of the site equipped with external surveillance cameras, it was decided to develop a model for post-processing of alarm signals using a neural network. For this, an analysis of scientific publications over the past 5 years was conducted and the most effective pre-trained models that provide the highest recognition results of various objects were investigated. An experiment was also conducted by testing three models on a custom dataset of 12 images, during which it was determined that the YOLOv8 neural network model currently gives the best result. For the security of the site, it was determined that it is most important to recognize a person in the images from the cameras. Therefore, the YOLOv8 neural network was pre-trained on the COCO dataset. The results of experiments on the recognition of people in the video stream made it possible to achieve higher efficiency.

Also, manual verification revealed a notable decrease in identification errors to 10.7% with the newly developed model. Training metrics, including Train Box Loss and mAP50, demonstrated improvement, while Validation Box Loss remained relatively stable.

Future efforts will focus on augmenting the training dataset to 10,000 objects, expanding the validation sample to 10-20%, increasing epochs to 300, and incorporating images from a broader range of cameras to enhance dataset diversity.

References

[1] O. Pavlova, V. Kovalenko, T. Hovorushchenko, Neural network-based image recognition method for smart parking. Comput. Syst. Inf. Technol.). 1 (2021) 49–55.

[2] Y. Zeng, J. Zhang, A machine learning model for detecting invasive ductal carcinoma with Google Cloud AutoML Vision. Computers in biology and medicine.). 122 (2020)

[3] Y. Lu, Y. Jia, J. Wang, B. Li, W. Chai, L. Carin, Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. CVF conference on Computer Vision and Pattern Recognition. 2020. pp. 940-949.

[4] P. Radiuk, O. Pavlova, N. Hrypynska, An Ensemble Machine Learning Approach for Twitter Sentiment Analysis. Proceedings of the 3d International Workshop, volume 3171, 2022, pp. 387–397.

[5] J. P. Sahoo, A. J. Prakash, P. Pławiak, S. Samantray, Real-time hand gesture recognition using fine-tuned convolutional neural network. Sensors.). 22 (2020).

[6] M. Gazda, M. Hireš, P. Drotár, Multiple-fine-tuned convolutional neural networks for Parkinson's disease diagnosis from offline handwriting. IEEE Transactions on Systems, Man, and Cybernetics. Systems.). 52(1), (2020) 78-89.

[7] P. Radiuk, O. Pavlova, H. El Bouhissi, V. Avsiyevych, V. Kovalenko, Convolutional Neural Network for Parking Slots Detection. CEUR Workshop Proceedings, 3156, 2022, pp. 284–293.

[8] S. Bussa, A. Mani, S. Bharuka, S. Kaushik, Smart attendance system using OpenCV based on facial recognition. Int. J. Eng. Res. Technol, 9 (2021), 54-59.

[9] A. P. Ismail, F. A. Abd Aziz, N. M. Kasim, K. Daud, Hand gesture recognition on python and OPENCV. IOP Conference Series: Materials Science and Engineering, Vol. 1045, No. 1, IOP Publishing, 2020, p. 012043.

[10] M. Kushal, M. Pappa, ID Card Detection with Facial Recognition using Tensorflow and OpenCV. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). 2020. pp. 742-746.

[11] A. Sharma, K. Shah, S. Verma, Face recognition using Haar cascade and local binary pattern histogram in OpenCV. In 2021 Sixth International Conference on Image Information Processing (ICIIP), Vol. 6, pp. 298-303.

[12] Tu-Liang Lin, Hong-Yi CHANG, Kai-Hong CHEN, The pest and disease identification in the growth of sweet peppers using faster R-CNN and mask R-CNN. Journal of Internet Technology, 21(2020) 605-614.

[13] T. Mahmood, M. Arsalan, M. Owais, M. Lee, K. Park, Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster R-CNN and deep CNNs. Journal of clinical medicine, 9 (2020) 749-759.

[14] N. Palanivel, Automatic number plate detection in vehicles using faster R-CNN. 2020 International conference on system, computation, automation and networking (ICSCAN), 2020. p. 1-6.

[15] S. WAN, S. GOUDOS, Faster R-CNN for multi-class fruit detection using a robotic vision system. Computer Networks.), 168 (2020)

[16] D. Al-Obidi, S. Kacmaz. Facial Features Recognition Based on Their Shape and Color Using YOLOv8. 7th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2023.

[17] M. Safran, A. Alajmi, S. Alfarhood, Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems. Journal of Sensors, 2024.

[18] D. Xiao, Y. Nguyen, W. Yan, Fruit ripeness identification using YOLOv8 model. Multimedia Tools and Applications.). 2023, 1-18.

[19] A. Chabi, E. Mahama, A. Gouton, P. Tosse, Automatic localization of five relevant Dermoscopic structures based on YOLOv8 for diagnosis improvement. Journal of Imaging, 9 (2023) 148-159.

[20] O. Melnychenko, O. Savenko, P. Radiuk, Apple Detection with Occlusions Using Modified YOLOv5-v1, 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 2023, pp. 107-112, doi: 10.1109/IDAACS58523.2023.10348779.

[21] O. Melnychenko, O. Savenko, A Self-Organized Automated System to Control Unmanned Aerial Vehicles for Object Detection. The 4th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2023): CEUR-Workshop Proceedings, volume 3373, 2023, pp.589-600

[22] Vision AI. Google Cloud. URL: https://cloud.google.com/vision

[23] PyTorch Faster R-CNN. URL: https://pytorch.org/vision/stable/models/faster_rcnn.html

[24] COCO. Common Objects in Context. URL: https://cocodataset.org

[25] Tips for Best Training Result URL: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/

[26] Model Training with Ultralytics YOLO. URL:https://docs.ultralytics.com/modes/train/

[27] Train YOLOv8 on Custom Dataset – A Complete Tutorial. URL:https://learnopencv.com/train-yolov8-on-custom-dataset/

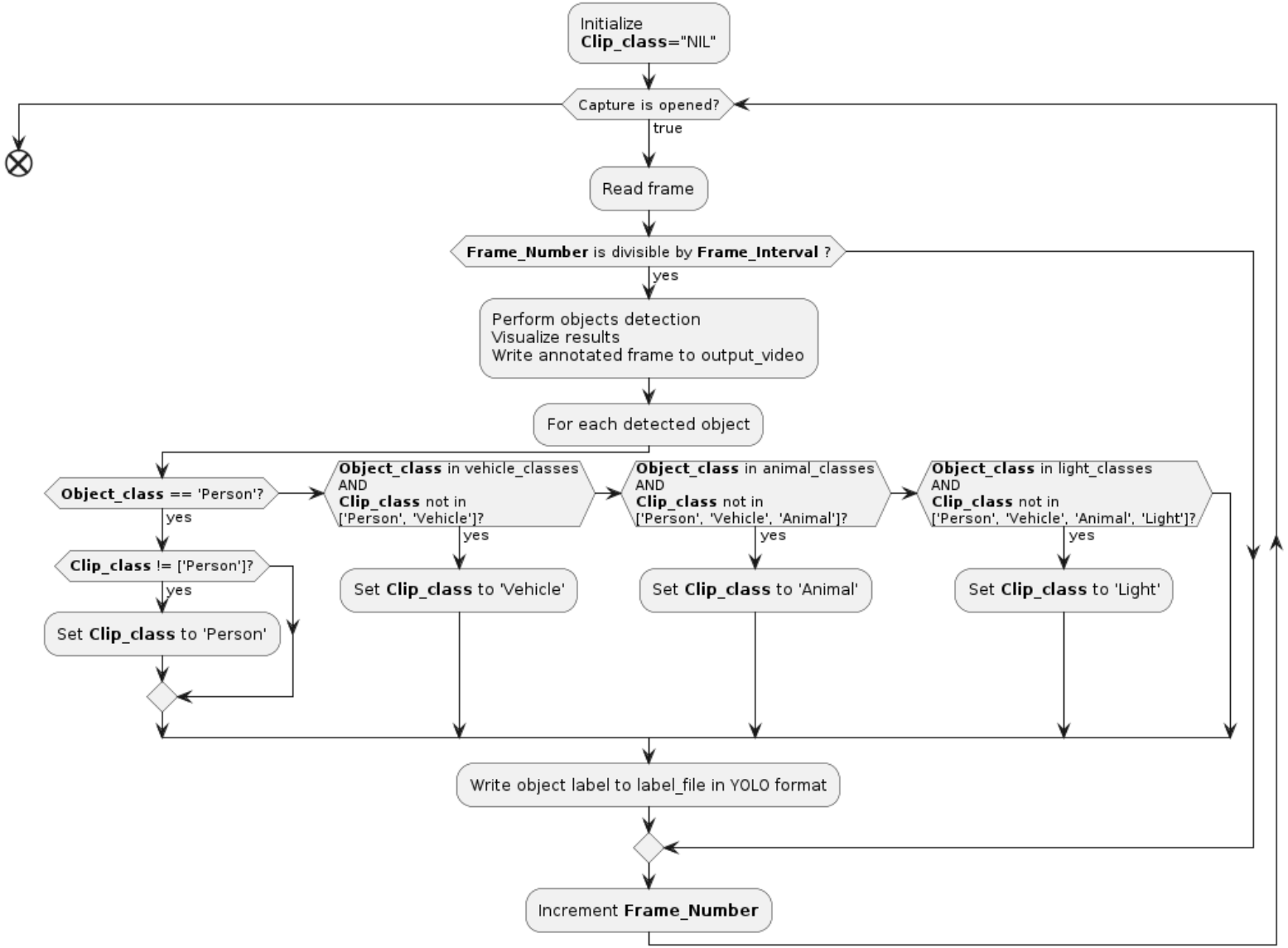
[28] Adding background images to a classification dataset. URL: https://github.com/ultralytics/ultralytics/issues/4479 (last accessed February 1, 2024)

ДОДАТОК В

СХЕМА КЛАСИФІКАЦІЇ ПОХИБОК КОМП'ЮТЕРНОГО ЗОРУ У КФС

ДОДАТОК Г

РОБОТА АЛГОРИТМУ РОЗПІЗНАВАННЯ В ЧАСТИНІ ВИЗНАЧЕННЯ КЛАСУ КЛІПІВ



ДОДАТОК Д

Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею

Кваліфікаційна робота магістра за спеціальністю
123 –Комп'ютерна інженерія

Виконав: студент 2 курсу Рудик І.В.
Керівник: д. ф., доцент Павлова О.О.

Хмельницький - 2024

Вступ

Досліджувана система відеоспостереження:

- містить понад 1200 камер спостереження;
- майже 3000 сигналів тривоги з понад 400 камер щодня;
- близько 90000 тривожних сигналів на місяць;
- більше 51% помилкових сигналів тривоги (понад 44000 щомісяця)

Фактори, що викликають помилкові спрацювання сигналів тривоги з камер спостереження:

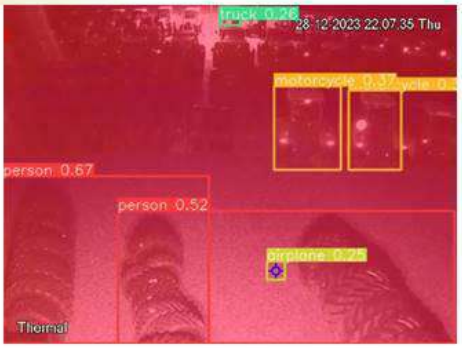
- опади (дощ, сніг);
- вітер (розгойдує предмети в полі зору камери, або саму камеру);
- світлові ефекти (мерехтіння світла на території або за її межами, фари автомобіля вночі, відблиски сонця або просто тінь від хмари вдень)

Є різноманітні:

- виробники камер;
- типи сенсора камери (ч/б, кольоровий, термо);
- розміри зображення (від 640 до 1920 по ширині з різним співвідношенням сторін);
- підсистеми сигналізації (від різних виробників і прокату);
- умови експлуатації (для внутрішнього / зовнішнього використання);
- типи носіїв (відео чи зображення); …

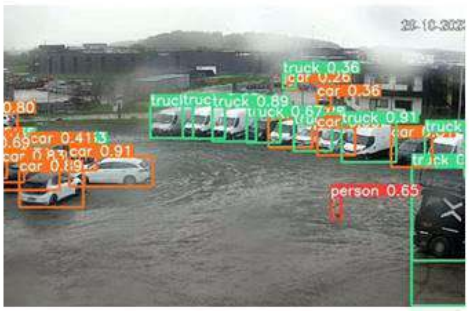
Актуальність роботи

(a)



(b)

(c)



(d)

Мета, предмет і об'єкт дослідження

Метою є забезпечення механізмів обробки похибок у відеозображеннях за допомогою нейронної мережі у кіберфізичній системі комп'ютерного зору

Об'єктом дослідження є підвищення якості роботи кіберфізичної системи комп'ютерного зору за рахунок обробки похибок у відеозображеннях.

Предметом дослідження є застосування нейронної мережі для обробки похибок у відеозображеннях.

Задачі дослідження

- розробка методів та алгоритмів обробки похибок у відеозображеннях;
- підготовка навчального датасету використовуючи відеозображення з цільових відеокамер;
- тренування моделі нейронної мережі;
- розробка програмно-технічного засобу для обробки похибок.

Наукова новизна

Наукова новизна отриманих результатів полягає у вдосконаленні існуючих методів та алгоритмів обробки нейронною мережею похибок у отриманих відеозображеннях.

Практична цінність отриманих результатів полягає в розробці програмно-технічного засобу для обробки похибок детекції об'єктів у отриманих відеозображеннях.

Апробація результатів

За темою дипломної роботи взято участь:

- у двох Всеукраїнських конференціях (АПКН-2023 у м. Хмельницький в листопаді 2023 року та IT&I у м. Миколаїв у лютому 2024 року) та опубліковано тези за матеріалами цих конференцій;
- у міжнародній конференції IntelITSIS-2024 та опубліковано статтю, яка індексується у наукометричній базі Скопус.

А також подано до друку:

- Тези для участі у Всеукраїнській конференції "Ольвійський форум" (22-24 червня м. Миколаїв).

Розділ 1

Технології та сфери застосування комп'ютерного зору в кіберфізичних системах

- Глибоке навчання та нейронні мережі
- Обробка зображень та відео в реальному часі
- 3D-зір
- Доповнена реальність (AR) та віртуальна реальність (VR)
- Периферійні обчислення (Edge Computing)

Сфери застосування

- Автономні транспортні засоби
- Виробництво та робототехніка
- Охорона здоров'я
- Сільське господарство
- Розумні міста
- Торгівля
- Моніторинг навколишнього середовища

Розділ 2

Порівняльна таблиця існуючих рішень

Criterion	Google Cloud Vision API	YOLOv8	PyTorch Faster R-CNN
Cases of count people matched	1	8	8
Found more of people than were presented really	2	1	2
Found less of people than were presented really	9	3	2
% of matched	8,3	66,7	66,7
Advantages	Good integration with other Google services	User-friendly API, high accuracy	High accuracy
Disadvantages	Low accuracy	-	Complicated API (compared to others)

Таблиця класів об'єктів за рівнем цікавості щодо розпізнавання

Пріоритет	Клас замовника	Класи моделі
1	Person	'person'
2	Vehicle	'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat'
3	Animal	'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe'
4	Light	'traffic light'
5	NIL	жоден з класів наведених вище не був виявлений

Розділ 2

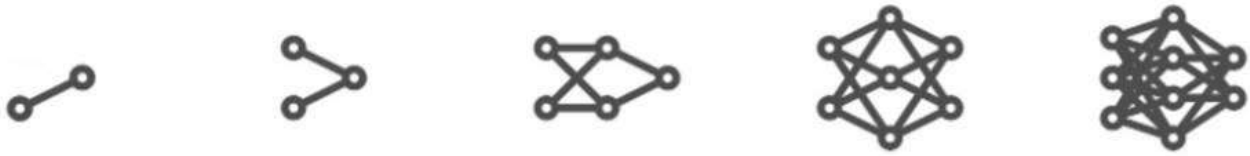
Оцінка датасету - баланс даних, репрезентативність датасету, якість розмітки.

- Кількість зображень на клас: >=1500;
- Кількість примірників (об'єктів з мітками) на клас: >=10000;
- Фонові зображення (Background images): 0-10% від загальної кількості.

  Фонові зображення – це зображення без об'єктів, які додаються до набору даних для зменшення помилкових спрацьовувань (FP). Наприклад, широко розповсюджений датасет COCO містить 1% фонових зображень.

  Крім кількісно виражених рекомендацій, існують і інші.
- Різноманітність зображень. Для випадків реального використання рекомендується використовувати зображення отримані в різний час доби, різні пори року, різну погоду, різне освітлення, різні ракурси, з різні джерел (камер) тощо.
- Послідовність маркування. Усі екземпляри всіх класів на всіх зображеннях мають бути позначені. Часткове маркування не працюватиме.
- Точність маркування. Мітки повинні щільно охоплювати кожен об'єкт. Між об'єктом і його обмежувальною рамкою не повинно бути проміжків.

Розділ 2



Nano	Small	Medium	Large	XLarge
YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
6.5 MB 0.99 ms_{A100} 37.3 mAP_{COCO}	22.6 MB 1.2 ms_{A100} 44.9 mAP_{COCO}	52.1 MB 1.83 ms_{A100} 50.2 mAP_{COCO}	87.8 MB 2.39 ms_{A100} 52.9 mAP_{COCO}	136.9 MB 3.53 ms_{A100} 53.9 mAP_{COCO}

Розділ 3

N = round (fps(source)/fps(target)), (3.1)

де fps(source) - fps вхідного відео;

fps(target) - fps вхідного відео.

Метою алгоритму YOLO є передбачення класу об'єкта та обмежувальної рамки (bounding box), яка визначає розташування об'єкта на вхідному зображенні. Алогритм розпізнає кожну обмежувальну рамку використовуючи чотири числа:

- центр обмежувальної рамки (b_x, b_y);
- ширина рамки (b_w);
- висота рамки (b_h).

Крім цього, YOLO передбачає відповідне значення C для прогнозованого класу, а також ймовірність цього прогнозу (P_c).

Для кожної клітинки сітки ми можемо закодувати вектор, який описуватиме її. Наприклад, перша клітинка зліва вгорі не містить жодного об'єкта, і ми описуємо її як:

$$C_{1,1} = (P_c, B_x, B_y, B_w, B_h, C_1, C_2) = (0, ?, ?, ?, ?, ?, ?), \quad (3.2)$$

де (P_c)– ймовірність класу об'єкта;

B_x, B_y – координати центру обмежувальної рамки відносно комірки;

B_w, B_h – ширина та висота обмежувальної рамки відносно всього зображення;

C_1, C_2 - клас який представляє обмежувальна рамка (C_1 для першого об'єкта і C_2 для другого відповідно).

Розділ 3

Похибки алгоритмів комп'ютерного зору. Характеризуються такими параметрами, як Точність (Precision), Акуратність (Accuracy), Відгук (Recall), F1-оцінка (F1 score).

Точність (Precision). Показує, скільки з позитивних прогнозів виявилося правильними.

$$Precision = TP / (TP + FP), \quad (3.3)$$

де TP - кількість правильно класифікованих позитивних прикладів;

TN - к ількість правильно класифікованих негативних прикладів.

Акуратність (Accuracy). Вимірює, скільки з усіх прогнозів, зроблених моделлю, є правильними.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN), \quad (3.4)$$

де TP - кількість правильно класифікованих позитивних прикладів;

TN - к ількість правильно класифікованих негативних прикладів;

FP -кількість помилково класифікованих позитивних прикладів (хибні спрацьовування);
FN - к ількість помилково класифікованих негативних прикладів (пропуски).

Відгук (Recall). Це частка позитивних випадків, які модель правильно визначила.

$$Recall = TP / (TP + FN) \quad , \tag{3.5}$$

де TP - кількість правильно класифікованих позитивних прикладів;

TN - к ількість правильно класифікованих негативних прикладів.

F1-оцінка (F1 score). Є метрикою для вимірювання продуктивності моделі у задачах класифікації. Вона поєднує точність і відгук в один показник, щоб забезпечити збалансовану оцінку точності моделі. Особливо актуальним є використання цього показника в наборах даних, розбалансованих за класами.

$$F1 = 2 * (Precision * Recall) / (Precision + Recall) \, , \tag{3.6}$$

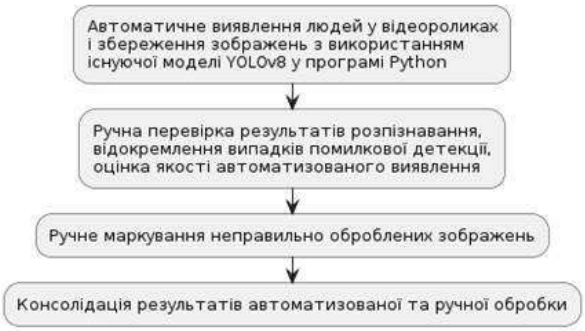
де Recall – відгук (1.7);
Precision – точність (1.5).

Цей же показник F1 може бути розрахований без необхідності розрахунку Precision та Recall.

$$F1=TP / (TP + (FP + FN) / 2) \tag{3.7}$$

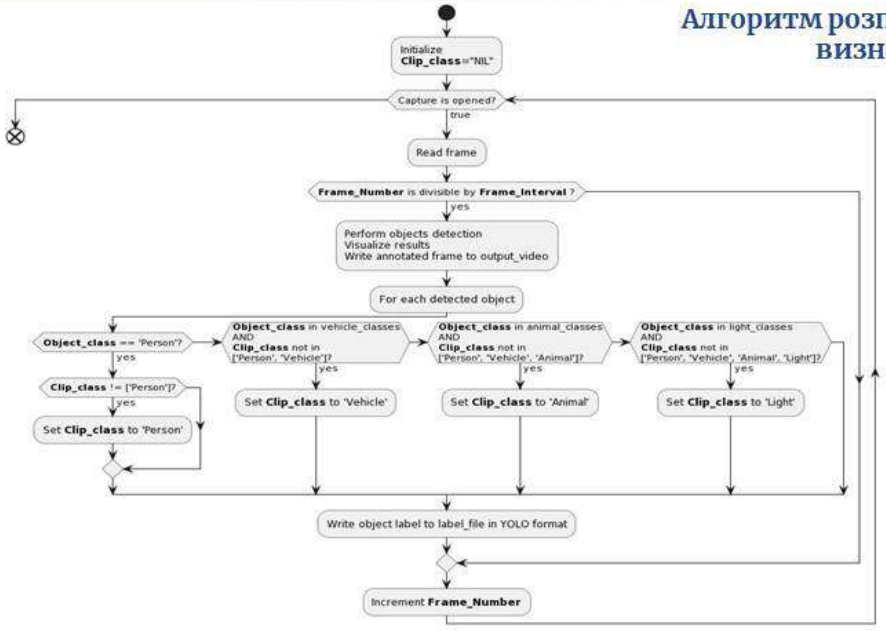
Розділ 3

Метод обробки похибок розпізнавання об'єктів за допомогою камер зовнішнього спостереження у кіберфізичних системах комп'ютерного зору



Алгоритм навчання нейронної мережі для обробки похибок

Розділ 3



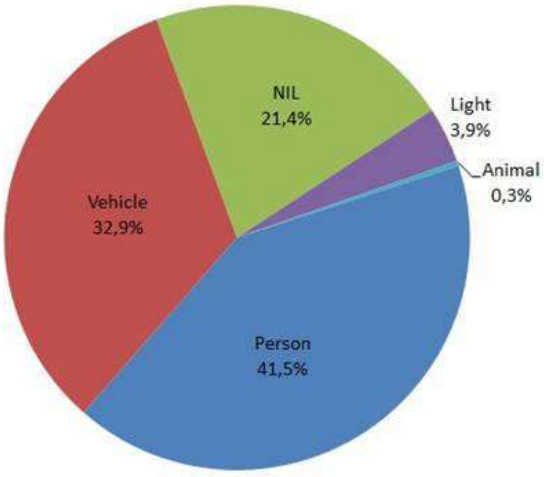
Алгоритм розпізнавання об'єктів в частині визначення класу кліпів

Розділ 4

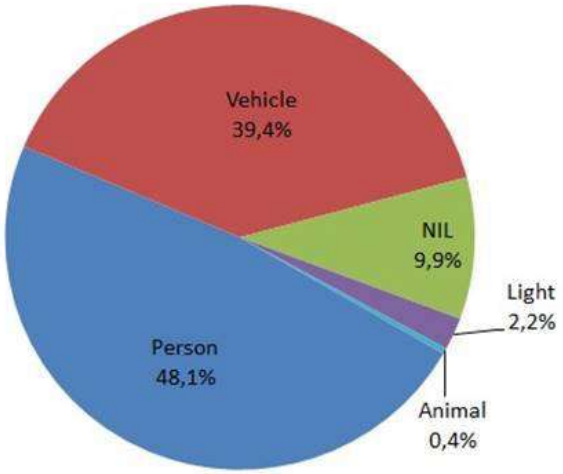
Реальний набір даних для проведення експериментів

Розділ 4

Фактичний розподіл кліпів на прикладі денної та тижневої вибірки



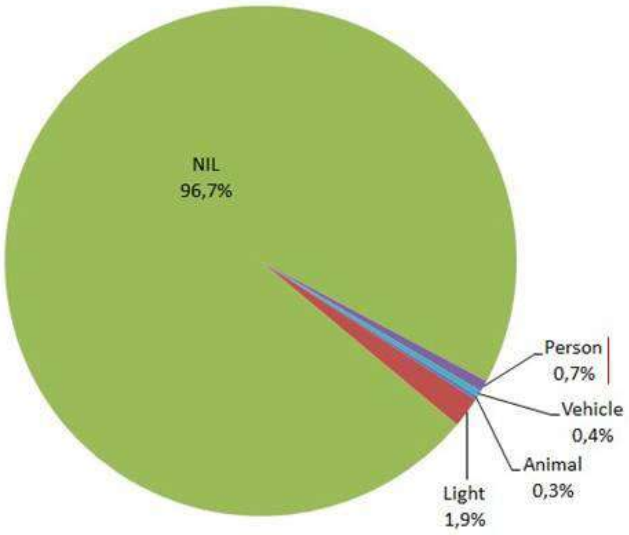
a) Денна вибірка (1440 кліпів)



b) Тижнева вибірка (11247 кліпів)

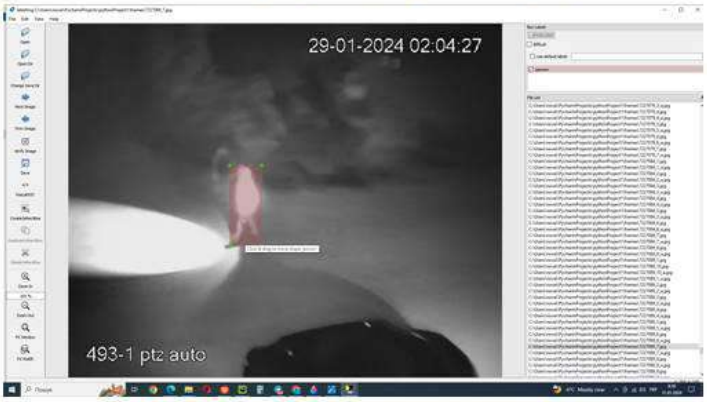
Розділ 4

Результати ручної перевірки у відео, де нейромережа знайшла людей



Розділ 4

Приклад головного вікна програми для ручного маркування Labelimg



Приклад маркування у форматі YOLO

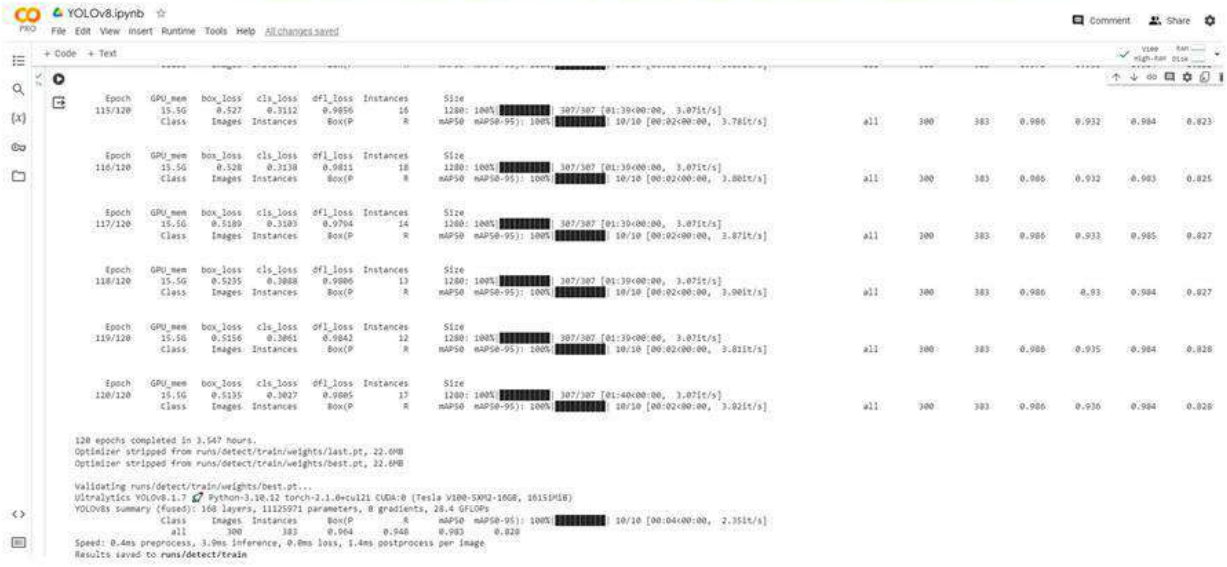
7086139_1.txt: Блокнот

Файл Редагування Формат Вигляд Довідка

```
0 0.4739784896373749 0.5528628826141357 0.06303367018699646 0.22730790078639984
0 0.8352502584457397 0.608235239982605 0.07534477859735489 0.3198038339614868
0 0.5934038758277893 0.5579432249069214 0.06809243559837341 0.306794852018563
0 0.3098285496234894 0.4880482256412506 0.06194643676280975 0.2799293100833893
0 0.6407550573348999 0.5961031317710876 0.06510868668556213 0.24754470586776733
```

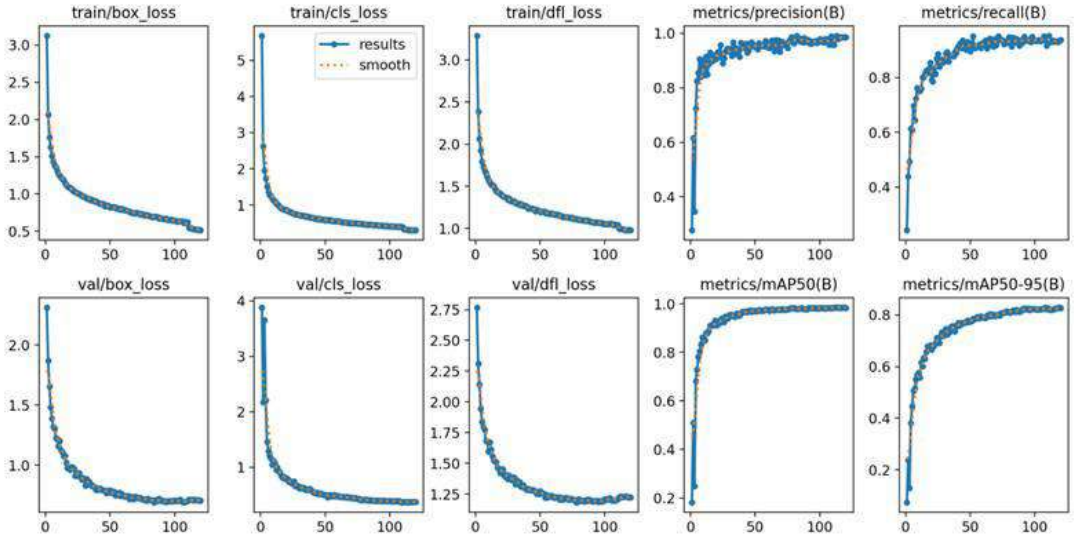
Розділ 4

Приклад робочої сторінки Google Colab під час навчання власної моделі YOLO



Розділ 4

Набір діаграм з результатами навчання моделі YOLO

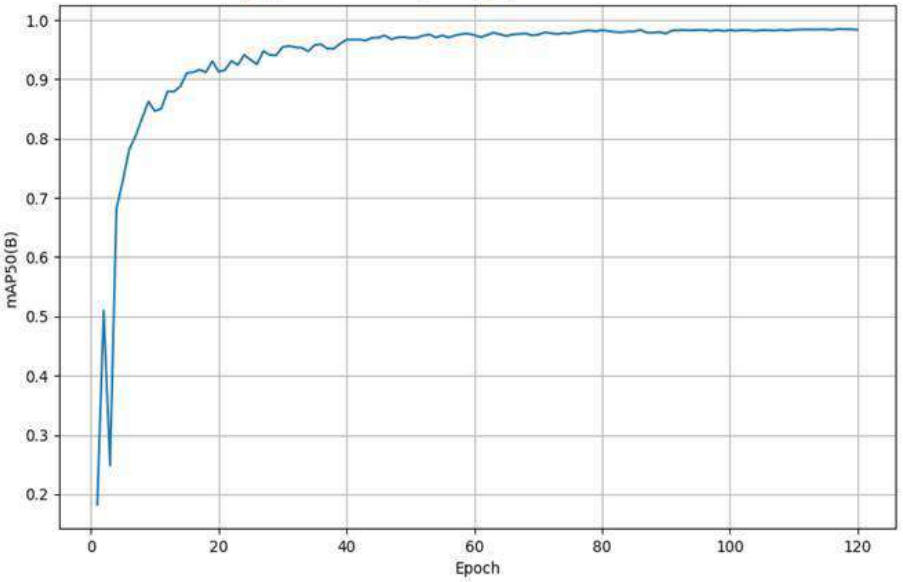


Розділ 4

Приклад файлу .csv із широким спектром параметрів щодо навчання YOLO

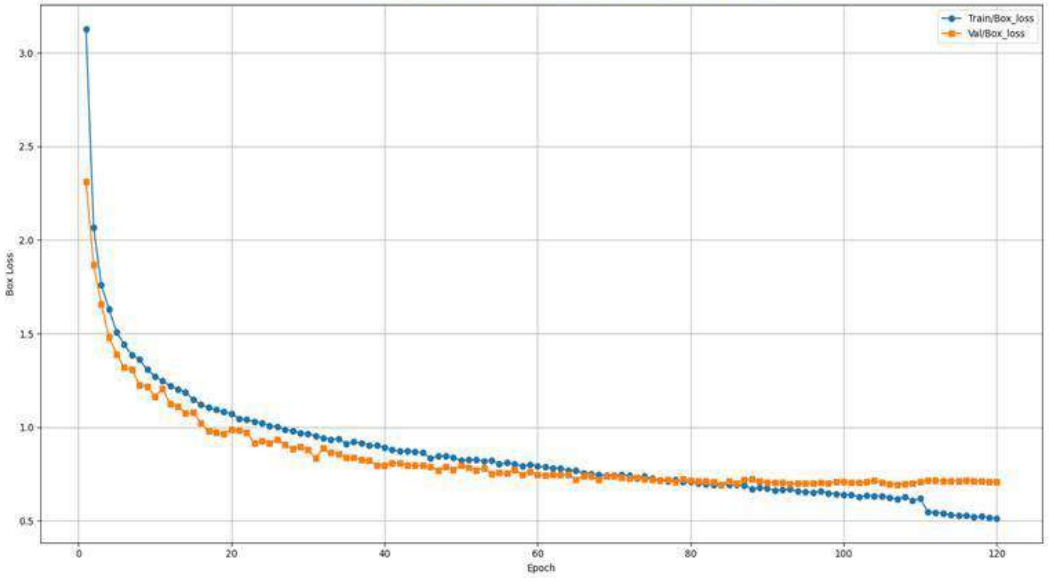
epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss
100	0.63855	0.41384	1.059	0.9781	0.93473	0.98284	0.82323	0.70666
101	0.63717	0.41908	1.054	0.97546	0.934	0.98203	0.82196	0.70425
102	0.6282	0.41191	10.512	0.97018	0.93434	0.98307	0.82104	0.7019
103	0.63299	0.40831	10.501	0.96761	0.93608	0.98258	0.8214	0.70693
104	0.63219	0.41154	10.554	0.97813	0.93409	0.98185	0.82124	0.71377
105	0.6312	0.40572	10.552	0.97813	0.93437	0.98283	0.82349	0.70218
106	0.62421	0.40146	10.507	0.97291	0.93753	0.98271	0.82406	0.69689
107	0.61364	0.39426	10.391	0.95791	0.95069	0.98218	0.82582	0.69314
108	0.62688	0.41037	10.469	0.96428	0.94778	0.98344	0.82862	0.69457
109	0.60943	0.39036	10.414	0.98874	0.91683	0.98234	0.82327	0.70108
110	0.6179	0.39998	10.406	0.95784	0.94909	0.98355	0.82722	0.70718
111	0.54635	0.32693	0.99632	0.95754	0.953	0.98389	0.82217	0.71411
112	0.54225	0.3248	0.99565	0.98553	0.9295	0.98411	0.82047	0.71433
113	0.54041	0.32098	0.99543	0.99165	0.93069	0.98389	0.82051	0.7131
114	0.53052	0.31578	0.9908	0.97817	0.93605	0.98414	0.82184	0.71046
115	0.52697	0.31117	0.98558	0.98593	0.93211	0.98417	0.82326	0.71033
116	0.52797	0.31383	0.98108	0.98585	0.93211	0.98339	0.82463	0.71336
117	0.51891	0.31029	0.97942	0.98619	0.93254	0.98474	0.82652	0.7116
118	0.5235	0.30881	0.98062	0.98623	0.9295	0.9845	0.82711	0.71063
119	0.51555	0.30607	0.98417	0.98617	0.93473	0.98439	0.82813	0.70794
120	0.5135	0.30273	0.98045	0.98625	0.93633	0.98367	0.82755	0.70797

Середня точність (mAP50) за всіма епохами навчання



Розділ 4

Train and Validation Box loss over epochs



Висновки

В ході роботи над проблемою підвищення безпеки об'єкта, обладнаного зовнішніми камерами спостереження, було вирішено розробити модель постобробки сигналів тривоги за допомогою нейронної мережі.

Також був проведений експеримент шляхом тестування трьох моделей на спеціальному наборі даних із 12 зображень, під час якого було визначено, що модель нейронної мережі YOLOv8 наразі дає найкращий результат. Для безпеки сайту було визначено, що найважливіше впізнати людину на знімках з камер.

Результати експериментів з розпізнавання людей у відеопотоці дозволили досягти більшої ефективності. Крім того, ручна перевірка виявила значне зменшення помилок ідентифікації з 19,4% до 10,7% у новій розробленій моделі.

Показники навчання, зокрема Train Box Loss і mAP50, продемонстрували покращення, тоді як Validation Box Loss залишалися відносно стабільними. Подальші зусилля будуть зосереджені на розширенні навчального набору даних до 10 000 об'єктів, розширенні перевірочної вибірки до 10-20%, збільшенні епох до 300 і включенні зображень із більш широкого діапазону камер для підвищення різноманітності набору даних.

Дякую за увагу!

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016263942

Дата перевірки:
20.05.2024 06:13:24 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
20.05.2024 06:16:36 EEST

ID користувача:
100005591

Назва документа: **Рудик_Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозо...**

Кількість сторінок: **101** Кількість слів: **16157** Кількість символів: **121793** Розмір файлу: **5.02 MB** ID файлу: **1016053539**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.15%
Схожість

Найбільша схожість: **0.89%** з джерелом з Бібліотеки (ID файлу: **1014732335**)

1.74% Джерела з Інтернету	133	Сторінка 103
1.5% Джерела з Бібліотеки	64	Сторінка 104

0.43% Цитат

Цитати	9	Сторінка 105
Посилання	1	Сторінка 105

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	11
Підозріле форматування	17 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 126558 Назва: МКР Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею Додано в БД: 2024-05-20 Автора: Рудик І.В. Керівники: Павлова О.О Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	103515	906	1969 (2%)	25 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Рудик Іван Вікторович

Тема: Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість сторінок записки 88 с.

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є забезпечення механізмів обробки похибок у відеозображеннях за допомогою нейронної мережі у кіберфізичній системі комп'ютерного зору.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проаналізовано галузь кіберфізичних систем із застосуванням методів розпізнавання зображень, проведено огляд літературних джерел та виконано порівняльний аналіз існуючих рішень із застосуванням методів розпізнавання зображень. У другому розділі побудовано математичну модель розпізнавання зображень для кіберфізичної системи комп'ютерного зору. А також досліджено функційні та нефункційні вимоги до пропонованої системи. У третьому розділі розглянуто принцип застосування технології методів розпізнавання зображень на основі нейронних мереж із застосуванням моделі YOLOv8. У четвертому розділі спроектовано архітектуру системи для обробки похибок у відеозображеннях за допомогою нейронної мережі у кіберфізичній системі комп'ютерного зору. Наукова новизна отриманих результатів полягає у вдосконаленні існуючих методів та алгоритмів роботи з технологією розпізнавання зображень та впровадженні даних методів і алгоритмів у кіберфізичній системі комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею. Вперше розроблено

метод застосування моделі нейронної мережі YOLOv8 для обробки похибок отриманих відеозображень нейронною мережею у кіберфізичній системі комп'ютерного зору.

4. Позитивні сторони роботи: розв'язання актуальної науково-прикладної задачі.

5. Негативні сторони роботи:

6.Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на високому науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно.

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Бармак Олександр Володимирович., д.т.н., професор, зав. кафедри комп'ютерних наук ХНУ

"17" травня 2024 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Рудика Івана Вікторовича

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-22-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Кіберфізична система комп'ютерного зору з обробкою похибок отриманих відеозображень нейронною мережею

Автор: Рудик Іван Вікторович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Павлова О.О., д.ф., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
2) зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості Unicheck, складає 2.15% і адресується до 197 першоджерел; найбільша схожість з джерелом сягає 0.89%. Обсяг запозичень, визначених системою Anti-Plagiarism складає 1.0%. Все це, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О.О. Павлова

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко