НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«На правах рукопису»

УДК *004.023*

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
«_____» _____ 2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою *«Інженерія програмного забезпечення комп'ютеризованих систем»*

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему *«Алгоритмічне забезпечення для розмітки надвеликих об'ємів даних для задачі детекції об'єктів методами комп'ютерного зору»*

Виконав *студент VI курсу ФІОТ, групи ІП-91мн*
Зарічковий Олександр Анатолійович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник *доц., к.т.н., викладач кафедри АСОІУ, І. П. Муха* _____
(посада, науковий ступінь, вчене звання, ініціали, прізвище) (підпис)

Рецензент *доц., к.т.н., викладач кафедри ТК, О. І. Лісовиченко* _____
(посада, науковий ступінь, вчене звання, ініціали, прізвище) (підпис)

Засвідчую підписом, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

Рівень вищої освіти *другий (магістерський)*
(повна назва)

Спеціальність *121 «Інженерія програмного забезпечення»*
(код та назва)

Освітньо-наукова програма *«Інженерія програмного забезпечення комп'ютеризованих систем»*
(повна назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
Олександр ПАВЛОВ
(підпис) (ініціали, прізвище)
«_____» _____ 2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Зарічковому Олександру Анатолійовичу
(прізвище, ім'я, по-батькові)

1. Тема дисертації *Алгоритмічне забезпечення для розмітки надвеликих об'ємів даних для задачі детекції об'єктів методами комп'ютерного зору*
науковий керівник *к.т.н., доц., викладач кафедри АСОІУ, Муха Ірина Павлівна*
(посада, науковий ступінь, вчене звання, ПІБ)
затверджені наказом по університету *№ 809-с* від «*12*» *березня* 2021 р.

2. Термін подання студентом дисертації «*06*» *травня* 2021 р.

3. Об'єкт дослідження *Процес автоматизації розмітки даних для задачі детекції об'єктів*

4. Предмет дослідження *Методи та засоби автоматизації процесу розмітки даних для задачі детекції об'єктів*

5. Перелік завдань, які необхідно розробити *Дослідити наявні методи та засоби процесу розмітки даних для задач комп'ютерного зору; дослідити наявні алгоритми постійного донавчання алгоритмів машинного навчання; удосконалити існуючі підходи до розмітки даних шляхом адаптації алгоритмів постійного; виконати експериментальні дослідження характеристик запропонованих рішень.*

6. Орієнтовний перелік графічного матеріалу *діаграма діяльності процесу розмітки даних, діаграма розгортання програмного забезпечення*

7. Орієнтовний перелік наукових публікацій *«Швидке навчання моделей детекції об'єктів на великих наборах даних», «Вплив підходів до агрегація експертних думок на якість детекції раку грудної клітки»*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис та дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «____» _____ 2020 р.

Календарний план

№ з/п	Назва етапу виконання магістерської дисертації	Термін виконання	Примітка
1	*Ознайомлення з предметною областю, вивчення літератури*	*29.11.2019*	
2	*Аналіз наявних методів та засобів автоматизації розмітки даних*	*28.02.2020*	
3	*Постановка та формалізація задачі дослідження*	*13.03.2020*	
4	*Аналіз вимог до програмного забезпечення*	*27.03.2020*	
5	*Розробка методів та засобів для розв'язання поставлених задач*	*29.05.2020*	
6	*Розробка архітектури програмного забезпечення*	*23.10.2020*	
7	*Виконання експериментальних досліджень*	*18.12.2020*	
8	*Оформлення пояснювальної записки*	*09.04.2021*	
9	*Подання дисертації на попередній захист*	*16.04.2021*	
10	*Подання дисертації на рецензію*	*06.05.2021*	
11	*Подання дисертації на захист*	*13.05.2021*	

Студент _____ *Олександр ЗАРІЧКОВИЙ*
(підпис) (прізвище та ініціали)

Науковий керівник _____ *Ірина МУХА*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація: 119 с., 33 рис., 6 табл., 2 додатки, 91 джерело.

Актуальність теми. Штучний інтелект все більше проникає у життя пересічних людей, автоматизуючи такі звичні сфери життя, як керування автомобілем, покупки в магазинах, діагностику захворювань та багато іншого. Для пришвидшення адаптації алгоритмів штучного інтелекту в програмне забезпечення були створені програмні платформи штучного інтелекту, такі як Microsoft Azure AI Platform, Amazon AWS SageMaker, Google Cloud AI Platform, IBM Watson, які спрощують процес створення та використання алгоритмів штучного інтелекту при вирішенні різноманітних прикладних задач. Головним недоліком даних платформ являється те, що вони концентруються лише на процесі створення та використання алгоритмів штучного інтелекту, хоча дослідження в даній сфері показують, що 80% часу на таких проектах витрачається на роботу з даними, зокрема на їх розмітку. В існуючих платформах підтримки задач штучного інтелекту автоматизація процесу розмітки даних або відсутня, або наявна лише для типових задач, що призводить до неефективного (повільного) процесу розмітки даних, а отже збільшує вартість розробки програмного забезпечення з використанням алгоритмів штучного інтелекту. Саме тому актуальною є задача розробки програмного забезпечення розмітки даних, що дозволить пришвидшити розмітку надвеликих об’ємів даних.

Мета досліджень. Підвищити ефективність, зокрема швидкодію, розмітки даних для задач детекції об’єктів шляхом авторозмітки даних за допомогою методів комп’ютерного зору, які виконують чорнову розмітку даних, на якій людині потрібно лише виправити помилки, допущені алгоритмом в процесі авторозмітки. Щоб досягнути поставленої мети необхідно організувати процес постійного донавчання алгоритму детекції об’єктів на нових даних та вірний порядок розмітки зображень, що дозволить пришвидшити розмітку усього наявного набору даних.

Для реалізації поставленої мети були сформовані **наступні завдання**:

– дослідити наявні підходи до автоматизації розмітки даних в програмних платформах для вирішення задач штучного інтелекту;

– дослідити наявні підходи донавчання алгоритмів машинного навчання;

– удосконалити процес авторозмітки даних для задач детекції об’єктів з метою пришвидшення розмітки надвеликих наборів даних;

– розробити програмне забезпечення авторозмітки даних для задач детекції об’єктів;

– виконати експериментальне дослідження характеристик розробленого програмного забезпечення.

Об’єкт досліджень. Процес авторозмітки даних для задач комп’ютерного зору, зокрема задач детекції об’єктів.

Предмет досліджень. Підходи до авторозмітки даних для задач детекції об’єктів.

Методи досліджень. Емпіричні дослідження.

Наукова новизна. Запропоновано механізм пріоритезації процесу розмітки даних як удосконалення механізму само-діагностики моделей детекції об’єктів, який встановлює порядок розмітки даних шляхом першочергового відбору для навчання алгоритму детекції об’єктів найскладніших зображень, вибір яких здійснюється на основі агрегатів ознак, згенерованих глибокою згортковою нейромережею для оцінки складності зображень, що пришвидшує процес навчання в порівнянні з використанням випадкових зображень з того ж набору даних.

Удосконалено алгоритм авторозмітки надвеликих об’ємів даних для задач детекції об'єктів за рахунок використання запропонованого механізму пріоритезації послідовності розмітки даних та адаптації підходів постійного донавчання алгоритмів детекції, що дозволяє збільшити швидкість розмітки даних та використати даний алгоритм для реалізації розмітки надвеликих даних для задач детекції як типових, так і нетипових об’єктів.

Практичне значення. Розроблено програмне забезпечення авторозмітки даних в рамках програмної платформи підтримки вирішення задач штучного інтелекту для задач детекції об’єктів, що вирізняється від існуючих аналогів підвищеною швидкістю процесу розмітки на надвеликих об’ємах.

Апробація. Основні положення роботи доповідались і обговорювались на IV Міжнародній конференція з комп’ютерних наук, інжинірингу та освітніх технологій (ICCSEEA 2021), а також на VI Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2021).

Ключові слова: ПРОГРАМНІ ПЛАТФОРМИ ШІ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ДЕТЕЦІЯ ОБ’ЄКТІВ, АВТОРОЗМІТКА ДАНИХ, НАДВЕЛИКІ ДАНІ

ABSTRACT

Topic: «Algorithmic software for big data annotation for object detection task using computer vision methods»

Master's degree thesis: 119 pages, 33 figures, 6 tables, 2 attachments, 91 references.

Relevance of the topic. AI algorithms are increasingly penetrating the lives of ordinary people, automating such common areas of life as driving a car, shopping, disease diagnosis, and much more. To simplify the development process of artificial intelligence algorithms, artificial intelligence software platforms such as Microsoft Azure AI Platform, Amazon AWS SageMaker, Google Cloud AI Platform, IBM Watson, and many others have been created. The major drawback of these platforms is that they focus on simplifying the process of creating artificial intelligence algorithms, but according to Cognilytica analysts, only 20% of the time on artificial intelligence projects is spent working with the algorithm, and the other 80% - working with data. The longest stage in working with data is data markup for further use in the learning process of machine learning algorithms, and in modern artificial intelligence platforms there is no automation of the markup process, or it available only for typical tasks, which leads to inefficient (slow) data markup process. That is why the task of developing data markup software, which will increase the speed of data markup, is relevant.

Research objective. Improve efficiency of data markup process for object detection tasks by auto-labeling data using computer vision techniques that perform pre-markup of data on which a person only needs to correct errors of an algorithm. To achieve this goal, it is necessary to organize the process of continuous learning of the algorithm for detecting objects on new data and the correct order of image markup, which will allow as soon as possible to mark the entire available data set.

The following objectives have been formulated to achieve this research objective:

– explore existing approaches to automating data markup in software platforms to solve AI problems;

– explore existing approaches for incremental learning of machine learning algorithms;

– improve the auto-labeling process for object detection tasks to speed up the markup of large data sets;

– develop data markup software for object detection tasks;

– study the effectiveness of the algorithm.

The object of research. The process of auto-markup data for computer vision tasks, including object detection tasks.

The subject of research. Approaches for auto-labeling of data for object detection tasks.

The scientific innovation. Proposed mechanism of prioritization of data markup process as improvement of the mechanism of self-diagnostics object detection models which establishes the labeling order by selecting the most difficult images based on predictions of neural network. Training on such images is much faster compared to using random images from the same data set.

The algorithm for auto-labeling of large volumes of data for object detection tasks has been improved by utilizing proposed mechanism of prioritization and by adaptating approaches for continuous learning, which allows to increase data labeling speed and can be used for labeling both typical and atypical objects.

The practical value. Developed auto-labeling software within the AI software platform for object detection problems, which differs from existing analogues by the increased speed of the markup process at very large volumes.

Approbation. The results of the master's degree thesis were reported in The Fourth International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2021) and a collection of materials from the student conference held in Kiev, Ukraine in 2021.

Keywords: AI SOFWARE PLATFORMS, COMPUTER VISION, NEURAL NETWORKS, OBJECT DETECTION, AUTOLABELING, BIG DATA

ЗМІСТ

ВСТУП

Штучний інтелект являється одним з найбільш швидкозростаючих ринків корпоративного програмного забезпечення. За даними галузевих аналітиків компанії McKinsey, в 2025 організації проінвестують більше ніж $250 млрд. в розробку програмного забезпечення з використання технологій штучного інтелекту [1] [2]. За рахунок цих інвестицій компанії розраховують отримати додатковий дохід на суму в $13 трильйонів. Це створює великий попит на розробку прикладного програмного забезпечення з використанням штучного інтелекту. Для того, щоб задовольнити цей попит, технологічні гіганти створили програмні платформи для підтримки задач ШІ такі як Microsoft Azure AI Platform [3], Google Cloud AI Platform [4], Amazon AWS SageMaker [5], IBM Watson [6] та безліч інших, які допомагають в створенні та використанні алгоритмів штучного інтелекту, що дозволяє розробляти програмні продукти з використанням алгоритмів штучного інтелекту значно швидше, в порівнянні з класичними підходами. Головним недоліком існуючих платформ являється те, що вони концентруються лише на процесі створення та використання алгоритмів штучного інтелекту, хоча дослідження в сфері показують, що 80% часу на таких проектах витрачається на роботу з даними [7] для їх подальшого використання в процесі навчання алгоритмів на основі популярних бібліотек Tensorflow [8], PyTorch [9], Scikit-learn [10], тощо.

Найдовшим етапом в роботі з даними являється їх розмітка для подальшого використання в процесі навчання алгоритмів машинного навчання. Деякі з існуючих платформ для підтримки задач штучного інтелекту пропонують вбудовані сервіси для розмітки даних, які виконуються класичним підходами без використання жодних методів автоматизації або з обмеженою підтримкою автоматизації для типових задач, що призводить до неефективного (повільного) процесу розмітки даних. Особливо гостро ця проблема постає для задач, які потребують надвеликих об’ємів даних для досягнення задовільного рівня якості – водіння автомобіля (автопілоти), верифікація пасажирів в

аеропортах (системи розпізнавання людей), покупок в магазині (магазини без кас), діагностики раку грудної клітини та багатьох інших, включаючи задачі детекції об’єктів на яких базуються всі вищезгадані задачі.

Метою даної магістерської роботи є підвищення ефективності, зокрема швидкодії, розмітки даних для задач детекції об’єктів шляхом авторозмітки даних за допомогою методів комп’ютерного зору, які виконують чорнову розмітку даних, на якій людині потрібно лише виправити помилки алгоритму. Щоб досягнути поставленої мети, необхідно організувати процес постійного донавчання алгоритму детекції об’єктів на нових даних та вірний порядок розмітки зображень, що пришвидшить процес розмітки даних.

Актуальністю цих досліджень, а також наукової діяльності у наведеній предметній галузі є те, що попри інтенсивний розвиток платформ розробки та підтримки задач штучного інтелекту, зокрема комп’ютерного зору та детекції об’єктів, засоби автоматизації розмітки даних рідко використовуються для пришвидшення процесу розмітки даних, що виливається у велику вартість збору даних для багатьох бізнес-задач, з яких лише 43% можуть зібрати достатній об’єм даних для того, щоб успішно завершити розробку програмного забезпечення з використанням алгоритмів штучного інтелекту [7]. Автоматизація процесу розмітки даних дозволить багатьом бізнесам зменшити вартість одиниці даних для задач штучного інтелекту, а отже дасть можливість зробити якісний стрибок в автоматизації вирішення прикладних задач.

Об’єктом досліджень даної магістерської роботи є процес авторозмітки даних для задач комп’ютерного зору, зокрема задач детекції об’єктів. В роботі акцент робиться на підходах для пришвидшення процесу розмітки даних для задач детекції об’єктів.

Для досягнення мети магістерської дисертації необхідно виконати наступні завдання:

– дослідити наявні підходи до автоматизації розмітки даних в програмних платформах для вирішення задач штучного інтелекту;

– дослідити наявні підходи донавчання алгоритмів машинного навчання;

– удосконалити процес авторозмітки даних для задач детекції об’єктів з метою пришвидшення розмітки великих наборів даних;

– розробити програмне забезпечення авторозмітки даних для задач детекції об’єктів;

– виконати експериментальне дослідження характеристик розробленого програмного забезпечення.

1 ОГЛЯД ПЛАТФОРМ ПІДТРИМКИ ЗАДАЧ ВИРІШЕННЯ ЗАДАЧ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Загальні положення

В останнє десятиріччя відбувся великий прорив в областях машинного навчання, штучних нейронних мереж, і, як наслідок, у комп'ютерному зорі. Можливо найкращим прикладом цього буде проект "Asirra" від Microsoft Research [11]. Проект був заснований у 2006 році як засіб для відрізнення ботів від людей в мережі Інтернет. За його задумом користувачу ставилась задача відповісти - кіт чи собака знаходиться на фотографії. Зображення брались з постійно зростаючої бази, що збиралася в притулках для тварин. Ця система безпеки вважалася майже ідеальною, бо задача відрізнення кота від собаки вважалася ШІ-повною, тобто для її вирішення необхідно створення повноцінного штучного інтелекту. У той час найкращі алгоритми мали точність близько 60%, тоді як випадкове вгадування на даній задачі має точність 50%. Тому ймовірність, що алгоритм зможе вірно розпізнати 15 зображень була мізерною (близько 0.047%), в той час як для людини ця задача не створить жодних проблем. Результатом проекту стало його закриття у 2014 році з причини вирішення поставленої задачі. Найкращі алгоритми на основі глибоких штучних нейронних мереж мали якість більше 99%, і, як наслідок, 15 зображень розпізнавалися вірно з ймовірністю 86%, а 50 – з ймовірністю 60.5% [12]. Кількість помилок у цій надзвичайно складній для комп'ютера задачі зменшилась у 40 разів та стала абсолютно мізерною. І все це лише за 8 років.

Звичайно, задача відрізнення котів від собак є далеко не найважливішим здобутком машинного навчання та комп'ютерного зору. Є безліч куди важливіших та корисніших для людства задач: детекція об'єктів на зображеннях, розпізнавання лиця, розпізнавання пішоходів і дорожніх знаків та багато інших. Але ця задача є дуже важливою з теоретичної точки зору – вона показала фундаментальні проблеми та

обмеження комп'ютерного зору, вирішення яких дуже сильно допомогло для розвитку інших напрямків.

На наш час більшість задач комп'ютерного зору, пов'язаних з розумінням та класифікацією того, що зображено на картинці, уже мають хороше рішення, а для деяких задачах комп'ютер навіть перевершив людину. Може скластися враження що усі задачі комп'ютерного зору буду повністю вирішеними у найближчі роки. Але це не так.

Щоб отримати повне розуміння того що відбувається на зображені, нам не достатньо лише розуміти, які об'єкти є на зображенні, а й – знати точне розташування усіх об'єктів, зрозуміти як вони взаємодіють один з одним та виявити інші шаблони, які допоможуть в більш глибоко зрозуміти усю сцену на зображені. Задача виявлення точного положення всіх об'єктів на сцені називається детецією об'єктів [13]. До типових задач детекції об'єктів відносять задачі детекції обличчя [14], пішоходів [15], скелету об'єктів [16] тощо. Алгоритми детекції об'єктів надають значно більше семантичної інформації про об'єкти на сцені, що активно використовується в різних застосунках, включаючи класифікацію зображень [17], [18], аналіз поведінки людини [19], розпізнавання обличь [20], автономному керуванню [21], [22], тощо.

Задача детекції об'єктів являється невід'ємною частиною наукових областей, які вивчають нейронні мережі, алгоритми комп'ютерного зору та машинного навчання. Отже, будь-який прогрес у цих галузях або в галузі детекції об'єктів матиме великий вплив на інші області, які можна розглядати як сукупність підходів до роботи зі ймовірнісними системи [23], [24], [25], [26]. Задача детекції об'єктів була в центрі уваги досліджень комп'ютерного зору, і нещодавній розвиток глибокого навчання значно покращив його ефективність. Сучасні алгоритми, такі як R-FCN [27], FPN [28], Cascade R-CNN [29] та Mask R-CNN [30], досягли високої точності навіть на складних наборах даних таких як PASCAL VOC [31] та MS COCO [32].

Через велику варіативність в тому з яких ракурсів можна фотографувати об'єкти, з якої дистанції, при якому освітленні, в яких позах та яких умовах перекриття об'єктів

один одним – створення ідеальної системи детекції об’єктів стає складною задачею, яку можна досягти лише при наявності великих об’ємів даних, що повністю покривають всю вище описану варіативність об’єктів та містить інформацію про точну локацію усіх об’єктів, які цікавлять дослідників. Вичерпна анотація місцеположення усіх об’єктів для великого набору даних має лінійну складність відносно кількості картинок в наборі даних та кількості унікальних категорій, а процес розмітки такого набору даних є надзвичайно трудомістким та дорогим процесом. Саме тому в останні роки дослідники приділили багато уваги вирішенню цієї проблеми [33], [34], [35], [36]. Навіть із сучасними підходами до анотування даних [37], [38] розмітка даних для задачі детекції об’єктів все ще більш як в 100 раз дорожче [39], ніж анотування аналогічного набору даних для задачі класифікації об’єктів. Саме тому, існуючі набори даних для детекції об’єктів невеликі за розміром та містять не багато категорій. Найбільш популярними наборами даних для задач детекції об’єктів серед дослідників є PASCAL VOC (12000 зображень та 20 категорій) [31] та MS COCO (120000 зображень та 80 категорій) [32], що значно менше за набір даних ImageNet (14000000 зображень та 1000 категорій), поява якого дала значний поштовх в розвитку сучасних алгоритмів комп’ютерного зору на основі підходів глибокого машинного навчання.

Анотація зображень є однією з найважливіших задач в комп'ютерному зорі [40], яка покладена на людину та полягає в тому, щоб позначити зображення відповідними мітками. Ці мітки обрані інженером, що розробляє алгоритми таким чином, щоб дати моделі комп'ютерного зору інформацію про те, що показано на зображенні. Залежно від задачі кількість міток на одному і тому ж зображенні може сильно змінюватися. Для деяких задач для передачі вмісту всього зображення потрібно лише одна мітка (класифікація зображень). Інші задачі можуть вимагати помітити кілька об'єктів на одному зображенні, кожен з яких має свою унікальну мітку (задача детекції об’єктів).

Вартість розмітки даних для задачі детекції об’єктів сильно залежить від природи даних, середньої кількості об’єктів на зображеннях та кількості унікальних категорій,

розмірів об’єктів, наявності перекриття об’єктів та кількості людей, що будуть анотувати та перевіряти розмітку – чим їх більше, тим більша якість даних. Проаналізувавши популярні платформи для розробки ШІ системи для розмітки даних [7], [42], [43] можна зробити висновок, що середня вартість одного зображення 2-ма людьми складає $2 [39], що виливається в мільйонні втрати для розмітки таких наборів даних аналогічних MS COCO. Основна причина полягає в відсутності широкої автоматизації процесу розмітки даних, що робить його неефективним та довготривалим.

1.2 Визначення платформи підтримки задач штучного інтелекту

Платформи штучного інтелекту – це програмні платформи, які допомагають користувачам легко використовувати різні можливості штучного інтелекту [1]. Ці платформи по підтримці вирішення задач штучного інтелекту, пропонують широкий спектр послуг по розробці та впровадженню алгоритмів штучного інтелекту, що допомагають абстрагувати та спростити складність штучного інтелекту, допомагаючи бізнесу легко розробляти та застосовувати рішення в своєму програмному забезпеченні. Платформи ШІ забирають основні технічні знання, необхідні для доступу до потужності штучного інтелекту, створюючи прості у користуванні інтелектуальні сервіси, якими може користуватися кожен.

Мета платформ штучного інтелекту – пришвидшити застосування штучного інтелекту, доносячи його до кожного розробника та бізнесу. Розробники можуть отримати доступ до передових алгоритмів ШІ, які готові до використання у виді окремих сервісів та послуг (SaaS). Використання таких алгоритмів пришвидшує розробку та впровадження рішень із штучного інтелекту, особливо для малого бізнесу, який не здатен утримувати великий штат розробників.

Деякі платформи дають доступ до налаштування цих платформ на високому рівні, що дозволяє забезпечити бізнесам персоналізовані послуги, якими вони можуть користуватися. Це комп’ютерне програмне забезпечення допомагає компаніям дуже

легко виконувати аналіз даних, проводити просунуту аналітику на базі цих даних, візуалізацію, оптимізацію бізнес-процесів, впроваджувати алгоритми розпізнавання зображень, сентиментальний аналіз та обробку тексту, тощо. Користувачам не потрібно бути технічно досвідченими для використання цих платформ, що значно розширює коло потенційних користувачів платформами для підтримки задач штучного інтелекту.

1.3 Огляд платформ підтримки задач штучного інтелекту

1.3.1 Microsoft Azure AI Platform

Microsoft Azure AI Platform [3] являє собою SaaS платформу, яка пропонує широкий набір рішень для типових задач ШІ в сферах фінансових технологій, виробництва, роздрібної торгівлі, послуг охорони здоров'я тощо. Microsoft Azure AI Platform не лише дає доступ до алгоритмів, які здатні вирішувати типові задачі (розпізнавання лиць, аналіз зображень, детекція фішінгу, тощо), а й дає можливість розробляти власні алгоритми ШІ з використанням Tensorflow [8], PyTorch [9] та Scikit-learn [10]. Головна сторінка Microsoft Azure AI Platform представлена на рисунку 1.1.

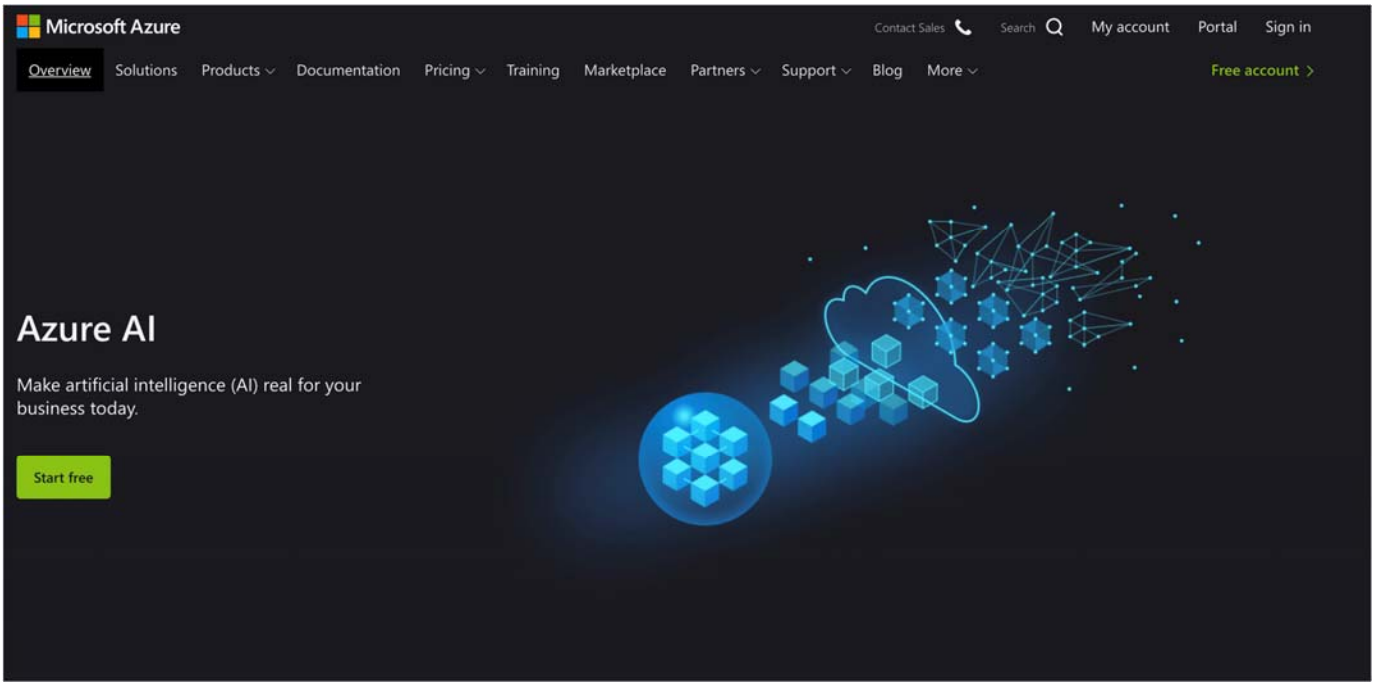


Рисунок 1.1 – Стартова сторінка Microsoft Azure AI Platform [3]

Також варто замітити, що Microsoft Azure AI Platform має зручний та зрозумілий для пересічного користувача перелік послуг, кожен з яких має детальний опис переваг та можливостей для використання в готових програмних продуктах.

1.3.2 Google Cloud AI Platform

Google Cloud AI Platform [4] – це SaaS платформа, що пропонує широкий набір рішень для побудови робочих пайплайнів для роботи з будь-якими задачами штучного інтелекту: підготовку даних (Cloud BigQuery, Cloud Storage, Cloud Data Labeling Service), побудову моделей штучного інтелекту (включаючи сервіс автоматичного пошук найкращого алгоритму Cloud AutoML), перевірки (Cloud What-if Tool), розгортання на базі Google Cloud та управління проектом. Стартова сторінка Google Cloud AI Platform наведена на рисунку 1.2.

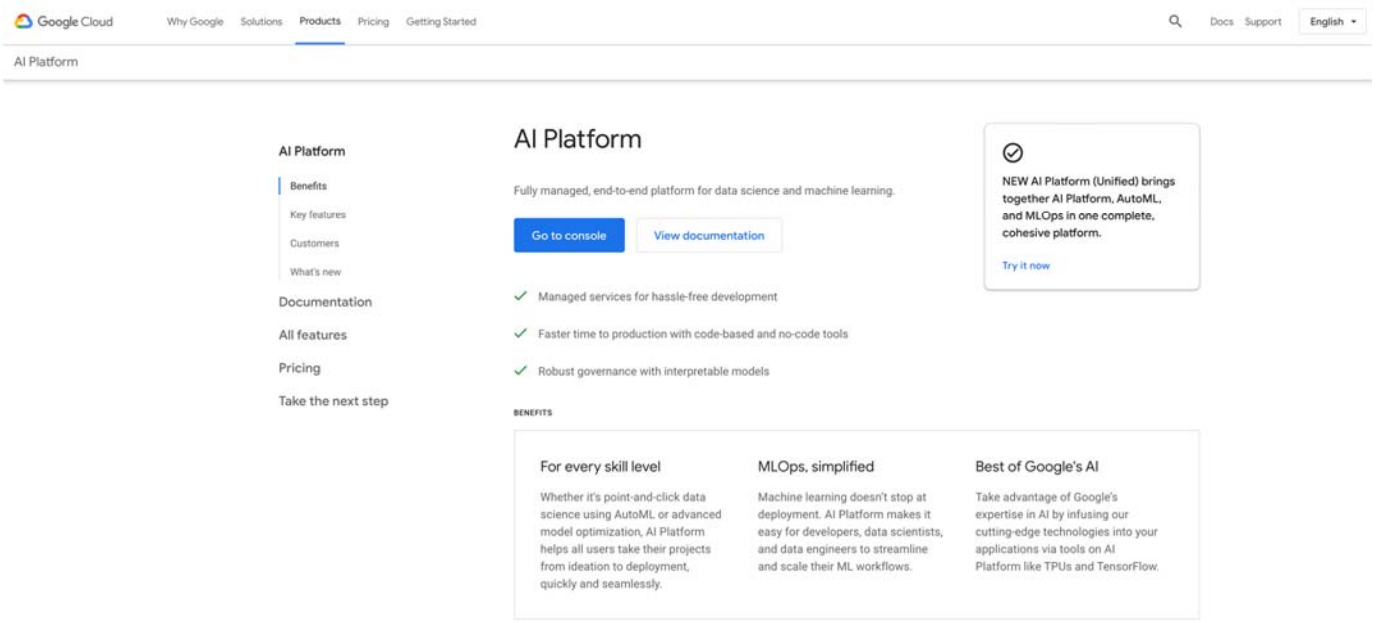


Рисунок 1.2 – Стартова сторінка Google Cloud AI Platform [4]

Хоча Google Cloud AI Platform позиціонує себе як платформа для підтримки задач штучного інтелекту, але вимагає від її користувача значних знань в області штучного інтелекту та інструментарію Google Cloud, що значно звужує коло потенційних

користувачів, але при цьому дозволяє тонко налаштовувати та керувати усіма процесами в проекті.

1.3.3 Amazon AWS SageMaker

Amazon AWS SageMaker [5] являє собою SaaS платформу, яка пропонує повний набір для розробки, тестування та впровадження алгоритмів машинного навчання. Amazon AWS SageMaker має хорошу бібліотеку рішень для типовий задач бізнесу, але при цьому дозволяє швидко розвернути довільний пайлайн по роботі з даними та розробці алгоритмів штучного інтелекту на базі Amazon Web Services. Стартова сторінка Amazon AWS SageMaker наведена на рисунку 1.3.

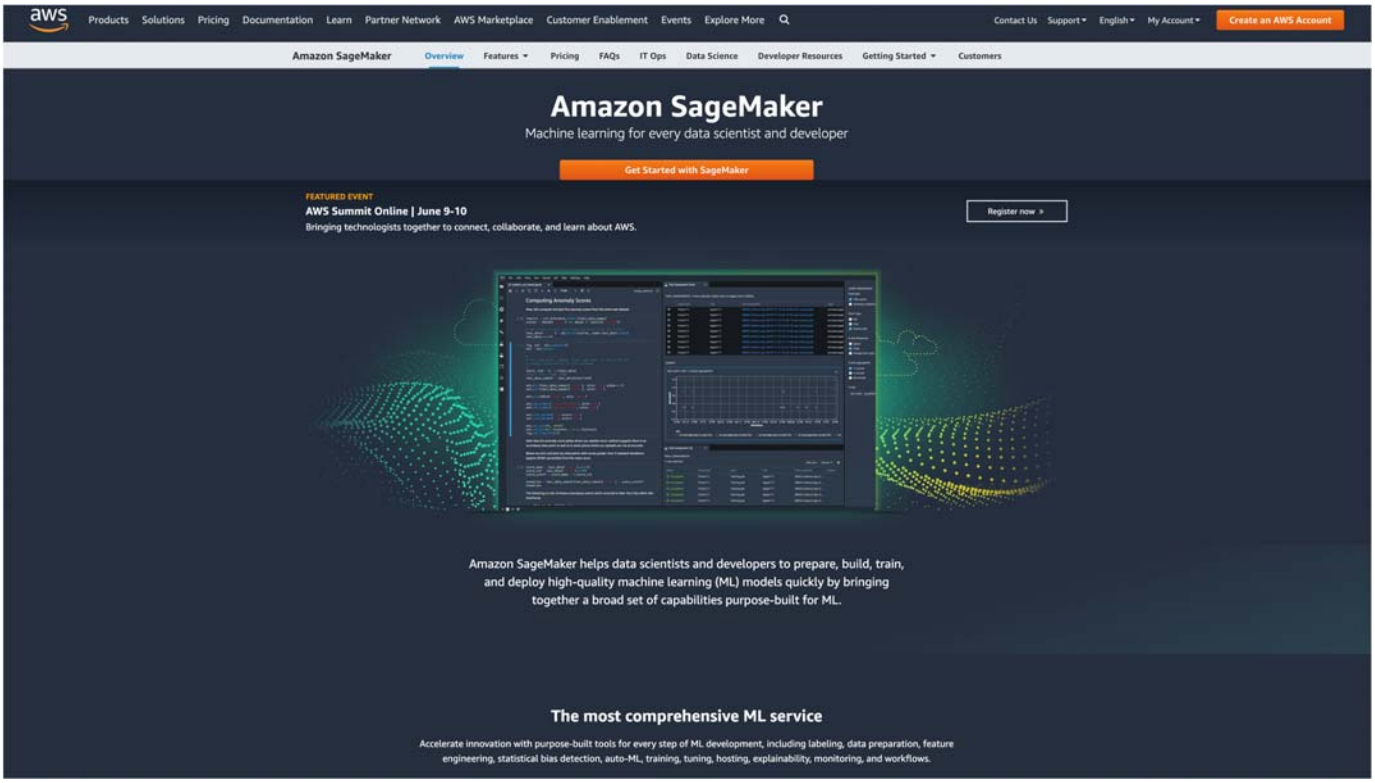


Рисунок 1.3 – Стартова сторінка Amazon AWS SageMaker [5]

Значущою перевагою Amazon AWS SageMaker [5] являється вбудована можливість пришвидшення навчання алгоритмів штучного інтелекту за рахунок

паралелізації процесу навчання на декількох спеціалізованих машина AWS для задач штучного інтелекту.

1.3.4 IBM Watson

IBM Watson [6] являє собою SaaS платформу, яка містить широкий спектр готових рішень для бізнесів в сфері фінансів, реклами, IT, охорони здоров'я, стрімінгових сервісів, прогнозування погоди тощо. Стартова сторінка IBM Watson наведена на рисунку 1.4.

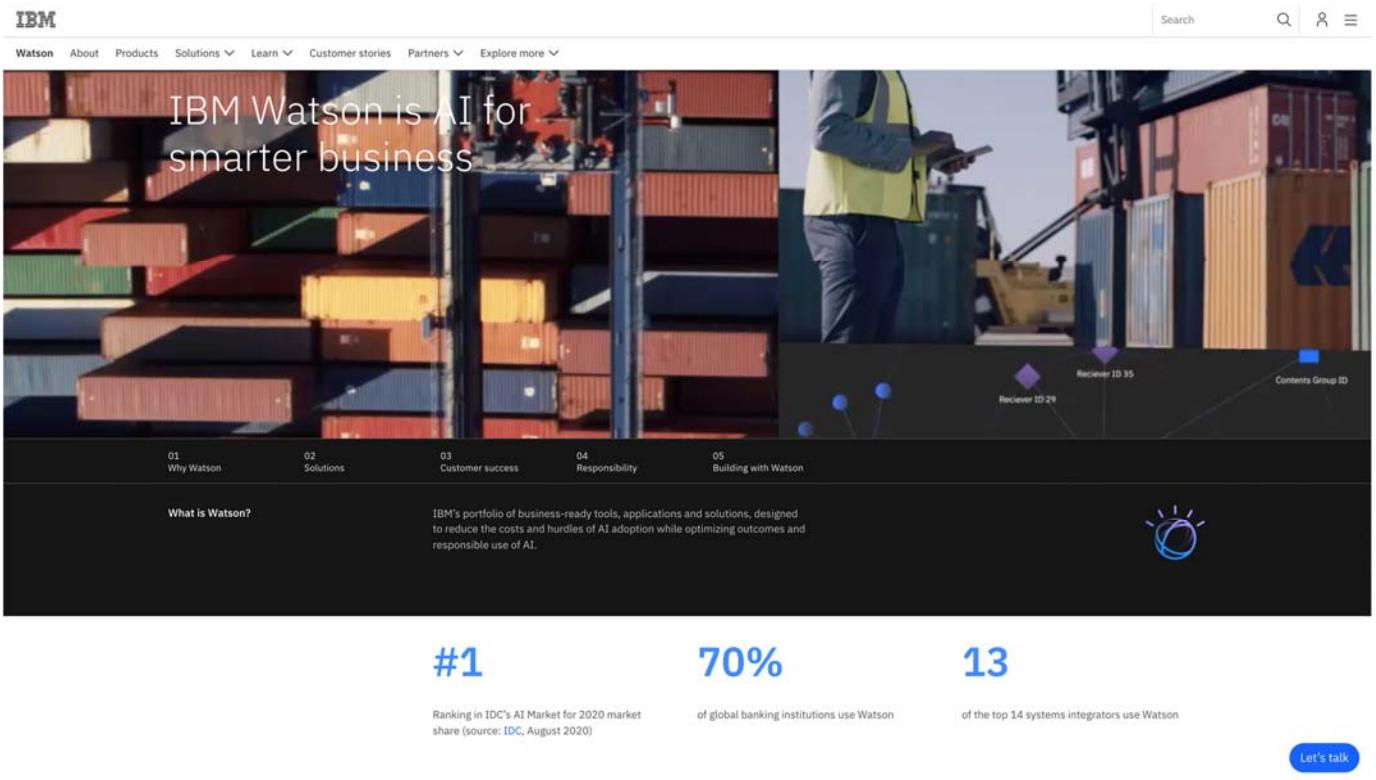


Рисунок 1.4 – Стартова сторінку IBM Watson [6]

IBM Watson [6] має зручний каталог рішень, який дозволяє швидко знайти та почати використовувати готові рішення від IBM для різних задач автоматизації бізнес-процесів, що значно мінімізує поріг входження. Значним недоліком даної платформи являється складність налаштування персоналізованих рішень, що значно збільшує витрати на підтримку таких рішень в довгостроковій перспективі.

1.3.5 Порівняльний аналіз платформ підтримки задач штучного інтелекту

Порівняльний аналіз платформ ШІ розглянутих вище наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз платформ підтримки задач ШІ

Платформа	Робота з алгоритмами	Робота з даними	Переваги	Недоліки
Microsoft Azure AI Platform [3]	Автоматизація розробки, каталог готових алгоритмів	Класичні підходи розмітки даних	Каталог рішень для типових задач, можливість створення складних пайплайнів по роботі з алгоритмами	Класичні підходи до організації розмітки даних
Google Cloud AI Platform [4]	Автоматизація побудови навчального пайплайну	Класичні підходи розмітки даних	Можливість тонкого налаштування, наявність доступу до Google Auto ML алгоритмів	Класичні підходи до організації розмітки даних, відсутність каталогу типових рішень
Amazon AWS SageMaker [5]	Автоматизація розробки	Автоматизація розмітки типових задач	Можливість тонкого налаштування, наявний каталог типових рішень, вбудована можливість дистрибутивного навчання, автомати розмітки даних для типових задач	Складність в створені та підтримці не типових задач
IBM Watson [6]	Каталог готових алгоритмів	Відсутня	Широкий каталог типових задач ШІ	Відсутня можливість тонкого налаштування

Отже, більшість платформ пропонують зручний доступ до бібліотек готових алгоритмів для роботи з типовими задачами, але при цьому розмітку даних для даних

алгоритмів дані платформи перекладаю на користувача платформи, що дещо ускладнює процес взаємодії та створення високоякісних продуктів.

Дана робота націлена на дослідження процесів розмітки даних задля автоматизації процесу розмітки як для типових задач та і не типових задач.

1.4 Задача розмітки даних

Основним завданням машинного навчання – навчити алгоритм, будь то традиційну статистичну модель або сучасну нейронну мережу, шукати і використовувати закономірності у навчального наборі вхідних та вихідних даних. При цьому ці вхідні та вихідні данні можуть мати форму тексту, картинки, аудіо або відео файлів. Наприклад, ми хочемо навчити модель визначати тональність голосу людини, яка телефонує в call-центр. Тоді задача, яку буде вирішувати модель: по вхідному наборі аудіо даних, визначити яка інтонація у абонента. А вихідними даними будуть мітки – позитивна чи негативна тональність. Аналогічний приклад можна привести для відеозаписів, які фіксують порушення автомобілем лінії STOP перед світлофором. Щоб штучний інтелект навчився давати правильні відповіді для поставленої задачі, за нього це завдання спочатку повинна вирішити людина – підготувати дані для машинного навчання. Цей процес підготовки даних для моделей машинного навчання і називається розміткою даних.

Аналогія, що дані – нафта XXI століття, особливо влучна якщо згадати, що саму по собі сиру нафту особливо ніде не можливо застосувати. Перед використанням нафту необхідно очистити та переробити. Аналогічно справа виглядає і з нашою «новою нафтою» – навіть зібравши велику кількість неструктурованих даних ви не зможете використати її в розробці своїх програмних продуктів на основі машинного навчання – спочатку її необхідно буде розмітити та почистити від не вірних або помилкових даних. Узагальнено процес отримання якісного набору даних [44] відображено на рисунку 1.5.

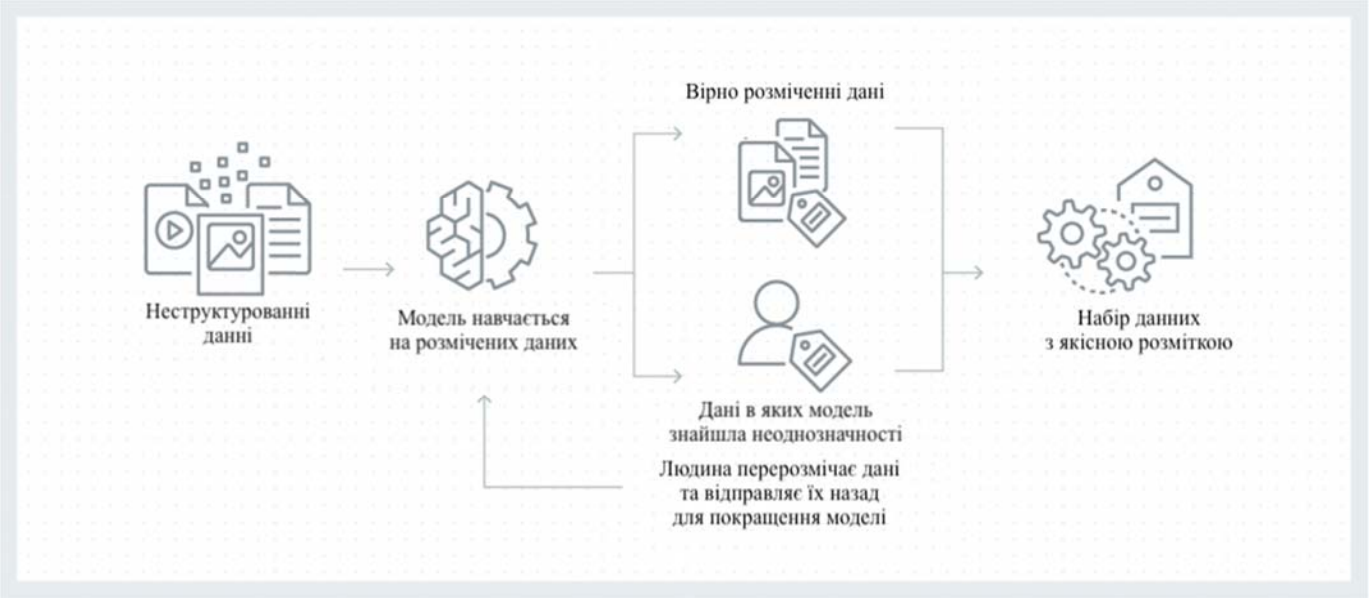


Рисунок 1.5 – Процес отримання якісного набору даних

Оскільки якісна розмітка даних – це ітеративний процес, який включає взаємодію з людиною – розмітчиком даних, то сам цей процес трудомісткий та тривалий по часу. Згідно з аналітикою Cognilytica [44] – компанії витрачають в середньому до 80% часу на роботу з даними та лише 20% – на побудову моделей машинного навчання та їх розгортання. Діаграма розбивки часу на різні активності в сучасних проектах з використанням машинного навчання наведено на рисунку 1.6.

Користуючись тим фактом, що створення великих наборів даних довготривалий та дороговартісний процес, технологічні гіганти захоплюють більшу частину програмного забезпечення пов’язаного з машинним навчанням, оскільки здатні швидко створювати великі набори даних за допомогою використання великої кількості розмітчиків – як штатних так і контрактних – так званих ферм по розмітці даних [45] (рисунок 1.7). Тому малим компаніям та стартапам необхідно знаходити інші підходи до використання публічних та нішевих наборів даних для вирішення своїх задач.

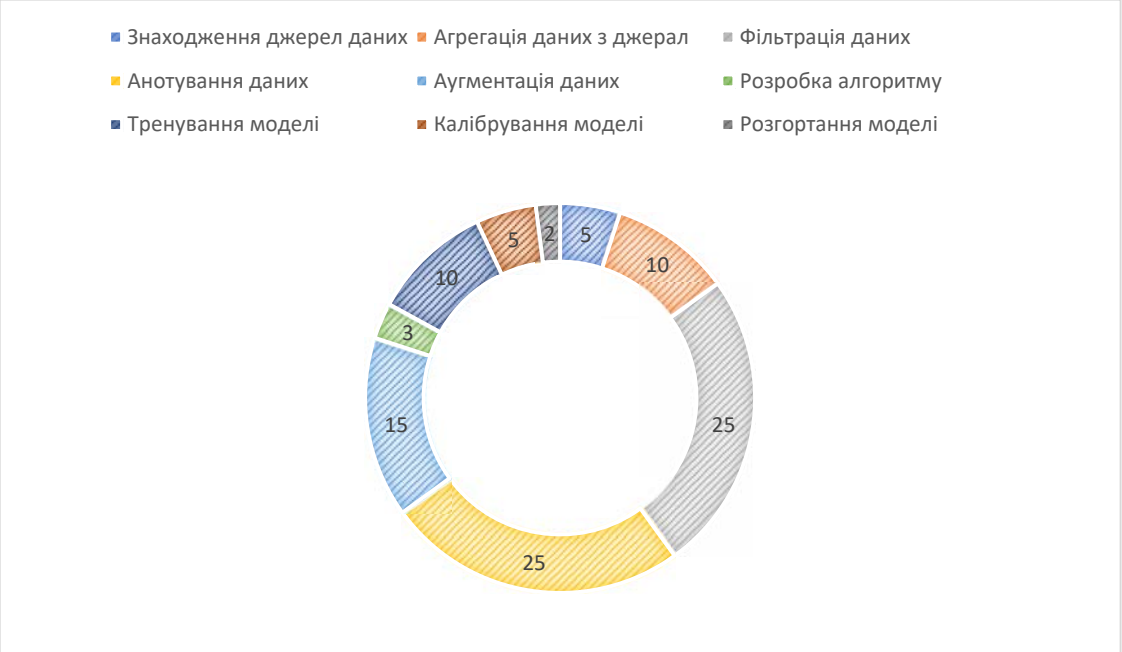


Рисунок 1.6 – Діаграма розбивки часу на різні активності в сучасних проектах з використанням машинного навчання

Рисунок 1.7 – Фабрика розмітки даних в Китаї

1.5 Організація процесу розмітки даних на платформах штучного інтелекту

Згідно з дослідженням Bloomberg [46] весь процес розмітки даних для проектів з використанням машинного навчання можна розділити на 9 ключових етапів:

1) формалізації цілей розмітки даних та визначення основних зацікавлених сторін;
2) формалізація проекту по розмітці даних;
3) планування дорожньої карти розмітки даних;
4) вибір розмітчиків даних;
5) вибір утиліти для розмітки даних;
6) створення керівних принципів;
7) навчання розмітчиків даних;
8) управління процесом розмітки даних;
9) використання розмічених даних.

В наступних підрозділах ми розглянемо основні положення та кроки, які необхідно виконати для того щоб побудувати ефективний процес розмітки даних.

1.5.1 Формалізації цілей розмітки даних та визначення основних зацікавлених сторін

Даний етап розпочинаєте з визначення основних зацікавлених сторін, зазвичай до них відносяться:

– проектний менеджер (і/або продуктовий менеджер), який визначає прикладне застосування проекту, розуміє його потенційний вплив на бізнес, надає пріоритети функціям на основі потреб користувачів та може мати інші доменні знання;

– менеджер по розмітці даних, який несе відповідальність за нагляд за проектом анотацій, відбирає та керує робочою силою, забезпечує якість та узгодженість даних;

– головний інженер, який відповідає за впровадження рішення, який виступає замовником розмітки даних та який надає інформацію щодо технічної стратегії та доцільності прийнятих рішень/компромісів в процесі розмітки даних.

Після визначення основних зацікавлених сторін необхідно встановити кінцеві цілі. Визначення цих цілей допоможуть учасникам проекту зрозуміти його мету та забезпечити, щоб усі працювали над досягненням цих цілей.

Останнім кроком даного етапу являється вибір методу регулярного та чіткого спілкування між зацікавленими сторонами на всіх етапах проекту. Це дозволить зберегти фокус на кінцевих цілях проекту. Цей крок важливий не залежно від того, чи бере участь у проекті одна команда чи декілька команд, які знаходяться в куточках планети.

1.5.2 Формалізація проекту по розмітці даних

На даному етапі зацікавлені сторони повинні визначити характеристики даних, що підлягають розмітці, спосіб відбору даних та тип розмітки даних.

Дослідження даних є важливим початковим етапом процесу формалізації проекту, оскільки розуміння особливостей даних, обмежень, закономірностей та крайових випадків дозволить тим, хто бере участь у проекті, приймати зважені рішення щодо типу необхідної розмітки.

Як тільки зацікавлені сторони зрозуміють дані, вони повинні вирішити, як визначити підвибірку даних для анотації та чи потрібна попередня обробка даних перед їх розміткою.

Вкінці зацікавленні сторони повинні визначитися з набором міток, які підлягають анотацій.

1.5.3 Планування дорожньої карти розмітки даних

На цьому етапі важливо, щоб усі зацікавлені сторони були залучені до побудови дорожньої карти проекту. Дорожня карта допоможе повідомити про очікування зацікавлених сторін, обмеження та залежності, що мають великий вплив на бюджет та успіх проекту. Під час створення дорожньої карти сторони повинні забезпечити чіткі терміни з детальною інформацією про проекти, їх залежності, а також ключові етапи. При встановленні термінів важливо враховувати час для створення керівних принципів та навчання робочої сили.

1.5.4 Вибір розмітчиків даних

Вибір розмітчиків даних відіграє ключову роль в успішному завершені проекту розмітки даних та вимагає глибокого розуміння вимог до конфіденційності та безпеки даних, складності проекту та бюджетних обмежень. Менеджер по розмітці даних повинен завжди брати участь у виборі розмітчиків; це рішення ніколи не повинно прийматися виключно проектним менеджером або головним інженером.

1.5.5 Вибір утиліти для розмітки даних

Під час вибору застосунку в якому буде виконуватися розмітка даних зацікавленим сторонам слід звернути увагу на:

– зацікавлені сторони повинні прагнути використовувати стандартну інфраструктуру, якщо вона здатна виконати поставлені завдання;

– основними драйверами для вибору інструментів має бути забезпечення конфіденційності даних та простота доступу робочої сили до даних;

– зацікавлені сторони повинні оцінити технічний беклог по впровадженню обраної утиліти;

– важливо врахувати, як інструмент полегшить управління проектами;

– зацікавлені сторони повинні встановити бажаний UI/UX для інструменту анотації та інтерфейсу проекту.

1.5.6 Створення керівних принципів

На цьому етапу відбуваєте створення керівних принців, які повинні покривати типові запитання, та якими повинні керуватися розмітчики даних при виконанні своїх щоденних обов'язків – які об'єкти необхідно розмічати, як працювати з крайовими випадками та інші.

Одна команда (в ідеалі, одна особа) повинна займатися розробкою та оновленням керівних принципів. Це допомагає забезпечити внутрішню узгодженість (тобто відсутність суперечливого змісту) та читабельність. Менеджер проекту розмітки, як правило, найкраще підходить для написання інструкцій щодо анотацій, враховуючи його розуміння набору даних, кінцевої мети та технічних вимог.

Хоча одна людина повинна нести відповідальність за розробку та оновлення керівних принципів, остаточні рекомендації завжди повинні бути співпрацею між усіма зацікавленими сторонами. Менеджер проекту розмітки повинен забезпечити чіткість та узгодженість вказівок на основі своїх знань про набір даних. Менеджери інженерних команд повинні забезпечити, щоб настанови узгоджувались із потребами, алгоритмів для яких відбувається розмітка даних. Менеджери проекту повинні забезпечити, щоб керівні принципи враховували усі вимоги до продуктової частини та очікування користувачів по продукту.

Після розробки керівних принципів необхідно провести пілотний тести для перевірки принципів, щоб підтвердити, що вони дають бажаний результат. Тестери можуть розмітити невелику кількість даних, а потім колективно переглянути їх, щоб оцінити, чи слід вносити зміни до рекомендації.

Після внесення всіх правок в керівні принци – вони повинні бути розміщені в утиліті по розмітці даних, де розмітчики даних можуть легко звернутися до них під час робочого процесу.

Зміни до керівних принципів повинні повідомлятися розмітчикам даних у письмовій формі, щоб всі отримували ту саму інформацію одночасно і могли підтримувати узгодженість в розмітці даних. Усі зацікавлені сторони повинні ретельно розглядати будь-які модифікації керівних принципів, які можуть призвести до змін в раніше анотованих даних, оскільки ці модифікації, ймовірно, вплинуть на вартість та терміни проекту.

1.5.7 Навчання розмітчиків даних

Етап навчання розмітчиків даних є важливою частиною процесу, щоб гарантувати, що вони адекватно розуміють проект та створюють розмітку, яка є одночасно точною та узгодженою. Як зазначено у розділі 1.2.3, графік проекту повинен враховувати період навчання, тривалість якого залежить від обраних розмітчиків даних, складності проекту та застосовуваного методу забезпечення якості даних.

Підходи до навчання розмітчиків будуть залежати від природи даних, часових рамок проекту, складності роботи з утилітою по розмітці даних, тощо. Важливо використовувати керівні принципи в процесі навчання розмітчиків, щоб правила залишалося узгодженими. Якщо керівні принципи потребують доповнення, додаткове навчання розмітчиків слід забезпечити у письмовій формі або записати відео-презентацію.

Під час навчального періоду слід заохочувати запитання від розмітчиків, щоб якомога швидше прибрати будь-яку плутанину або нерозуміння в керівних принципах.

1.5.8 Управління процесом розмітки даних

Під час цього етапу зацікавлені сторони повинні переглянути інструмент анотації та підготовку команди розмітки даних, щоб підтвердити, що вибір відповідає як цілям, так і вимогам до якості розмітки даних. Після цього відбувається запуск процесу розмітки. Менеджер по розмітці даних організовує процес перевірки якості даних та займається вирішенням поточних проблем на проекті. За необхідності зацікавлені стороно можуть повернутися до попередніх етапів для внесення змін.

1.5.9 Використання розмічених даних

Після завершення розмітки зацікавлені сторони повинні визначити, чи потрібно розмічати більше даних, ніж передбачалося спочатку, для досягнення кінцевої мети проекту. Враховуючи, що розмітка додаткових даних буде вимагати більше ресурсів з точки зору часу, бюджету та зусиль, то подальший збір анотацій повинен бути ретельно продуманий.

Якщо отримана кількість даних та їх якість задовольняють усі зацікавлені сторони, то проект можна вважати успішно виконаним, а інженери можуть приступати до побудови моделей машинного навчання на основі отриманих даних.

1.6 Підходи організації розмітки даних з використанням платформ штучного інтелекту

З огляду на те, що розмітка даних часто трудомістка та складна, то від вірної організації розмітки даних залежить успішність проекту, оскільки саме від процесів залежать якість даних, час виконання та вартість розмітки. Аналітики Lionbridge.ai виділять 5 основних підходів [47] до організації процесу розмітки даних:

- набір внутрішньої команди розмітки;
- аутсорсинг задачі спеціалізованим компаніям по розмітці даних;
- розмітка на краудсорсингових площадках;

– створення синтетичних даних;

– авторозмітка даних.

1.6.1 Набір внутрішньої команди розмітки

Як випливає з назви, цей підхід передбачає набір команди розмітчиків даних в штат співробітників. Підхід має ряд безпосередніх переваг: відстеження прогресу є простим, а точність та рівень якості отриманих даних високий. Однак, поза великими компаніями, що мають великі штатні групи по розмітці даних, такий підхід до організації розмітки даних не є життєздатним.

1.6.2 Аутсорсинг задачі спеціалізованим компаніям по розмітці даних

Аутсорсинг задачі розмітки даних – це хороший варіант для створення тимчасової команди професіоналів для роботи над проектом протягом певного періоду часу. Аутсорсингові компанії зазвичай мають добре налагоджені процеси по організації процесу найму розмітчиків, їх навчанню, організацією процесу розмітки, перевірки та комунікації нових змін. Ви, як замовник, маєте можливість відбирати кандидатів на проект лише з відповідним набором навичок. Даний варіант чудово підходить малим компанія, які мають чітке розуміння щодо фінального результату, оскільки дозволяє добитися його за короткий проміжок часу.

1.6.3 Розмітка на краудсорсингових площадках

Платформи для краудсорсингу – це спосіб залучити допомогу людей по всьому світу для роботи над певними завданнями. Оскільки роботу на краудсорсингових площадках можна взяти з будь-якої точки світу і виконувати їх у міру того, як завдання стають доступними, це надзвичайно швидко, ефективно та економічно. Однак платформи краудсорсингу можуть відрізнятися один від одного з точки зору якості «працівників», забезпечення якості та інструментів управління проектами та

працівниками. Тому важливо знати, як платформа підходить до цих факторів, розглядаючи різні варіанти краудсорсингових платформ.

1.6.4 Створення синтетичних даних

Синтетичні дані – це підхід до генерації нових даних, що містять атрибути, необхідні для вашого проекту. Одним із способів створення таких даних – генерація за допомогою генеративних змагальницьких мереж (GANs). GANs використовують дві нейронні мережі – генератор та дискримінатор, які, відповідно, змагаються у створенні реалістичних «підробок» даних та відрізненню реальних даних від підроблених, що призводить до створення нових реалістичних даних. GANs дозволяють створювати абсолютно нові дані з вже існуючих наборів даних. Це робить їх ефективними для отримання великої кількості високоякісних даних за короткий проміжок часу. Однак в даний час методи з використанням GANs вимагають великих обчислювальних потужностей, що робить їх дуже дороговартісним підходом.

1.6.5 Авторозмітка даних

Авторозмітка даних – це процес використання скриптів для автоматичного маркування даних. Цей процес може автоматизувати завдання, включаючи розмітку зображень та тексту, що позбавляє потреби у великій кількості людських маркувальників. Комп’ютерна програма також не потребує відпочинку, тому ви можете очікувати результатів набагато швидше, ніж під час роботи з людьми. Однак даний підхід все ще далекий від досконалості. Тому програмне маркування даних часто поєднують із спеціальною командою із забезпечення якості.

1.6.6 Порівняльний аналіз підходів до організації розмітки даних

Порівняльний аналіз усіх вище наведених підходів наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз підходів до організації розмітки даних

Підхід	Переваги	Недоліки
Внутрішня команда розмітки	Простота в відслідковуванні прогресу; можливість внесення правок в правила; швидкість комунікації з усіма зацікавленими сторонами; висока якість отриманих даних.	Довготривалий якщо відсутні налагоджені процеси; необхідно наймати та навчати розмітчиків, погано масштабується.
Аутсорсинг розмітки	Не потрібно налагоджувати процеси та набирати команду; простота в управлінні проектом; висока якість отриманих даних.	Дороговизна; затримки в комунікації; ускладнений процес в разі необхідності внесення право до правил/задач.
Краудсорсинг площадки	Масштабованість процесу розмітку; залучення розмітчиків з різним бекграундом; швидкість розмітки даних; низька ціна одиниці даних.	Не можливо гарантувати якість даних; необхідність в багаторазовій перевірці щоб отримати якісні дані; складність в відслідковуванні статусу розмітки; не можливо розмічати приватні дані користувачів.
Синтетичні дані	Можливість генерувати великі об’єми даних за короткі проміжки часу; висока якість даних.	Дороговизна, яка викликана необхідність в великій кількості обчислювальних потужностей.
Авторозмітка даних	Масштабованість процесу; можливість розмітки великих об’ємів даних за короткі проміжки часу; простота в використані, низька вартість одиниці даних.	Низька якість отриманої розмітки, необхідність перевірки результуючих даних людиною.

1.7 Огляд інструментів для розмітки даних

Розмітка об'єктів на зображеннях – трудомістке і досить дороговартісне завдання, особливо якщо потрібно виконати розмітку великих наборів даних. Для спрощення цього процесу створюється спеціальне програмне забезпечення – інструменти розмітки даних. Від вірного вибору інструменту для розмітки даних залежить швидкість розмітки та якість даних, тому даний розділ посвячений огляду популярних відкритих засобів розмітки даних.

1.7.1 LabelImg

LabelImg – безкоштовний відкритий графічний інструмент для розмітки зображень, написаний на мові програмування Python, який використовується для виділення об'єктів на зображенні [48]. Розмітку можна зберегти як XML-файли аналогічно до формату PASCAL VOC [31] . LabelImg дозволяє створювати обрамляючі прямокутники (bounding boxes) для розмітки об'єктів у графічному інтерфейсі Qt. Інтерфейс LabelImg наведено на рисунку 1.8.

Для старту роботи з LabelImg нехідно мати певний рівень технічної обізнаності (наприклад, уміння використання командного рядка). Найпростіший спосіб розвернути LabelImg – через pip, який передбачає запуск у командному рядку команди “pip install labelImg”. Після інсталяції запуск LabelImg буде доступний через командний рядок.

LabelImg – чудовий інструмент для невеликих за розміром компаній, оскільки дозволяє відразу приступити до розмітки даних без необхідності створювати дороговартісну інфраструктуру та проводити найм співробітників, які б підтримували цю її в робочому стані. Також LabelImg відрізняється інтуїтивним інтерфейсом в якому просто розібратися і почати розмічати дані. До недоліків даної утиліти можна віднести відсутність версійонування розмітки та складнощі в розмітці зображень з великою кількістю категорій і/або об’єктів.

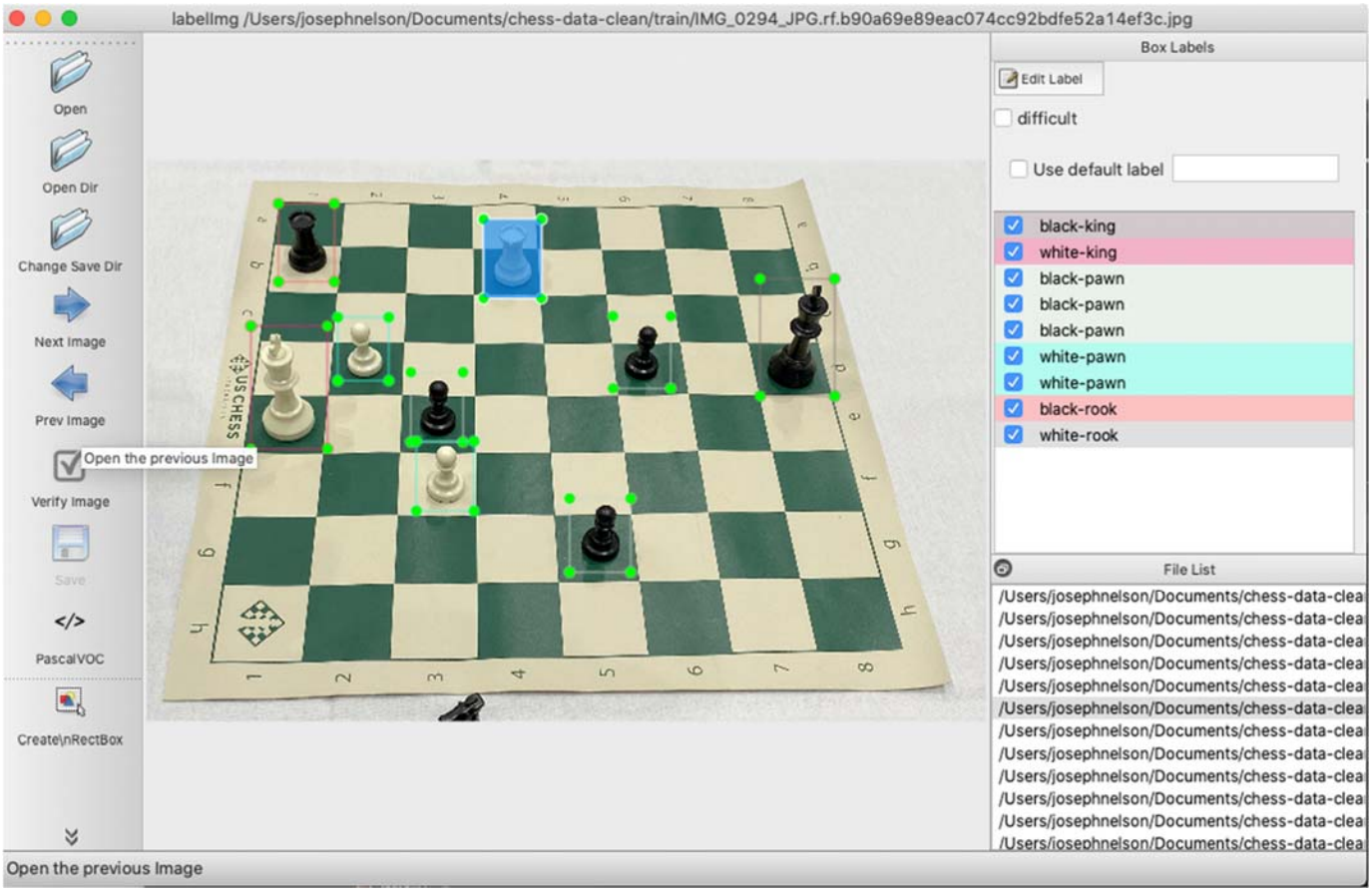


Рисунок 1.8 – Інтерфейс утиліти LabelImg

1.7.2 Computer Vision Annotation Tool

Computer Vision Annotation Tool (CVAT) – це інструмент з відкритим кодом для розмітки цифрових відео та зображень [49]. Основним його завданням є надання користувачеві зручних і ефективних засобів розмітки наборів даних. CVAT задумувався як універсальний сервіс, що підтримує різні типи і формати розмітки.

Для користувачів CVAT представляє собою web-додаток, що працює в браузері. Він підтримує різні сценарії роботи і може бути використаний як для персональної, так і для командної роботи. Основні завдання машинного навчання з учителем в області обробки зображень можна розбити на три групи:

– детектування об'єктів;
– класифікація зображень;

– сегментація зображень.

CVAT можна використати для розмітки даних у всіх цих сценаріях. Інтерфейс CVAT наведено на рисунку 1.9.

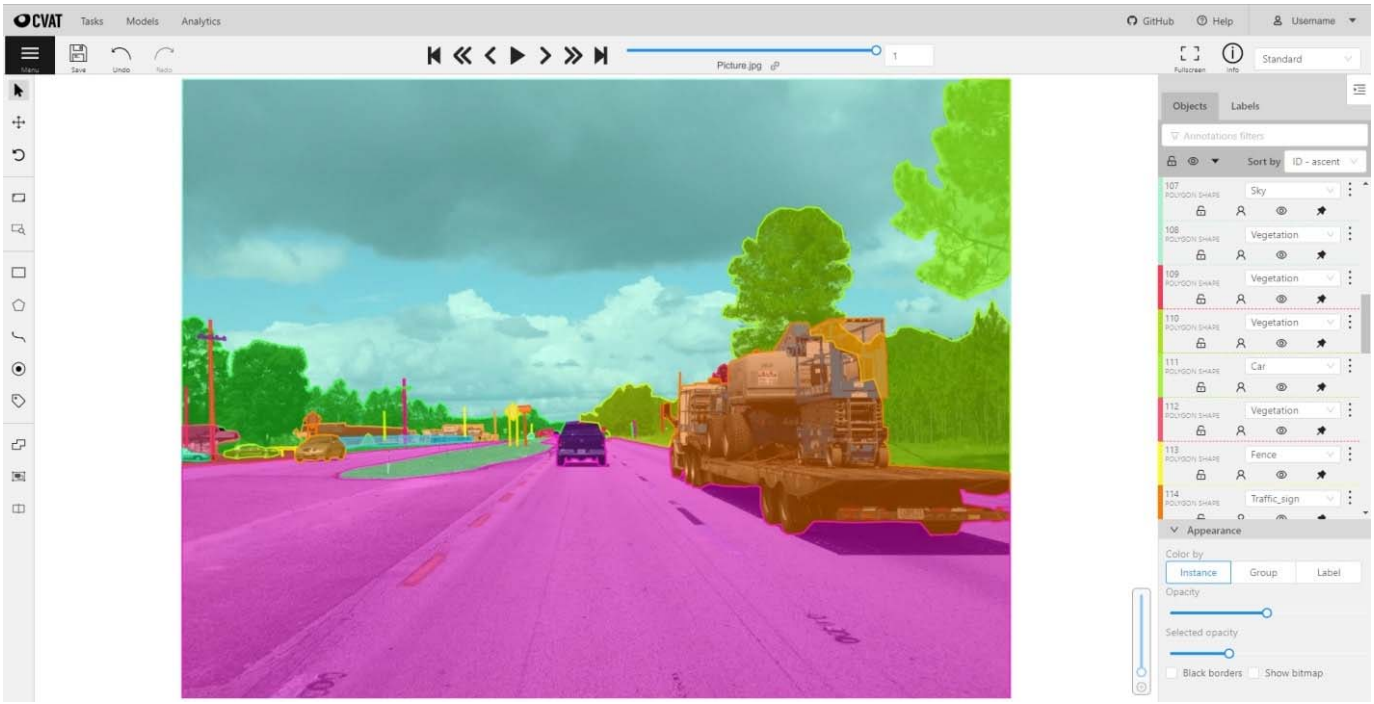


Рисунок 1.9 – Інтерфейс утиліти CVAT

До переваг CVAT можна віднести:

– відсутність необхідності інсталяції інструменту у розмітчиків даних, для створення завдання або розмітки даних досить відкрити посилання в браузері;

– можливість спільної роботи, існує можливість зробити задачу загальнодоступною для користувачів і розпаралелити роботу над нею;

– простота розгортання, установка CVAT в локальній мережі можна виконати лише за допомогою декількох команд Docker;

– автоматизація процесу розмітки, інтерполяція, наприклад, дозволяє отримати розмітку на безлічі кадрів, при реальній роботі лише над деякими ключовими;

– досвід професіоналів, інструмент розроблявся за участю анотаційної і декількох алгоритмічних команд;

– підтримка різних анотаційних сценаріїв;

– просунута система стрімінга зображень і відео з сервера на клієнт;

– відкритий вихідний код під простий і вільною ліцензією MIT.

До недоліків CVAT можна віднести:

– обмежену підтримку браузерів, CVAT не тестувався на великій кількості браузерів, рекомендується використовувати Google Chrome останньої версії або інший браузер на движку Chromium, очікується робота в Mozilla Firefox. Однак, наприклад в Safari на даний момент існують проблеми з роботою утиліти;

– відсутність широкого покриття тестами, розроблено деякі модульні тести, але вони не покривають значну частину функціоналу;

– відсутня документація по вихідного коду, включитися в розробку може бути досить важко;

– обмеження по продуктивності, зі збільшенням вимог за обсягами розмітки можна зіткнутися з різними проблемами, як наприклад, обмеження Chrome Sandbox по використанню оперативної пам'яті.

1.7.3 V7 Darwin

V7 Darwin – комерційна платформа для розмітки даних. Вона дає доступ до автоматизованої анотації зображень та навчання нейронних мереж з активним навчанням, що створює ідеальну піксельну анотацію, що дозволяє в 10 разів пришвидшити розмітку даних в порівнянні з традиційними інструментами [50]. Платформа дозволяє ефективно співпрацювати різним командам, а наявністю утиліт по менеджменту процесу розумітки дозволяє в деталях відслідковувати прогрес в розмітці даних. Інструменти API та CLI дозволяють гнучко працювати з конвеєром даних та моделями машинного навчання. Платформа добре масштабується відповідно до вимог користувача. Документація містить детальну та актуальну інформацію про кожен набір

інструментів на платформі, що робить його зручним у використанні. Інтерфейс V7 Darwin наведено на рисунку 1.10.

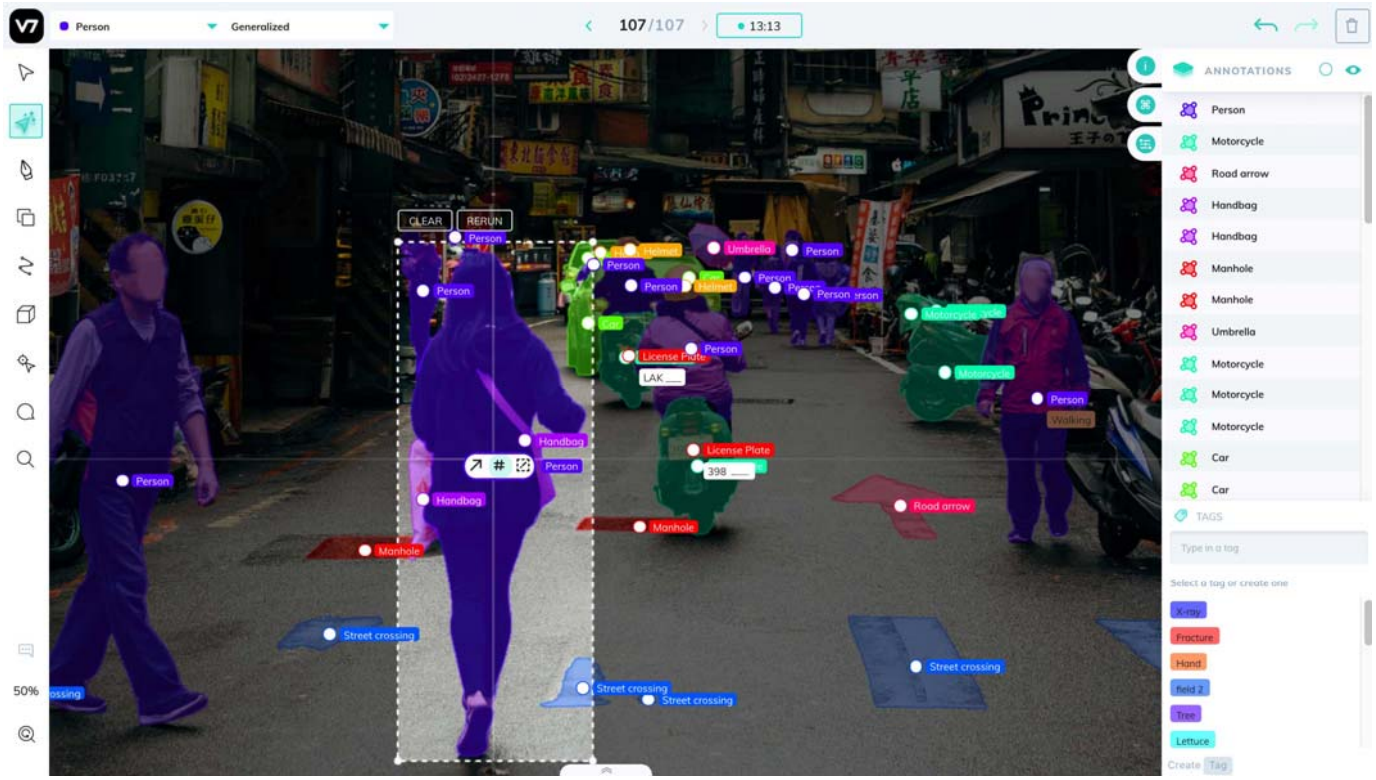


Рисунок 1.10 – Інтерфейс утиліти V7 Darwin

1.8 Постановка задач

Проведений аналіз існуючих платформ по підтримці вирішення задач штучного інтелекту показав, що вони пропонують широкий спектр послуг по розробці та впровадженню алгоритмів штучного інтелекту. Однак, головним недоліком даних платформ є те, що вони концентруються лише на створенні алгоритмів штучного інтелекту, в той час як робота з даними займає близько 80% усього часу і автоматизація процесу розмітки даних або відсутня, або підтримується лише для типових задач, що збільшує вартість розробки програмного забезпечення з використанням алгоритмів штучного інтелекту.

Тому були сформовані наступні задачі дослідження для даної магістерської роботи:

– дослідити наявні підходи до автоматизації розмітки даних в програмних платформах для вирішення задач штучного інтелеку;

– дослідити наявні підходи донавчання алгоритмів машинного навчання;

– удосконалити процес авторозмітки даних для задач детекції об’єктів з метою пришвидшення розмітки великих наборів даних;

– розробити програмне забезпечення авторозмітки даних для задач детекції об’єктів;

– виконати експериментальне дослідження характеристик розробленого програмного забезпечення.

1.9 Висновки до розділу

У даному розділі був проведений порівняльний аналіз існуючих платформ для підтримки вирішення задач штучного інтелекту. Даний аналіз показав, що у розглянутих платформах автоматизація процесу розмітки даних або відсутня, або наявна лише для типових задач, що призводить до неефективної роботі з даними та до збільшення термінів створення програмного забезпечення на основі алгоритмів штучного інтелекту.

Також проведено розгорнутий аналіз усіх процесів розмітки даних та розглянуто різні підходи до організації процесу розмітки даних, які використовуються як в розглянутих платформах підтримки штучного інтелекту, так і в індустрії штучного інтелекту загалом.

Серед розглянутих підходів до організації процесу розмітки даних, найбільш ефективним є підхід авторозмітки даних, який передбачає автоматизацію процесу розмітки даних шляхом використання алгоритмів штучного інтелекту для попередньої розмітки даних – створення чорнової розмітки даних. Далі таку чорнову розмітку повинна переглянути людина та лише виправити помилки, допущені алгоритмом, а не проводити розмітку даних вручну, що призводить до значного підвищення ефективності процесу розмітки даних.

Наявні програмні засоби для авторозмітки даних передбачають лише роботу з типовими задачами, оскільки використовують уже готові алгоритми штучного інтелекту, а не навчають їх в процесі розмітки даних. Саме тому актуальною є задача розробки програмного забезпечення розмітки даних, що дозволить пришвидшити розмітку надвеликих об’ємів даних.

2 ОГЛЯД ПІХОДІВ ТА МЕТОДІВ ПОСТІЙНОГО ДОНАВЧАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1 Машинне навчання

Машинне навчання або Machine learning – один з підрозділів алгоритмів штучного інтелекту, що вивчає алгоритми, які дозволяють комп'ютеру робити висновки на підставі даних, не слідуючи жорстко заданим правилами [51]. Тобто машина може знайти закономірність у складних завданнях, які мозок людини не здатен знайти.

Ціль машинного навчання – частково або й повністю автоматизувати рішення різних складних аналітичних задач. На даний момент машинне навчання охоплює широкий спектр додатків від банків, ресторанів, заправок до роботів на виробництві. Нові завдання, що виникають практично щодня, призводять до появи нових напрямків машинного навчання.

Машинне навчання будується на трьох китах (рисунок 2.1):

– даних – базова інформація, сюди входять будь-які вибірки даних, роботі з якими потрібно навчити систему;

– ознаках – ця частина роботи проводиться в тісній співпраці з власником системи, оскільки потребує чіткого визначення ключових бізнес-потреби, які відграють ключову роль до визначення характеристики та властивості системи;

– алгоритм – метод для вирішення поставленого бізнес-завдання.

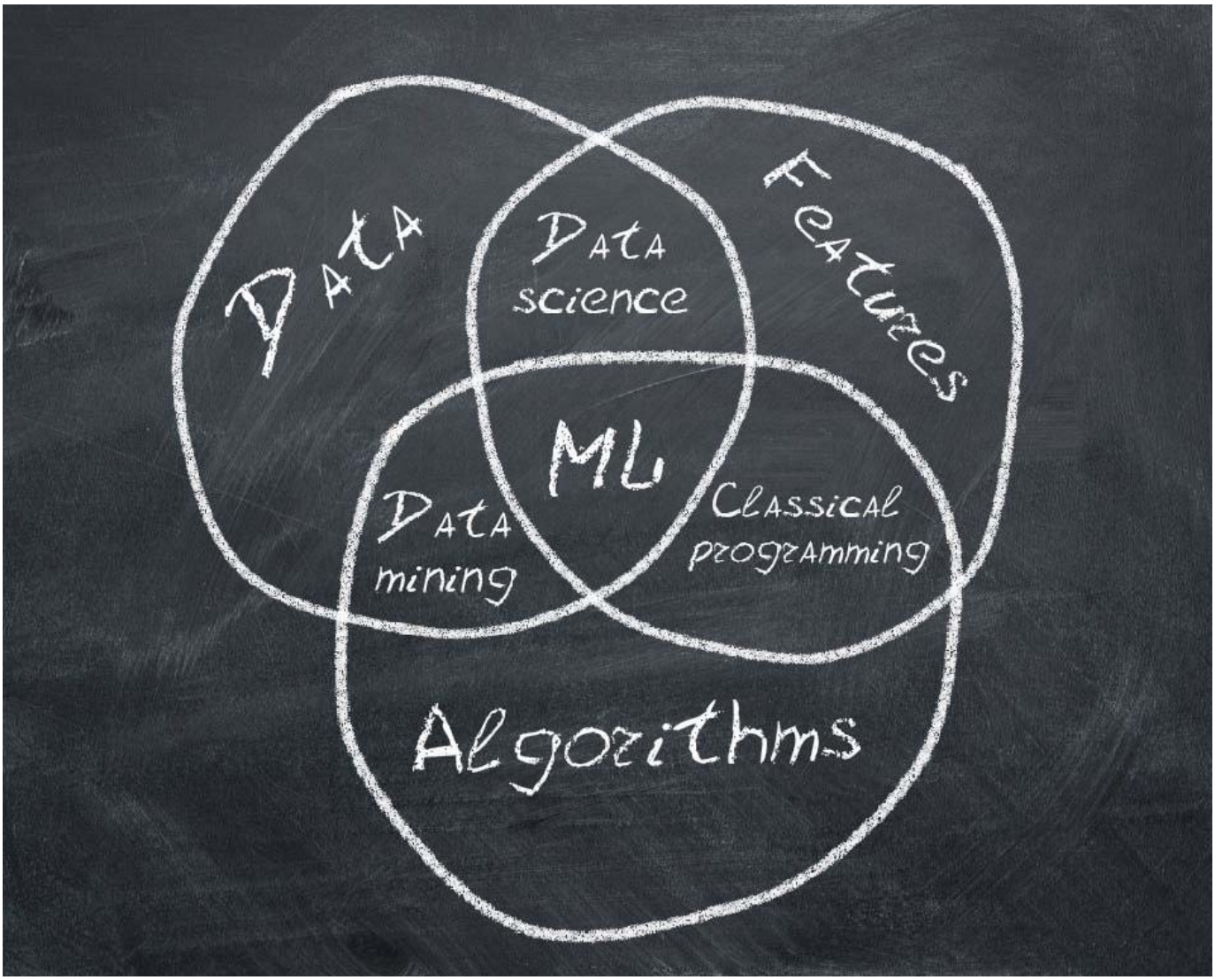


Рисунок 2.1 – «Три кити» машинного навчання

2.1.1 Дані

Чим більше даних завантажити в алгоритм, тим краще, швидше і точніше він працюватиме [51]. Самі дані безпосередньо залежать від завдання, яке стоїть перед машиною. Найскладніша і водночас найбільш об'ємна частина роботи – збір даних. Існує два методи збору даних: вручну і автоматично. Ручний метод набагато повільніший, але при цьому точний. Автоматичний же набагато швидший, але при цьому він допускає велику кількість помилок в даних. Хороша вибірка даних коштує дорогого, адже саме

вона відповідає за точність прогнозування, яку ви отримаєте в результаті. Дуже важливо не обмежувати збір даних людським мисленням, а надавати максимум розрізненої інформації, оскільки машина може побачити користь і взаємозв'язки там, де людина їх не помітить.

2.1.2 Ознаки

До ознак належать ті характеристики, від яких безпосередньо залежить вихідний результат [51].

Найвідомішим прикладом набору даних, складеного за ознаками являється датасет «Wine Quality Data Set» [52], який був створений університетом Каліфорнії у 2009 році. Ці дані є результатами хімічного аналізу вин, вирощених в одному регіоні Італії, але отриманих із трьох різних сортів. Аналіз визначив 13 компонентів, знайдених у кожному з трьох типів вин. Саме виходячи з компонентів вина можна визначити його клас.

2.1.3 Алгоритм машинного навчання

Алгоритмом називають сукупність послідовних операцій для вирішення певної задачі [51]. Іншими словами – метод вирішення. Під кожну конкретну задачу можна підібрати окремий витончений алгоритм. Саме від обраного методу безпосередньо залежить швидкість і точність результату обробки вхідних даних.

2.2 Види алгоритмів машинного навчання

Вирішувати завдання можна за допомогою різних методів. Усі методи діляться на три основних групи [53]:

- навчання з учителем;
- навчання без учителя;
- навчання з підкріпленням.

2.2.1 Навчання з учителем

У цьому випадку машина має «наставника» – учителя, який говорить їй, як правильно вчинити. Вчитель заздалегідь відмічає всі потрібні дані, щоб машина навчалася на конкретних прикладах. Так він показує, що на цій світлині автомобіль, на іншій – велосипед. У термінах машинного навчання вчитель – це саме втручання людини в процес обробки інформації. З учителем машина вчиться набагато краще й швидше, тому для вирішення практичних завдань такі алгоритми використовують частіше. До алгоритмів навчання з учителем належать такі типи задач, як регресія і класифікація.

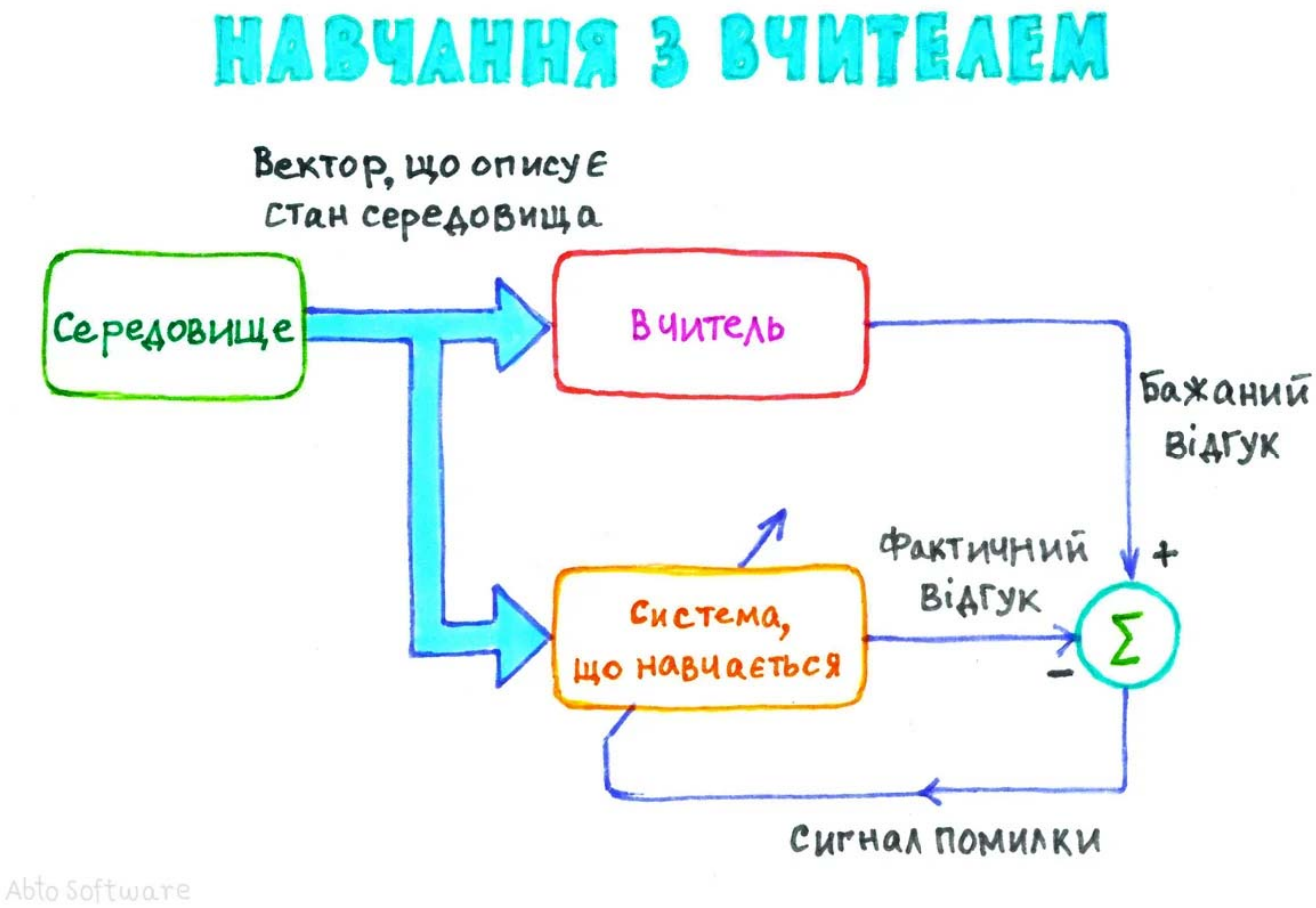


Рисунок 2.2 – Схема підходів машинного навчання з учителем

2.2.2 Навчання без учителя

У навчанні без учителя машині просто дають набір даних і просять розібратися, що тут і до чого. Дані в такому разі не розмічені, і машина сама намагається знайти закономірності. На практиці такі алгоритми використовують рідше, зазвичай як методи аналізу та підготовки даних, а не як основний алгоритм, що вирішує конкретні завдання за допомогою цих даних. У реальності добре розмічені дані – це велика рідкість, тому для їх розмітки зазвичай використовують або спеціальні сервіси, у яких реальні люди з країн з дешевою робочою силою (зазвичай Індія або Китай) за мінімальну плату вручну класифікують дані, або спеціальні алгоритми для розмітки.

Великі корпорації, які мають великий потік користувачів, можуть використовувати своїх клієнтів для розмітки даних. Наприклад, Google і його anticaptcha (рисунок 2.3): знаходячи автобуси на фото, ви не тільки підтверджуєте, що ви не бот, а й навчаєте нейронну мережу тому, який вигляд має автобус. У випадках, коли розмітка даних неможлива, вдаються до методів навчання без учителя.

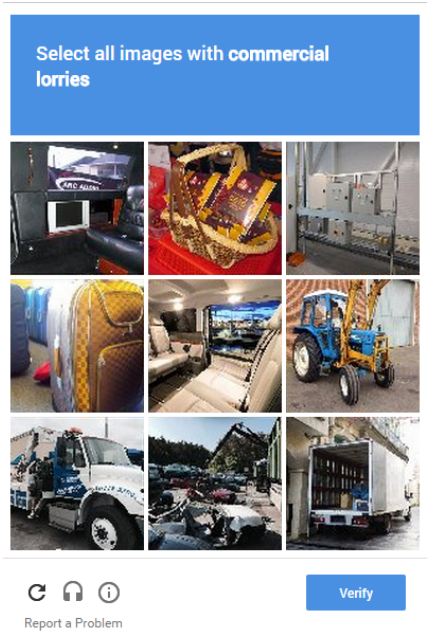


Рисунок 2.3 – Google anticaptcha та розмітка даних

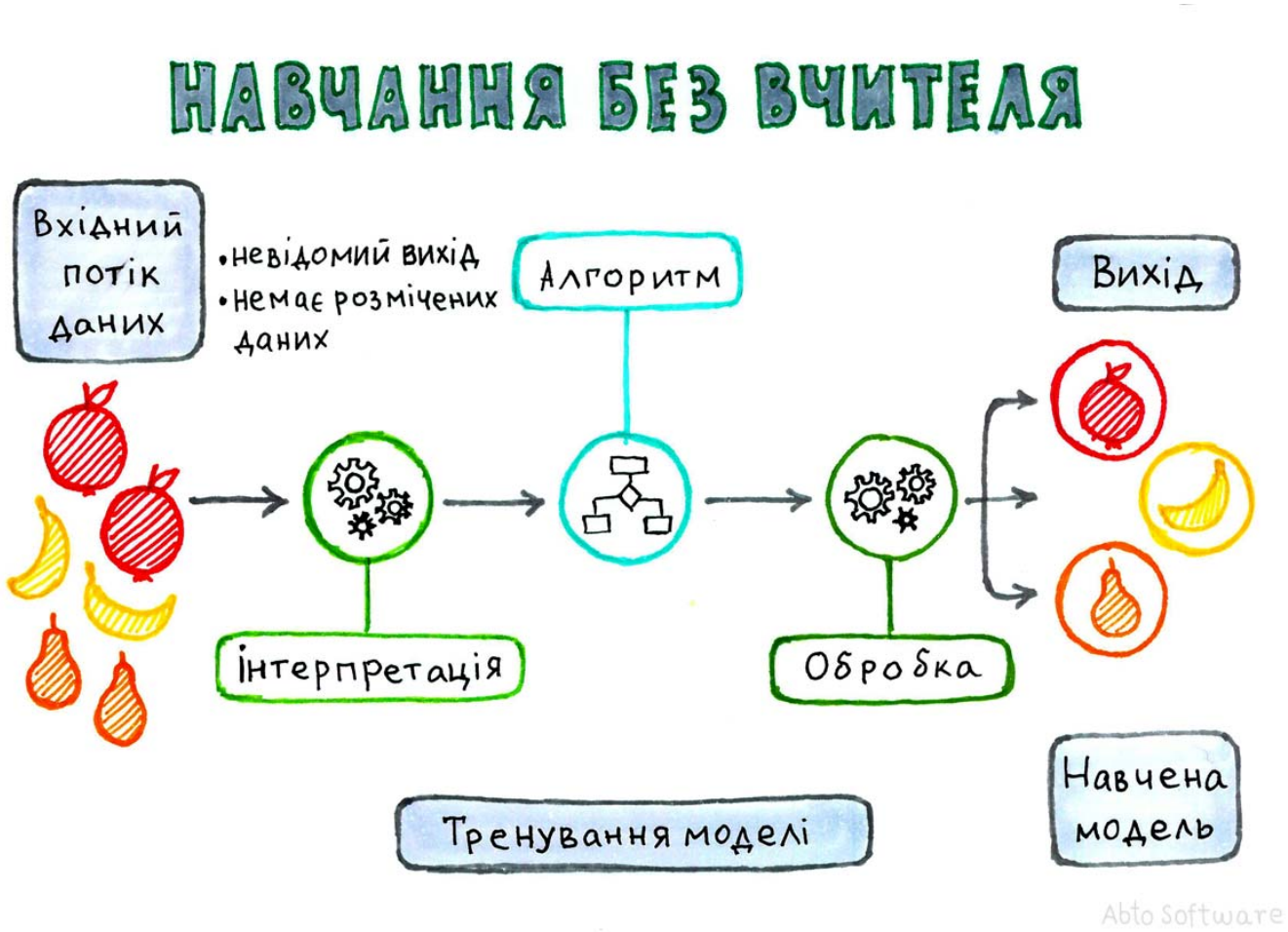


Рисунок 2.4 – Схема підходів машинного навчання без учителя

2.2.3 Навчання з підкріпленням

Навчання з підкріпленням менше схоже на попередні види, бо нагадує швидше той штучний інтелект, яким нас намагаються вразити у фантастичних фільмах. Такі алгоритми використовують не там, де потрібно проаналізувати дані, а там, де потрібно вижити в реальному середовищі. Середовищем може бути що завгодно: як реальний світ, так і симуляція, і навіть комп'ютерна гра. Наприклад, існують роботи, які навчилися грати в Dota2 просто поринувши в середовище, або автопілот Tesla, який у симуляції вчиться не збивати пішоходів.

Алгоритм AlphaGo, створений кілька років тому, який обіграв найкращих гравців в світі в Go створений за допомогою використання підходів навчання з підкріпленням.

Число комбінацій у грі перевищує кількість атомів у Всесвіті, тому всі комбінації машині запам'ятати неможливо. AlphaGo просто обирала найкращий вихід із ситуації, що склалася, і робила це краще за людину. Успіхи машин у комп'ютерних іграх, спорті або технології автопілоту виглядають досить ефектно. Та насправді алгоритмів навчання з підкріпленням не так багато, цей напрям лише починає розвиватися, але саме тому за ними, безсумнівно, майбутнє.

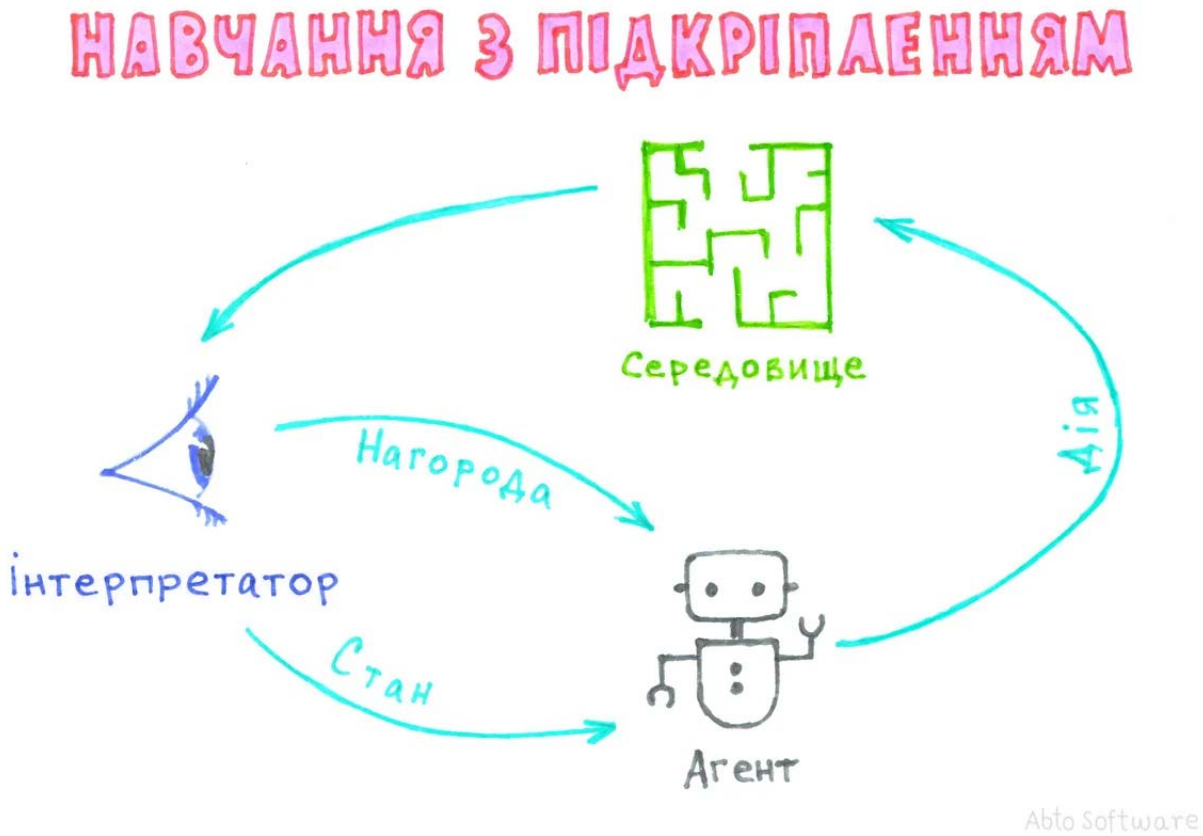


Рисунок 2.5 – Схема підходів машинного навчання з підкріпленням

2.3 Катастрофічне забування нейронних мереж

Катастрофічне забування – це тенденція штучної нейронної мережі повністю і різко забувати раніше вивчену інформацію при вивченні нової інформації [54], [55]. Нейронні мережі є важливою частиною підходу до когнітивної науки. Ці мережі використовують комп’ютерне моделювання, щоб спробувати змоделювати поведінку

людини, таку як пам’ять та навчання. Катастрофічне забування – важлива проблема, яку слід враховувати при створенні штучних нейронних моделей пам'яті. Вперше увагу наукової спільноти до цієї проблеми звернули дослідження Макклоскі та Коена (1989) [54], та Раткліфа (1990) [55]. Це радикальний прояв дилеми "чутливість-стабільність" [56] або дилеми "стабільність-пластичність" [57]. Зокрема, ці проблеми стосуються питання можливості створення штучної нейронної мережі, чутливої до нової інформації, але які залишаються стабільними при вивчені нової інформації. Таблиці пошуку та штучні нейронні мережі лежать на протилежних сторонах спектру пластичность-стабільность [58]. Перший залишається повністю стабільним при появі нової інформації, але йому не вистачає можливості узагальнювати, тобто робити висновки щодо загальних принципів, з нових входів. З іншого боку, штучні нейронні мережі, такі як стандартна мережа зворотного розповсюдження, дуже чутливі до нової інформації і можуть узагальнювати на нових шматках вхідної інформації. Моделі зворотного розповсюдження можна вважати хорошими моделями людської пам'яті, оскільки вони відображають людську здатність до узагальнення, але ці мережі часто виявляють меншу стабільність, ніж людська пам'ять. Примітно, що ці мережі зворотного розповсюдження сприйнятливі до катастрофічного забування. Це вважається проблемою при спробі моделювати людську пам'ять, оскільки, на відміну від цих мереж, люди, як правило, не демонструють катастрофічного забування. Таким чином, питання катастрофічного забування повинна бути викоріненою з цих моделей зворотного розповсюдження, щоб підвищити їх схожість до людської пам'яті.

2.4 Підходи до вирішення проблеми катастрофічного забування

2.4.1 Регуляризація мережі

Методи регуляризації базуться на застосовані додаткових обмежень щодо оновлення ваг, завдяки чому нові концепції вивчаються при цьому зберігаючи попередні концепті не торкнутими. Наприклад, Гудфеллоув виявив [59], що підходи з

використанням dropout-методів [60] менше проявлять ефект катастрофічного забування. Даний підхід обмежує параметри мережі, важливі для старих завдань, щоб залишатися близькими до своїх старих значень та одночасно шукаючи рішення нового завдання в районі старого локального мінімума. Наприклад, EWC [61] та його варіанти [62], [63] використовують інформаційну матрицю Фішера для оцінки важливості ваг; MAS [64] використовує градієнти вихідних даних мережі; SI [65] замість цього використовує інтеграл шляху по траєкторії оптимізації. RWalk [62] поєднує EWC [61] та SI [65].

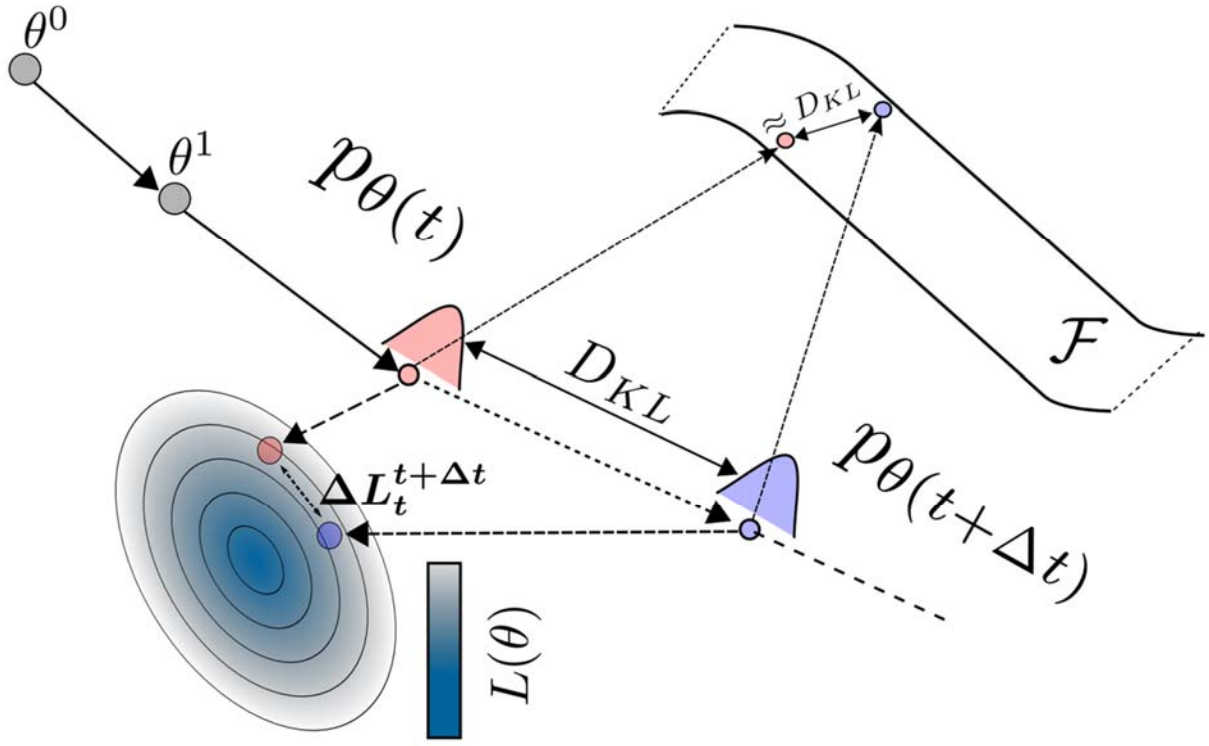


Рисунок 2.6 – Мінімізація дистанції вирішення нової задачі відносно старого локального мінімуму методом RWalk [62]

Інформація про старі і нові задачі не є симетричною під час навчання за цими методами; крім того, мережа може ставати все жорсткішою та жорсткішою в відношені адаптації до нових задач, оскільки з часом вона вивчає дуже багато різних задач. Хойем [66] дотримувався іншого підходу до проблеми катастрофічного забування та запропонував метод навчання без забуття (LwF), який доопрацьовує мережу,

використовуючи зображення нових класів із втратою знань [67], щоб заохотити вихідні ймовірності старих класів для кожного зображення бути близьким до вихідних мережевих виходів. Однак інформаційна асиметрія між старими та новими класами все ще існує. Зображення для нових задач можуть серйозно відрізнятися від розподілу старих даних, що ще більше погіршує інформаційну асиметрію.

2.4.2 Динамічні нейронні мережі

В підходах з використанням динамічні нейронних мереж [68], [69], [70], [71] виділяють частину мережі або унікальний шлях прямого розповсюдження через нейрони для кожного завдання. Під час тестування їм потрібно вказати мітку завдання, щоб перейти в правильний стан мережі.

Приклад динамічного перемикання режиму роботи нейронної мережі за допомогою маскування типу задачі наведено на рисунку 2.7.

2.4.3 Повторення старих задач

В підходах з повторенням попередніх задач [72], [73], [74], [75], [76], [77] минула інформація періодично подається на вхід мережі, щоб зміцнити зв'язок нейронів для уже засвоєних задач. Зазвичай повторення попередніх задач виконується шляхом чергування даних минулих задач з даними поточної задачі [78]. Зберігання минулих даних може бути ресурсозатратним або порушувати деякі практичні обмеження, такі як авторські права або конфіденційність даних. Псевдоповторювальні методи намагаються полегшити цю проблему за допомогою генеративних моделей для генерації псевдо-подібних даних [78], які поєднуються з даними для поточної задачі. Однак для цього підходу потрібно навчити генеративну модель та поступово збільшувати кількість класів для генерації, що є ще складнішою проблемою для вирішення. Існуючі такі методи не дають конкурентних результатів [79], [80].

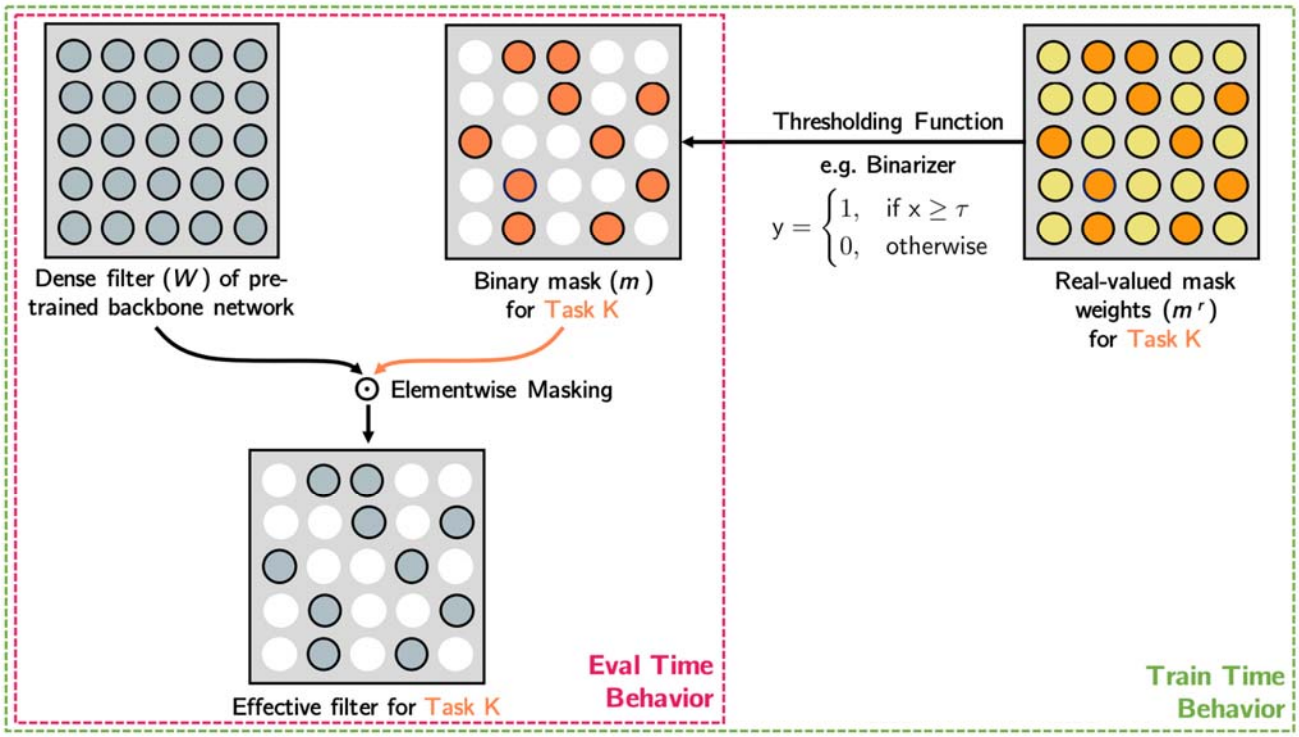


Рисунок 2.7 – Динамічне перемикання режиму роботи нейронної мережі запропонованої в [68]

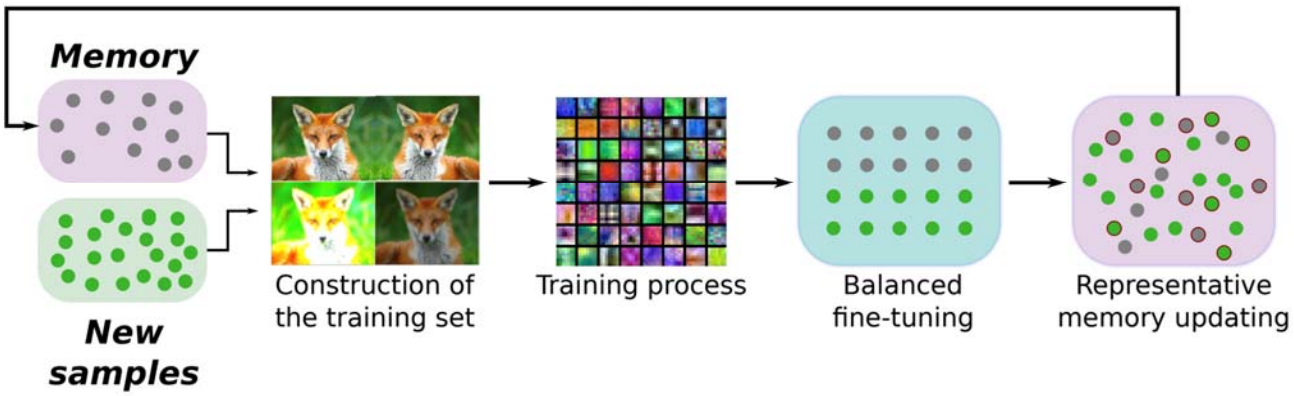


Рисунок 2.8 – Процедура навчання з повторенням старих задач запропонована в [71]

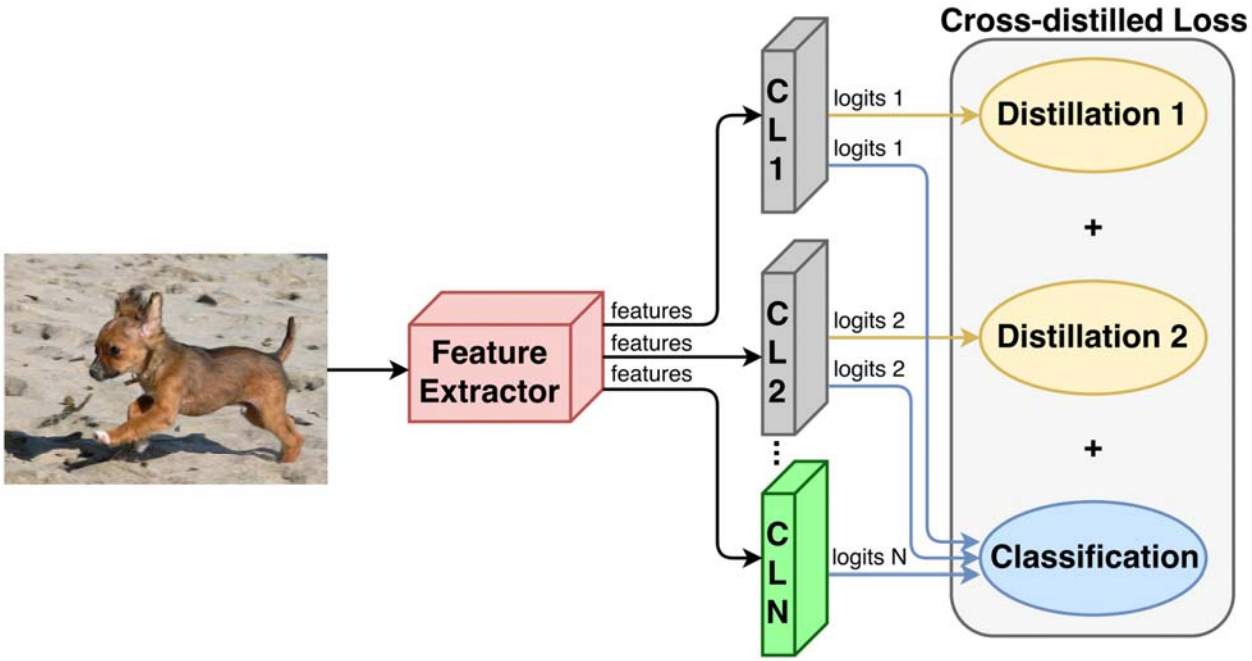


Рисунок 2.9 – Архітектура нейронної мережі для навчання з повторенням старих задач запропонована в [72]

2.5 Підходи інкрементального навчання нейронних мереж

Підходи щодо інкрементального навчання нейронних мереж базуються на наступних спостереженнями та припущеннях [81]:

– задача інкрементального навчання являється задачею навчання моделі на незбалансованому наборі даних, якщо можливо зберігати К точок даних минулих задач [82], [83];

– для інкрементального навчання може бути використана одна модель, яка буде зберігатися протягом усього процесу, а процес інкрементального навчання реалізується через донавчання на новій задачі.

Архітектура інкрементального навчання, яка базується на вище описаних спостереженнях зображена на рисунку 2.10.

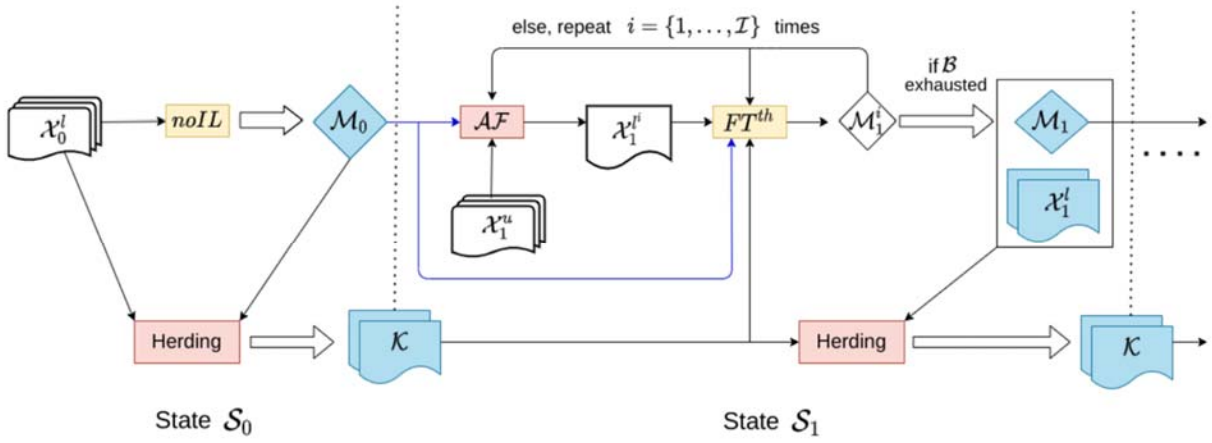


Рисунок 2.10 – Архітектура інкрементального навчання запропонованого в [81]

Запропонований в [81] підхід полягає в використанні однієї моделі машинного навчання M, яка буде ітеративно вивчати нові задачі базуючись на нових вхідних даних. Оскільки, вивчення нових задач погіршує якість моделі на старих задачах, то автори ввели поняття бюджету деградації – межі максимально можливої деградації якості роботи на старих задачах при вивчені нової задачі. Якщо в процесі навчання на новій задачі бюджет деградації буде вичерпаний, то донавчання припиняється, а процес навчання переходить до наступної фаза. Також даний підхід передбачає використання проміжної пам’яті для зберігання K точок даних з попередніх задач на яких можна відкалібрувати модель, після навчання на новій задачі для зменшення ефекту катастрофічного забування. Оскільки, пам'ять для зберігання точок з попередніх задач обмежена, то автори запропонували використати механізм групування даних запропонований в [84] для відбору найкращих кандидатів.

2.6 Підходи постійного донавчання

Підходи до постійного донавчання (online algorithms) дозволяють моделювати проблеми, коли ми маємо постійний потік або безперервний потік даних, які надходять, і ми хотіли б, щоб алгоритм навчався на ньому. Сьогодні багато веб-сайтів та корпорацій

використовують різні версії алгоритмів донавчання в мережі Інтернет, щоб навчитися на потоку даних користувачів, які постійно відвідують їх веб-сайти. Зокрема, якщо у вас є безперервний потік даних, що генерується безперервним потоком користувачів, що приходять на ваш веб-сайт, тоді ви можете використати алгоритми донавчання, щоб дізнатися вподобання користувачів із потоку даних і використовувати це для оптимізації рішення на вашому веб-сайті.

Припустимо, ви запустили службу доставки, тож, знаєте, що користувачі просять вас допомогти перевезти їхній пакунок з місця А до місця В і припустимо, ви керуєте веб-сайтом, куди користувачі неодноразово приходять і кажуть вам, що вони хочуть відправити пакет. Користувачі зазначають звідки і куди вони хочуть його відправити. Ваш веб-сайт пропонує доставити пакет за певну ціну, наприклад за 50 або ж за 20 гривень, і виходячи з ціни, яку ви пропонуєте користувачам, вони вирішують чи скористаються вашою послугою доставки чи ні. Отже, скажімо, що ми хочемо, щоб алгоритм навчання допоміг нам оптимізувати ціну, яку ми хочемо запропонувати своїм користувачам. А конкретно, припустимо, ми придумали якісь функції, які фіксують властивості користувачів (наприклад в якому районі проживає користувач, його вік, тощо). Тоді ми можем промоделювати функцію залежності ймовірності того що користувач скористається нашою послугою доставки за умови що ми виставимо певну ціну, а саму ймовірність оцінимо за допомогою логістичної регресії, нейронної мережі або іншого подібного алгоритму, але давайте розглянемо приклад з використанням логістичної регресії.

Тепер, якщо у вас є веб-сайт, який просто працює безперервно, ось що може зробити алгоритм онлайн-навчання. Коли користувач приходить на наш веб-сайт, ми отримуємо пару {x, y}, що відповідає кожній сесії клієнта на веб-сайт. За властивості {x} будем вважати характери, які описують користувача, та випадкову ціну, яку ми запропонували користувачеві. За мітки {y}, приймемо бінарні мітки – чи скористувався користувач нашою послугою доставки чи ні. Тепер, коли ми має цю пару {x, y}, ми

можемо оновити ваги нашої логістичної регресії використовуючи даний приклад {x, y} за допомогою звичайного алгоритму градієнтного спуску. Якщо наш алгоритм працює на великому веб-сайті, де у вас постійно надходить великий потік користувачів, то такий підхід онлайн-навчання насправді є цілком раціональним, а кожний екземпляр даних {x, y} буде використаний моделлю лише один раз. Звичайно, якби у нас була лише невелика кількість користувачів, то значно раціональніше зберігати кожен екземпляр та формувати набір фіксований набір даних.

Слід також зазначити, що одним із цікавих ефектів такого алгоритму онлайн-навчання є те, що він може адаптуватися до змін вподобань користувачів. І зокрема, якщо з часом через зміни в економіці, можливо, користувачі починають ставати більш чутливими до ціни і менш готовими платити високі ціни. Або якщо вони стають менш чутливими до ціни і готові платити вищі ціни. Або якщо різні речі стають більш важливими для користувачів, якщо у вас з’являються нові типи користувачів, які відвідують ваш веб-сайт. Цей різновид онлайн-алгоритму навчання також може адаптуватися до змін вподобань користувачів. Це відбувається тому що, якщо ваш пул користувачів зміниться, то поступове оновлення ваг логістичної регресії просто повільно адаптуватимуть ваші параметри до того, як виглядає ваш поточний пул користувачів.

2.7 Висновки до розділу

У даному розділі був проведений аналіз існуючих підходів до навчання алгоритмів машинного навчання.

Показано, що сучасні алгоритми машинного навчання на базі глибокого машинного навчання піддатливі ефекту катастрофічного забування, що призводить до повного забування попередньо вивченої інформації при роботі з надвеликими об’ємами даних та великою кількістю категорій.

Для досягнення мети магістерської роботи – удосконалення існуючих підходів розмітки даних – необхідно використати підхід авторозмітки даних за допомого алгоритму машинного навчання, який буде навчатися лише на розміченій частині даних, яка буде поступово збільшуватися за розміром по мірі просування процесу розмітки даних. Оскільки набір даних буде поступово змінювати в часі, то для навчання алгоритму необхідно використати підходи до роботи з динамічними наборами даних. Для даної задачі підходять як методи інкрементального навчання, так і донавчання алгоритмів машинного навчання, оскільки набір категорій, які будуть розмічатися для даного набору даних відомий до старту процесу розмітки.

3 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ АВТОРОЗМІТКИ ДАНИХ ДЛЯ ЗАДАЧ КОМ'ЮТЕРНОГО ЗОРУ

3.1 Класичний підхід до розмітки даних

Згідно з дослідженням Bloomberg [46] класичний підхід до розмітки даних для задач комп'ютерного зору передбачає те, що всі картинки, які наявні в наборі даних, будуть переглянуті людиною-розмітчиком, яка відмітить усі наявні на зображені категорії. Даний процес схематично зображений на рисунку 3.1.

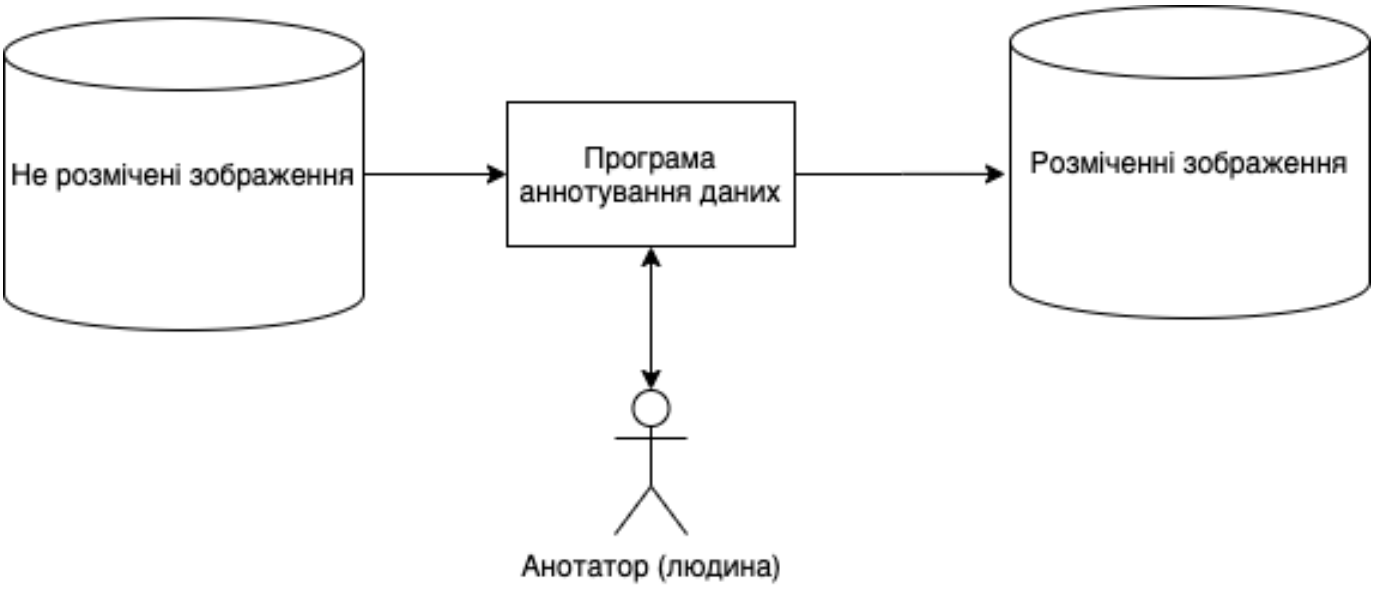


Рисунок 3.1 – Схема класичного підходу до розмітки даних для задач комп'ютерного зору

Даний підхід можна покращити за допомогою використання алгоритмів комп'ютерного зору, які будуть навчатися на уже розмічених даних, і з часом зможуть давати рекомендації щодо наявності того чи іншого об'єкта на зображені. При такому підході людині потрібно буде лише підтвердити припущення алгоритму про наявність об'єкта, а не розмічати його самому, що значно підвищує продуктивність праці. Даний процес схематичного зображений на рисунку 3.2. Підхід з використанням алгоритмів

комп’ютерного зору для являється наразі домінуючим підходом в популярних комерційних утилітах для розмітки даних [50].

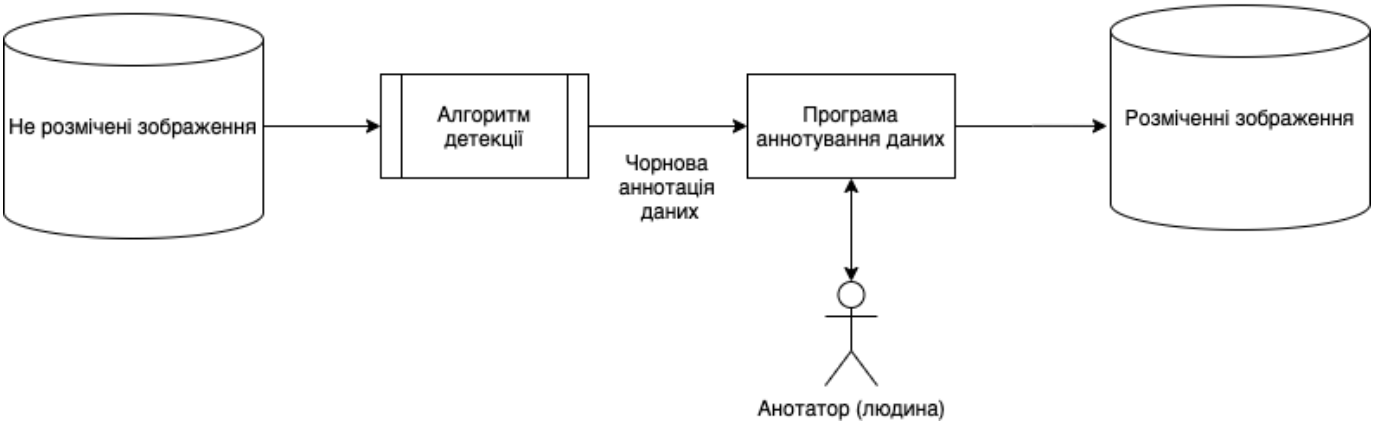


Рисунок 3.2 – Схема підходу до розмітки даних з використанням алгоритмів комп’ютерного зору

Алгоритми комп’ютерного зору, які використовуються в даному процесі називаються алгоритмами авторозмітки даних.

3.2 Механізм пріоритезації розмітки даних

3.2.1 Алгоритм само-діагностики моделей машинного навчання

У роботі [85] був запропоновано модуль само-діагностики алгоритмів детекції об’єктів, основною метою якого було виявлення будь-якої деградації якості роботи алгоритмів, яка може спричинити серйозне зниження продуктивності основних компонентів систем автоматизації і, отже, призвести до небажаної поведінки, як от при використанні алгоритмів детекції об’єктів в програмному забезпечені для автономного керування дорожнім транспортом, яке значною мірою покладається на надійну роботу детектора об’єктів.

Автори запропонували додати до детекторів об'єктів на базі архітектури R-CNN [27], [28], [29], [30] підмережу, яка використовуватиме ознаки $F = \{F_1, F_2, \dots, F_n\}$

отримані глибокою нейронною мережею, для прогнозування ймовірності погіршення продуктивності цієї моделі . Ця підмережа навчається окремо від мережі R-CNN шляхом оптимізації наступного функціоналу:

$$\mathcal{L}(I) = \begin{cases} 1, & mAP_I < \lambda \\ 0, & otherwise \end{cases} \tag{3.1}$$

де $\mathcal{L}(I)$ – функціонал для зображення I, mAP_I – якість роботи алгоритму детекції об'єктів на зображенні I, λ – мінімальний поріг якості роботи алгоритма, який вважається за допустиму якість роботи.

Для покращення роботи цього підходу автори [85] запропонували навчати нейронну підмережу не на сирих ознаках згенерованих глибокою нейронною мережею, а на агрегатах даних ознак, які наведені в формулах 3.2 – 3.4.

$$F_{mean} = \frac{\sum_{x=1}^{H} \sum_{y=1}^{W} f(x,y)}{W * H} \tag{3.2}$$

$$F_{max} = max_{x \in [1,\ H]}\ max_{y \in [1,\ W]}\ f(x,y) \tag{3.3}$$

$$F_{std} = std(f_1) \oplus std(f_2) \ldots \oplus std(f_n) \tag{3.4}$$

де, H – висота вхідних ознак f, W – ширина вхідних ознак f, max – функція максимума вхідних ознак, std – стандартне квадратичне відхилення розподілу вхідних ознак, $\oplus$ - функція конкатенацію ознак.

Остаточний вхід у підмережу створюється шляхом простої конкатенації цих агрегатів, як наведено в формулі 3.5.

$$F_{mean\_\max\_std} = F_{mean} \oplus F_{max} \oplus F_{std} \tag{3.5}$$

Огляд запропонованої в [85] архітектури зображено на рисунку 3.3.

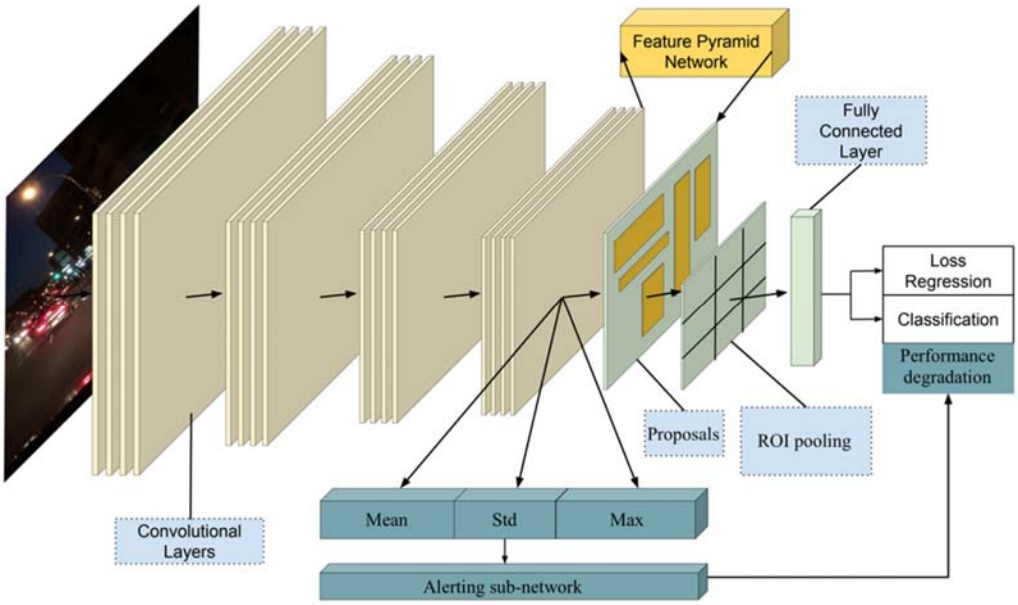


Рисунок 3.3 – Архітектура детектора об’єктів з підмережою для само-діагностики якості роботи запропонованого в [85]

3.2.2 Пріоритезація процесу розмітки даних на основі алгоритму само-діагностики

Базуючись на ідеї само-діагностики моделей детекції об’єктів запропонованої в [85], можна побудувати алгоритм пріоритезації розмітки даних.

Для цього необхідно змінити задачу з бінарної класифікації (хороша/погана якість роботи алгоритму детекції об’єктів) на задачу наближення розподілу очікуваної функції втрат для зображення $P(I) > 0$ до реального розподілу функції втрат $Q(I) > 0$, яка може бути оптимізована за допомогою мінімізації відносної ентропії Кульбака-Лейблера:

$$D_{KL}(P \,||\, Q) = \sum_{I \in X} P(I) * \log(\frac{Q(I)}{P(I)}) \tag{3.6}$$

Тоді, якщо нейронна мережа достатньо точно передбачає розподіл функції втрат $Q(I)$, так що ми можем вважати, що $P(I) \approx Q(I)$, то тоді ми можем відранжувати усі зображення в нашому наборі даних X, таким чим, що очікувана функція втрат на

відібраних зображеннях буде максимальна. Далі необхідно навчати нейронну мережу на відібраних даних, що максимізує якість фінального алгоритму.

3.3 Побудова процесу розмітки даних з використанням алгоритмів авторозмітки даних та механізму пріоретизації

Базуючись на використанні механізму пріоритезації процесу розмітки даних ми можем представити процес розмітки даних наступним чином:

1) визначається список зображення з найбільшою складністю для алгоритму детекції зображень за допомогою механізму пріоритезації;

2) виконується авторозмітка відібраних даних за допомогою алгоритму детекції об’єктів;

3) розмітчик аналізує розмітку отриману від алгоритму авторозмітки; зберігає ті анотації, які вірні; видаляє хибні анотації; виконує дорозмітку даних;

4) інформація про правки внесені розмітчиком відправляється на алгоритм детекції об’єктів для донавчання і мінімізації таких помилок в майбутньому;

5) інформація про реальну складність обраних зображень передається від алгоритму детекції об’єктів до механізму пріоритезації процесу розмітки даних.

Варто замітити, що крок 5 в даному алгоритму являється критичним, оскільки модель детекції об’єктів постійно покращується за рахунок зворотного зв’язку, що отримується на кроці 4, а отже ті зображення, які були раніше складними могли стати простими для алгоритма детекції об’єктів і навпаки. Саме тому нам потрібно постійно адаптуватися механізм пріоритезації процесу розмітки даних, щоб він залишався актуальним.

Схематично процес розмітки даних з використання алгоритму авторозмітки та механізмом пріоритезації представлений на рисунку 3.4.

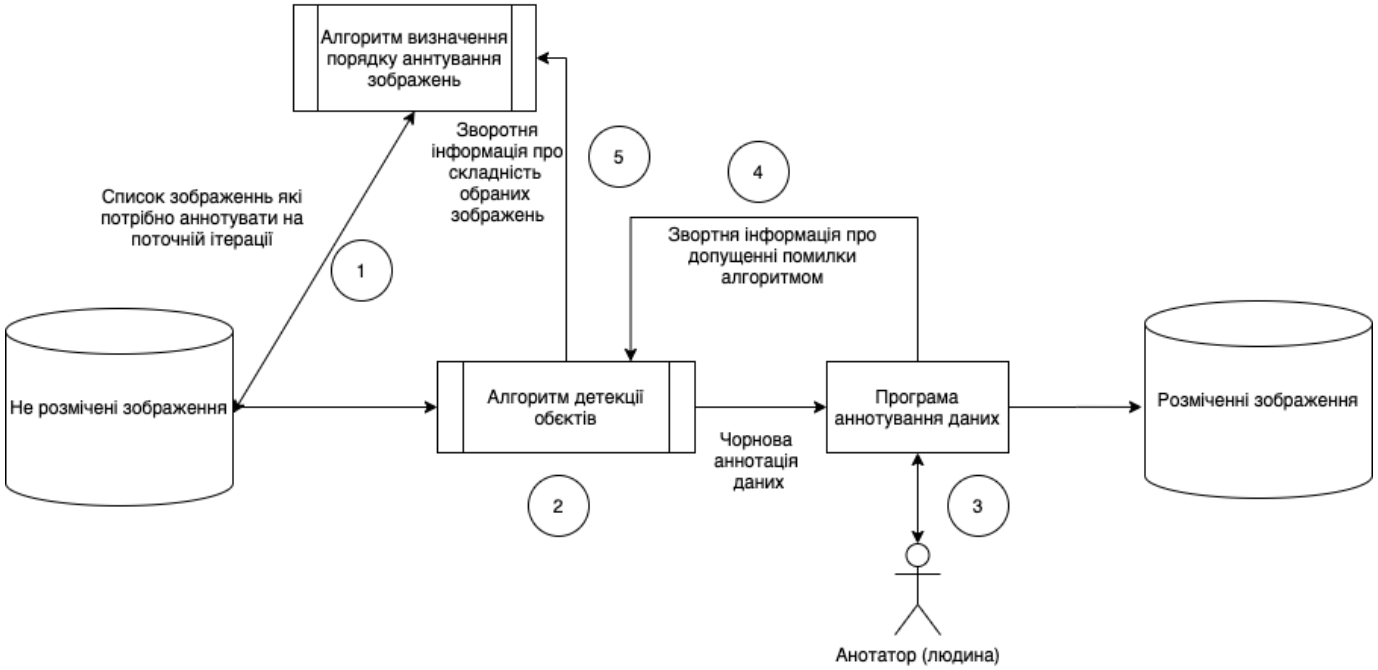


Рисунок 3.4 – Схема підходу до розмітки даних з використанням алгоритмів авторозмітки та механізмом пріоритезації процесу розмітки

3.4 Висновки до розділу

В даному розділі проаналізовано та узагальнено процес розмітки даних з використанням підходу, який передбачає авторозмітку даних.

Також розглянутий механізм само-діагностики моделей детекції об’єктів, який базується на використанні агрегатів ознак, які згенерованих глибокою згортковою нейромережою.

Запропоновано покращення до механізму само-діагностики шляхом заміни функціоналу оптимізації з задачі погана/хороша якість алгоритму детекції на задачу прогнозування функції втрат в процесі оптимізації нейромережі, яка буде оптимізуватися за рахунок мінімізації відносної ентропії Кульбака-Лейблера між реальним та прогнозованим розподілами функції втрат. За рахунок цього вдалося отримати новий механізм пріоретизації, який дозволяє впорядкувати усі зображення в наборі даних від найбільш складніших до найбільш простих.

Механізм пріоретизації дозволяє змінити порядок розмітки даних таким чином, що в першу чергу алгоритм буде навчатися на тих даних, які він вважає найскладнішими, а отже процес навчання буде відбуватися швидше в порівнянні з використанням випадкових зображень з того ж набору даних.

Удосконалено алгоритм авторозмітки надвеликих об’ємів даних для задач детекції об'єктів за рахунок використання запропонованого механізму пріоритезації послідовності розмітки даних та адаптації підходів постійного донавчання алгоритмів детекції, що дозволяє збільшити швидкість розмітки даних та використати даний алгоритм для реалізації розмітки надвеликих даних для задач детекції як типових, так і нетипових об’єктів.

Розроблено алгоритмічне забезпечення для розмітки надвеликих об’ємів даних для задачі детекції об’єктів методами комп’ютерного зору.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до програмного забезпечення

Було показано, що функціонал по авторозмітки даних являється актуальною задачею в сучасних платформах платформах для підтримки задач штучного інтелекту для пришвидшення процесу розробки та впровадженню алгоритмів штучного інтелекту в нові сфери життєдіяльності людини. Однак, якісна реалізація авторозмітки даних або відсутня або наявна лише для типових задач. Тому доцільною є розробка програмного забезпечення для авторозмітки зображень з організацією порядку розмітки даних на основі механізму пріоритезації, запропонованого у розділі 3, у вигляді веб-застосунку.

Враховуючи запропоновані у розділах 1 та 2 підходи до організації розмітки даних, вирішення задачі донавчання алгоритмів машинного навчання та ґрунтуючись на наявних недоліках в сучасних платформах по підтримці задач штучного інтелекту, були сформовані функціональні та нефункціональні вимоги до реалізації програмного забезпечення. До нефункціональних вимог входять:

– розроблене програмне забезпечення представляє собою веб-застосунок;

– розроблене програмне забезпечення працює на базі удосконаленого методу з використанням механізму пріоритезації.

До функціональних вимог входять:

– можливість зміни порядку розмітки зображень;

– можливість авторозмітки даних;

– можливість розмітки зображення.

На рисунку 4.1 зображена діаграма використання, що відповідає функціональним вимогам:

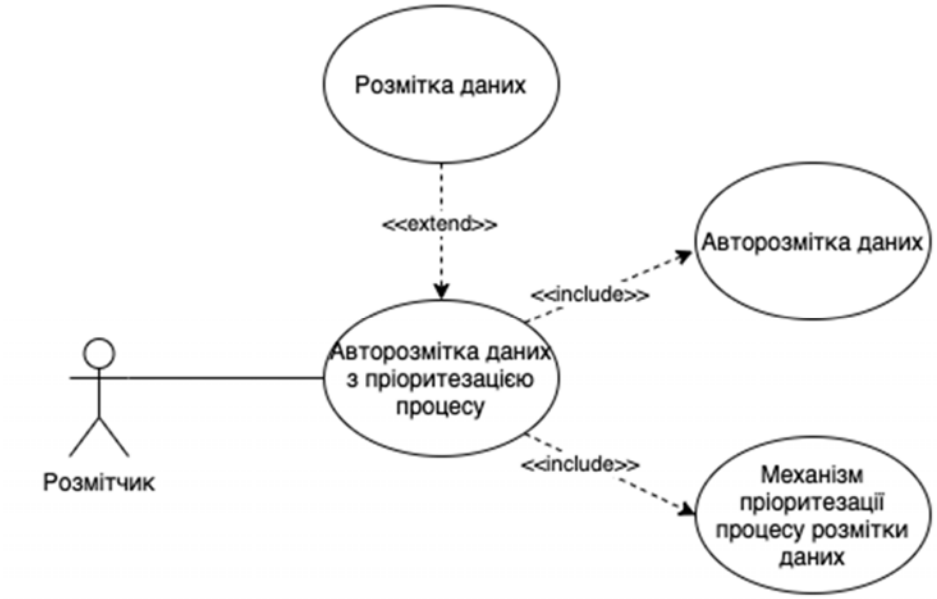


Рисунок 4.1 – Діаграма використання

Слід зазначити, що важливою вимогою для прискорення роботи розробленого застосування є необхідність використання декількох серверів з потужним апаратним забезпеченням, включаючи графічні прискорювачі.

4.2 Опис архітектури програмного забезпечення

Веб-застосунок було розроблено за клієнт-серверною архітектурою, яка наведена в додатку Б та яка складається з наступних компонентів:

- бази даних картинок;
- бази даних розмітки;
- алгоритму авторозмітки даних;
- утиліти розмітки;
- черг розмітки даних.

В якості бази даних для картинок використано сховище AWS S3, яке забезпечує високу надійність зберігання та швидкий доступ до даних при низькій вартості.

В якості бази даних для розмітки була використана база даних Postgres.

Алгоритм авторозмітки даних базується на бібліотеці mmdetection [86], в якій реалізовано усі сучасні підходи до детекції об’єктів.

Рисунок 4.2 – Логотип бібліотеки mmdetection

В якості утиліти розмітки даних була використана утиліта Computer Vision Annotation Tool (CVAT), яка була розроблена командою Intel та викладена в публічний доступ як частина Intel OpenVINO toolkit [49].

Рисунок 4.3 – Логотип Intel OpenVINO toolkit

Для поєднання всіх компонентів архітектури були використані черги на основі сервіса AWS SQS.

4.3 Опис бібліотеки засобів навчання детекторі об’єктів

Розроблена бібліотека у програмному забезпеченні представляє собою набір функцій, що описані у таблиці 4.1:

Таблиця 4.1 – Опис функцій бібліотеки

Назва функції	Вхідні параметри	Призначення
set_random_seed	seed: int, deterministic: bool	Фіксує початковий стан генераторів випадкових чисел
train_detector	model: torch.nn.Module, dataset: torch.utils.data.Dataset, cfg: dict, distributed: bool, validate: bool, timestamp: int, meta: dict	Виконує навчання алгоритму детекції об’єктів на заданому наборі даних
single_gpu_test	model: torch.nn.Module, data_loader: torch.utils.data.DataLoader, show: bool, out_dir: str, show_score_thr: float	Виконує запуск алгоритму детекції об’єктів на заданому наборі даних з використанням одного графічного прискорювача
multi_gpu_test	model: torch.nn.Module, data_loader: torch.utils.data.DataLoader, tmpdir: str, gpu_collect: str	Виконує запуск алгоритму детекції об’єктів на заданому наборі даних з паралелізацією процесу тестування на декількох графічних прискорювачах
collect_results_gpu	result_part: iterable size: int	Виконує reduce-операцію по агрегації результатів запуску алгоритму детекції об’єктів на декількох графічних прискорювачах

Продовження таблиці 4.1

Назва функції	Вхідні параметри	Призначення
init_detector	config: dict, checkpoint: str, device: str, cfg_options: dict	Виконує завантаження та підготовку алгоритму детекції об’єктів до роботи на заданому графічному прискорювачу
inference_detector	model: torch.nn.Module, imgs: np.ndarray	Виконує запуск алгоритму детекції об’єктів на заданих зображеннях
show_result_pyplot	model: torch.nn.Module, img: np.ndarray, result: tuple, score_thr: float, title: str, wait_time: int	Візуалізує результат роботи алгоритму детекції об’єктів на заданому зобреження з виведення в окремому вікні

4.4 Функціональність програмного забезпечення

Розроблений веб-застосунок представляє собою односторінковий додаток, на єдиній сторінці якого розташований весь функціонал сервісу, що робить його простим у використанні.

До функціоналу веб-застосунку передбачає:

– завантаження зображення для розмітки;

– розмітку обраних об’єктів;

– наближення зображення для більшої точної розмітки;

– збереження результатів.

На рисунках 4.4-4.8 представлені екрани додатку при використанні кожного з пунктів функціоналу відповідно:

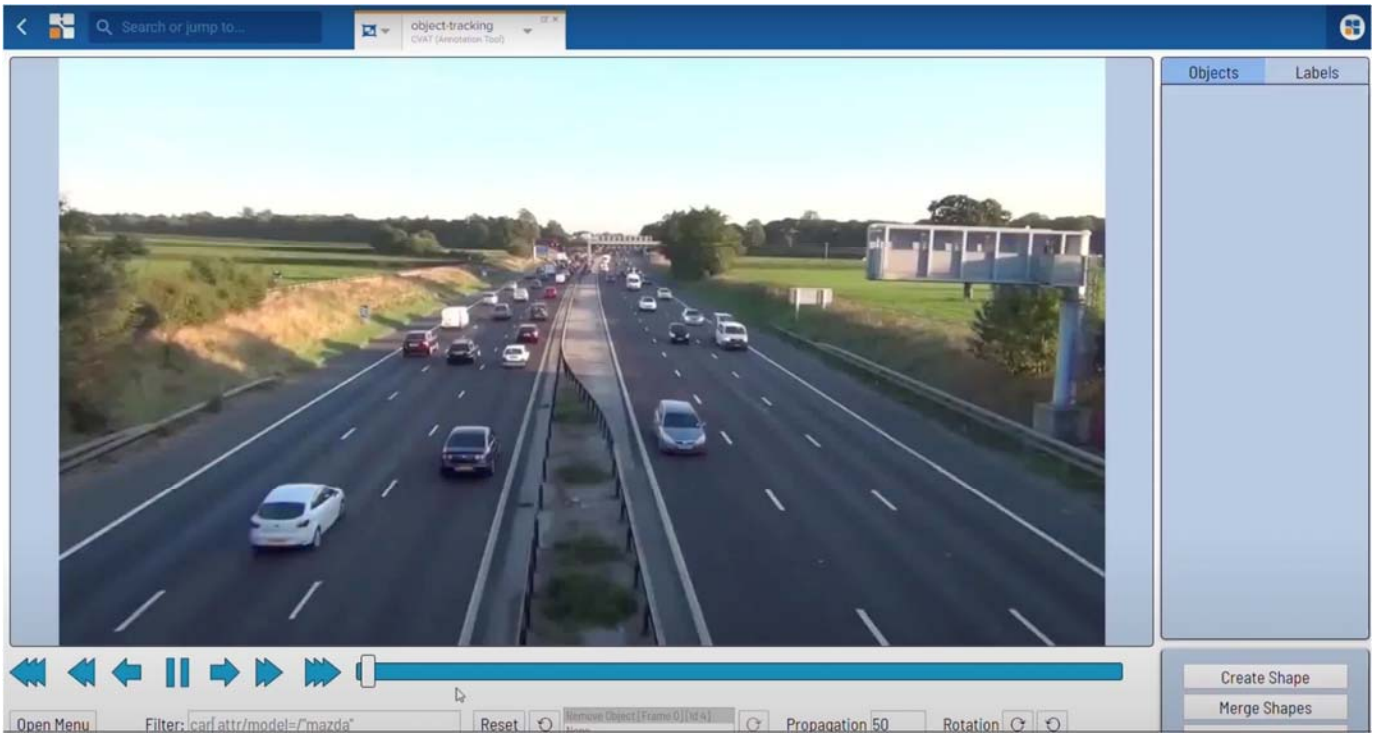


Рисунок 4.4 – Завантаження зображення для розмітки

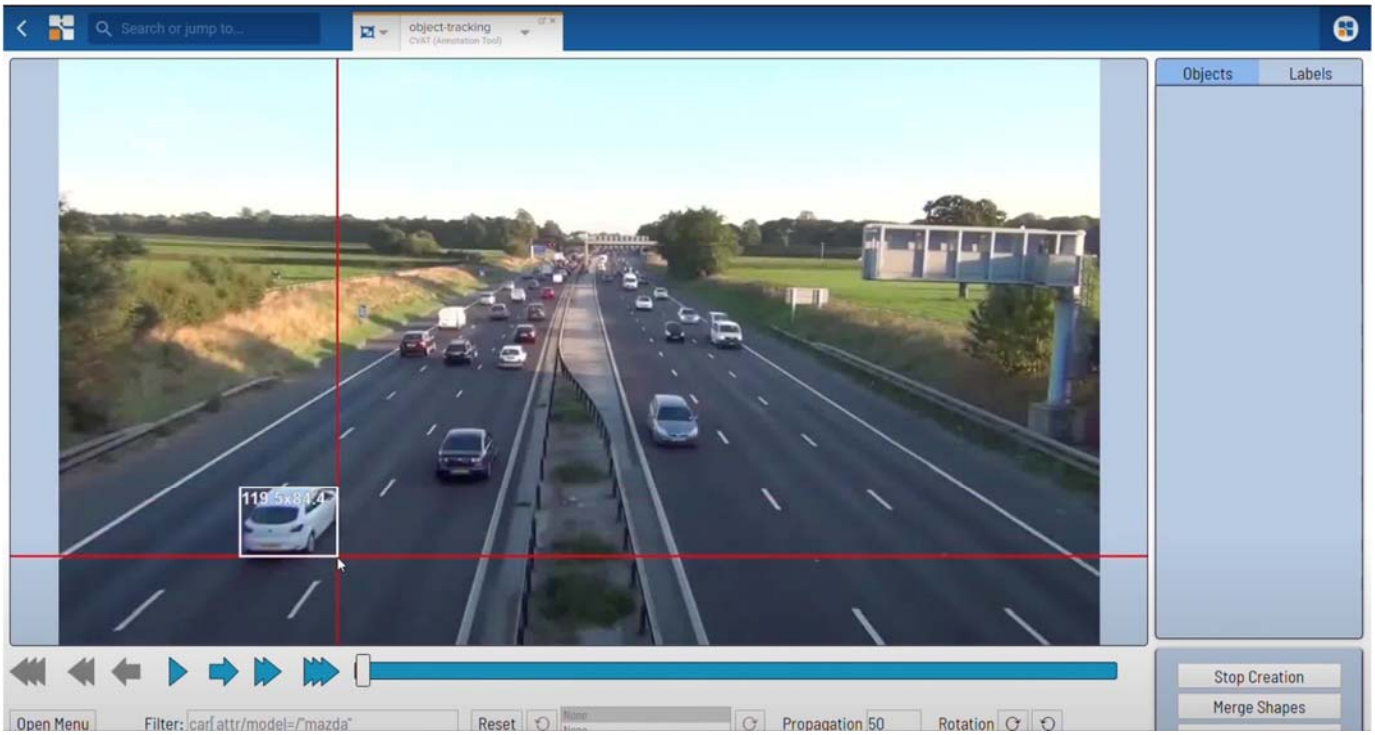


Рисунок 4.5 – Розмітка обраного об’єкту

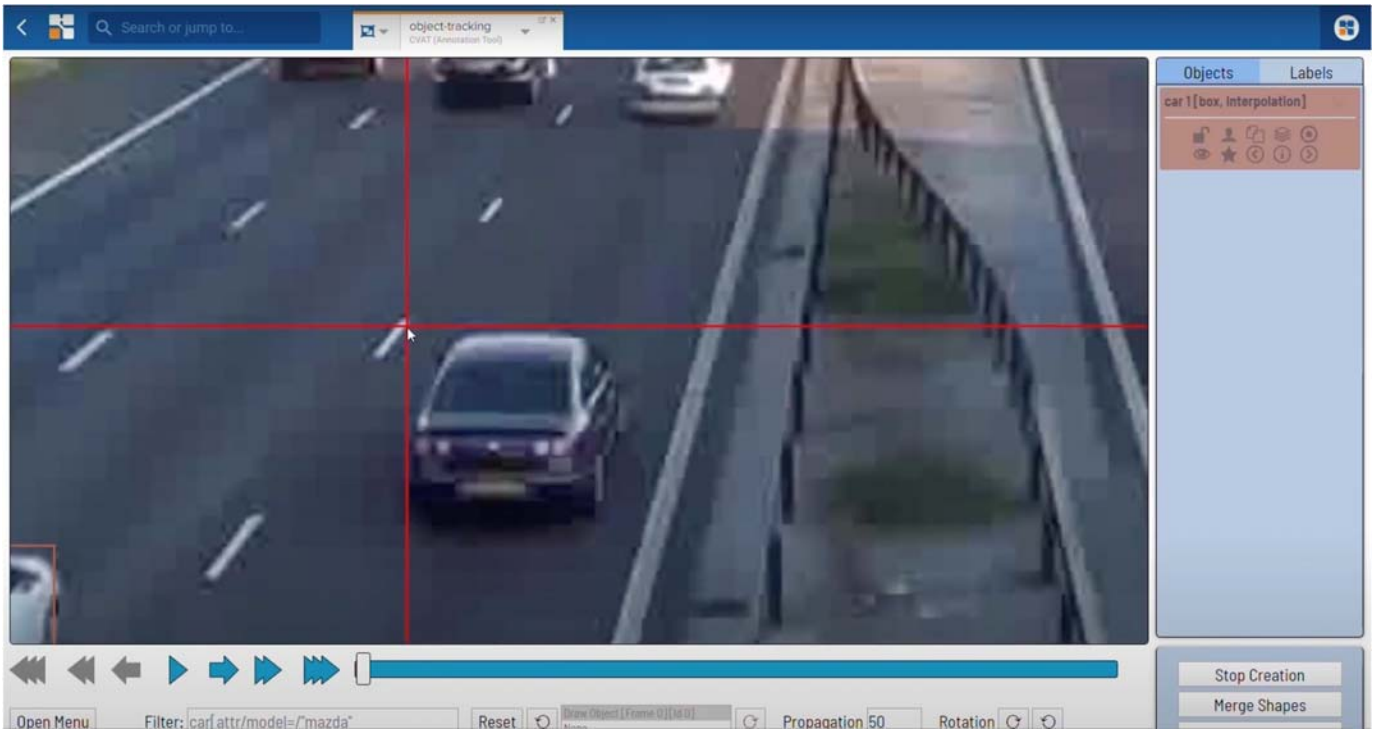


Рисунок 4.6 – Наближення зображення для збільшення точності розмітки

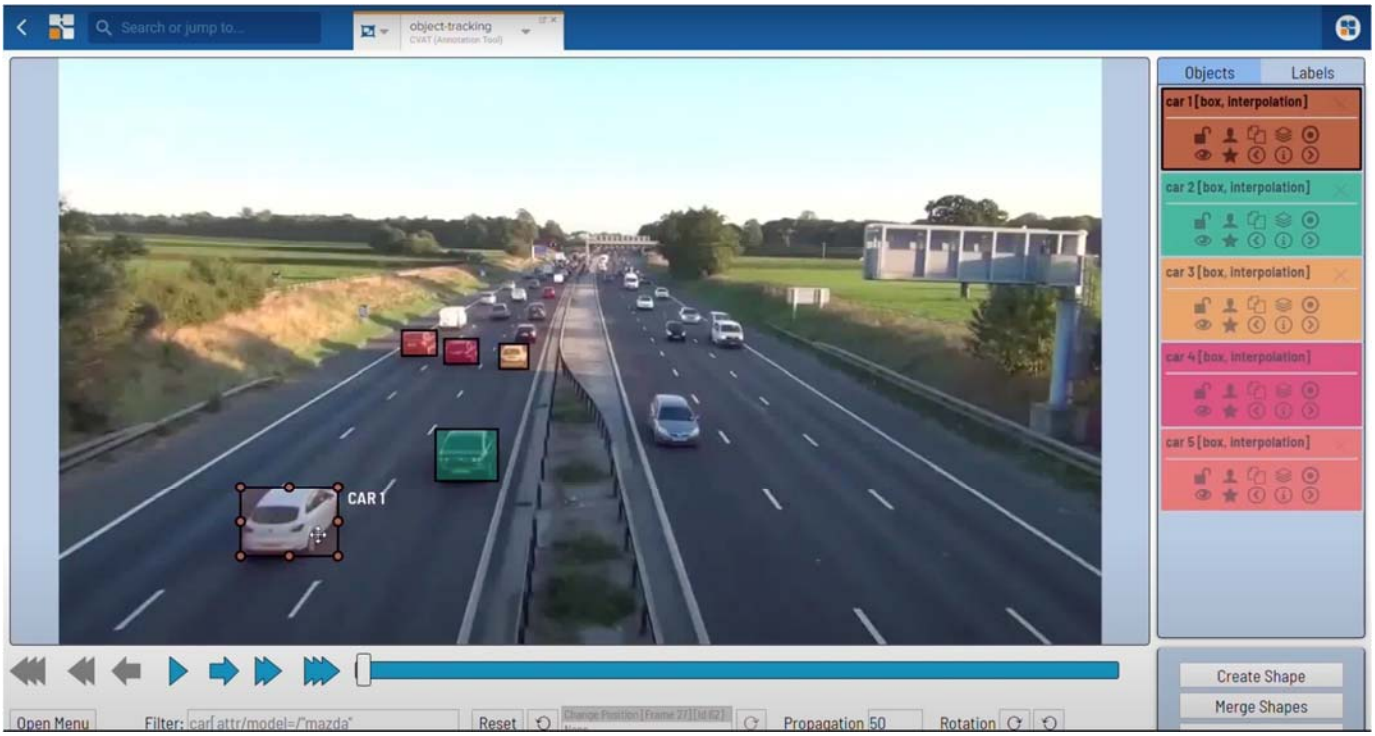


Рисунок 4.7 – Збереження результатів розмітки

4.5 Висновки до розділу

У даному розділі представлено програмну реалізацію запропонованого підходу по пришвидшенню процесу розмітки надвеликих об’ємів даних для задач детекції об’єктів, яке відповідає сформованим вимогам, а саме: програмним забезпеченням, яке працює на базі удосконаленого методу з використанням механізму пріоритезації, описаного у розділі 3 даної роботи, а також надає можливість розмітки зображення.

Розроблене алгоритмічне програмне забезпечення, яке реалізоване на мові програмування Python з використанням бібліотек mmdetection та CVAT. Програмне забезпечення містить набір функцій, що підтримує реалізацію запропонованого алгоритму авторозмітки даних з використання механізму пріоритезації даних для зміни порядку розмітки даних. Ресурсоємні операції обробки зображень та донавчання алгоритму детекції об’єктів розроблений застосунок виконує на віддалених серверах з використанням графічних прискорювачів. Також, розроблене програмне забезпечення підтримує режим тестування, що надає можливість проаналізувати ефективність запропонованого підходу.

Практичне значення розробленого програмного забезпечення для авторозмітки даних в рамках програмної платформи підтримки вирішення задач штучного полягає в підвищені швидкістю процесу розмітки на надвеликих об’ємах, отже зменшення собівартості розробки програмного забезпечення з використання алгоритмів штучного інтелекту.

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

5.1 Опис набору даних

Для постановки експериментів був використаний набір даних Objects 365 [87], який зібраний компанією Megvii на основі зображень завантажених з сайту Flickr.

Рисунок 5.1 – Приклади зображень з набору даних Objects365

Розвернутий аналіз та порівняння набору даних Objects 365 [87] з іншими публічними наборами даних наведено у таблиці 5.1.

Таблиця 5.1 – Порівняльний аналіз наборі даних для задачі детекції об’єктів

Набір даних	Зображень	Об’єктів	Категорій	Об’єктів на зображення
Pascal VOC [21]	11.5k	27k	20	2.4
MS COCO [22]	123k	896k	80	7.3
Objects365 [78]	1800k	29000k	365	14.8

Отже, набір даних являється на сьогоднішній день найбільшим набором даних для задачі детекції об’єктів.

5.2 Опис моделі детекції об’єктів

Для проведення всіх експериментів був використаний алгоритм детеції об’єктів Cascade R-CNN [29], який побудований на базі mmdetection [86] та має наступні характеристики:

– базовий розмір вхідного зображення – 1280x1280;

– кількість стадій детекції об’єктів – 3 [29];

– базова глибока нейронна мережа – ResNeXt-101-64x16d [88],

– предтренерова мережа на 10 мільярдах картинок з набору даних Instagram hashtags [89];

– генератор ознак – FPN [28];

– генератор регіонів інтересу – GA-RPN [90]

– оптимізатор – LookAhead [91].

5.3 Постановка експериментів

Для постановки експериментів була зібрана група з 9 людей-розмітчиків, які займалися розміткою 20 випадкових зображень для унеможливлення заповнювання зображень та механізації процесу розмітки. Для простоти експериментів, розмітчикам було необхідно розмітити лише 6 найчастіших категорій: людина, автомобіль, автобус, літак, собака, кіт.

Для кожного із учасників експерименту було замірено час розмітки усього набору 20 зображень. Результатом кожного експерименту являється медіанне значення часу розмітки усіма учасниками задля мінімізації впливу випадкових чинників.

Порівняння усіх експерименті виконано з контрольним експериментом, який являє собою експеримент з розміткою 20 випадкових зображень без попередньої обробки жодним алгоритмом авторозмітки даних.

5.4 Експерименти з послідовність розмітки даних

Як було зазначено в розділах 2.6 та 2.7, є два підходи до інкрементального навчання моделей машинного навчання:

– інкрементальний по даних: навчати модель одночасно знаходити усі категорії;

– інкрементальний по категоріях: поступово добавляти категорії в процесі навчання.

В даному експериментів ми перевіримо обидва підходи.

Для експерименту з інкрементальним навчання по даних була навчена модель на 9000 та 18000 картинках на усіх 6 категоріях.

Для експерименту з інкрементальним навчання по категоріях було навчено 6 моделей для кожної категорії окремо на 1500 та 6000 картинках відповідно.

Результати експерименту наведено в таблиці 5.2.

Таблиця 5.2 – Результати експерименту з послідовністю розмітки

Експеримент	Медіанний час розмітки 20 зображень (хв)
Контрольний експеримент	17
Інкремент по даних (сумарно 9000 картинок)	15
Інкремент по даних (сумарно 36000 картинок)	14
Інкремент по категоріях (сумарно 9000 картинок)	17
Інкремент по категоріях (сумарно 36000 картинок)	16

Як видно з таблиці 5.2 найкращий результат отриманий при використанні інкрементального підходу по даних, це пов’язано з наявністю додаткової інформації в алгоритму детекції об’єктів про інші категорії, що дозволяє допускати менше помилок в авторозмітці, наприклад розмічати котів як людей, оскільки модель, яка навчена лише на людях, не має жодної інформації про концепцію котів. Отже, для подальших експериментів буде використаний підхід з використанням інкрементального навчання по даних.

5.5 Експерименти з механізмом пріоритезації розмітки

В даному розділі ми порівняємо вплив механізму пріоритезації та вплив різнорідності даних на швидкість розмітки. Аналогічно до розіділу 5.4 було проведено по 2 експерименту з різною кількість даних (9000 та 36000 картинок відповідно) та порівняно з контрольним експериментом та результатами інкрементального навчання по даних з розділу 5.4. Результати експериментів з механізмом пріоритезації розмітки наведено в таблиці 5.3.

Таблиця 5.3 – Результати експериментів з механізмом пріоретизації розмітки

Експеримент	Медіанний час розмітки 20 зображень (хв)
Контрольний експеримент	17
Інкремент по даних (сумарно 9000 картинок)	15
Інкремент по даних (сумарно 36000 картинок)	14
З механізмом пріоретизації (сумарно 9000 картинок)	14
З механізмом пріоретизації (сумарно 36000 картинок)	10

Отже, як видно з результатів експериментів, механізм пріоритезації пришвидшує процес розмітки за рахунок швидшого навчання алгоритму детекції об’єктів, який уже на ранніх етапах навчання дає аналогічні результати до тих, що можна отримати в системах автоматичної розмітки даних, навчених на більших об’ємах даних.

5.6 Оцінка впливу запропонованого підходу на вартість розмітки даних

Для оцінки впливу запропонованого підходу на вартість розмітки набору даних скористаємося наступними припущення:

– час на розмітку даних лінійно росте з лінійним збільшенням кількості зображень в наборі даних;

– час на розмітку даних лінійно росте з лінійним збільшенням кількості категорій розмітки;

– час на розмітку даних лінійно падає з лінійним ростом якості алгоритму детекції об’єктів;

– середня вартість розмітки 100 різних категорій на одному зображені для задачі детекції об’єктів складає $2 [39];

– середній час на розмітку одного зображення без використання алгоритмів авторозмітки складає 1хв;

– розмітка набору даних буде тривати рівно 18 місяців.

Тоді, можна змоделювати процес розмітки даних та оцінити вартість розмітки такого набору даних як Objects365 (2 мільйони картинок, 365 категорій) при використанні різних підходів до розмітки даних. Змодельована вартість процесу розмітки Objects365 за 18 місяців наведено на рисунку 5.2.

Для розмітки набору даних аналогічного до Object365 класичними підходами необхідно 18 місяців, а вартість сягне 14.6 млн. доларів США. На противагу, для розмітки цього ж набору з використанням підходів до авторозмітки даних потрібно уже 15 місяців та бюджет в 12.3 млн. доларів США, тобто на 16% менше ресурсів, а затрати на автоматизацію процесу розмітки почнуть окупатися через 9 місяців.

Якщо ж використовувати запропоновай підхід, то уже необхідно 13 місяців та 10.9 млн. доларів для розмітки даного набору даних, або на 26% ефективнішим в порівняні з класичним підходом та на 11% в порівняні з іншими підходами до авторозмітки даних. Також варто замітити, що запропонований підхід виходить на окупність за 6 місяців.

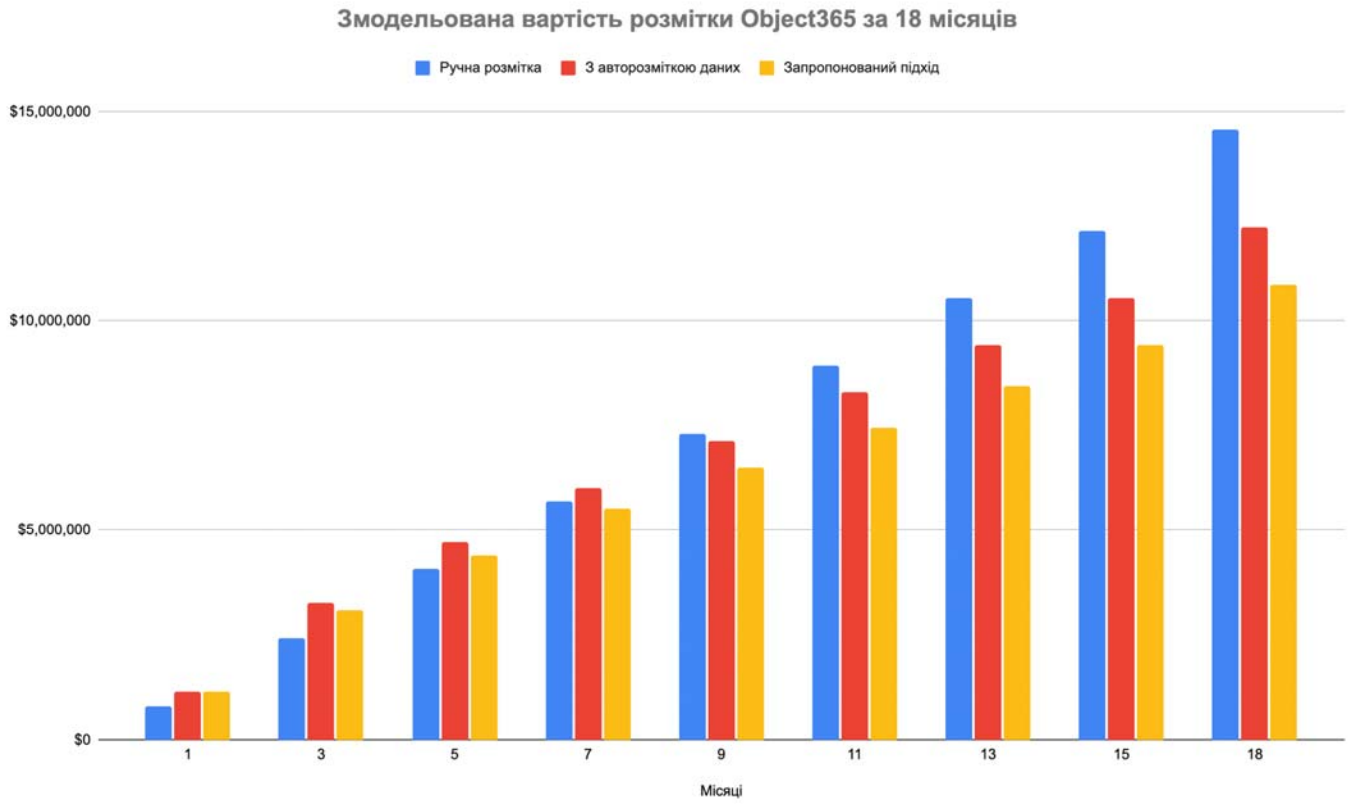


Рисунок 5.2 – Змодельована вартість розмітки набору даних Object365 з використанням різних підходів до розмітки даних

5.7 Висновки до розділу

В цьому розділі наведено результати експериментальних досліджень підходу до авторозмітки даних з використанням запропонованого механізму пріоритезації для зміни порядку розмітки даних з метою пришвидшення процесу розмітки даних.

Було проведено порівняльний аналіз результатів роботи існуючих програмних засобів та розробленого програмне забезпечення. Розроблене програмне забезпечення авторозмітки надвеликих об’ємів даних з використанням пріоритезації процесу розмітки показав на 26% кращу швидкість даного процесу в порівнянні з класичним підходом до розмітки даних. Виявлено, що в порівнянні з підходами авторозмітки, що не використовують механізм пріоритезації в процесі розмітки даних, запропонований

підхід показує на 11% кращу швидкість розмітки даних, а період виходу складає лише 6 місяців в порівнянні з існуючими аналогами.

Отже, запропоноване програмне забезпечення показує кращу ефективність в термінах швидкості розмітки даних в порівняні з існуючими програмними засобами.

ВИСНОВКИ

У даній магістерській дисертації було удосконалено підхід до авторозмітки надвеликих об’ємів даних для детекції об’єктів шляхом адаптації алгоритмів постійного донавчання алгоритму авторозмітки та розроблено механізм пріоритезації послідовності розмітки даних, що дозволяє розміти весь наявний набір даних в коротші терміни.

В процесі виконання роботи був проведений аналіз існуючих платформ підтримки задач штучного інтелекту. Результати розгорнутого аналізу показали, що дані платформи концентруються на процесі створення та підтримки алгоритмів штучного інтелекту, хоча дослідження в сфері створення програмного забезпечення на основі підходів штучного інтелекту показали, що 80% часу на таких проектах витрачається на роботу з даних, зокрема на розмітку даних. Виявлено, що сучасні платформи штучного інтелекту або не мають вбудованих засобів автоматизації процесу розмітки або мають підтримку лише типових задач.

З метою усунення цього недоліку розроблено удосконалений підхід до авторозмітки даних шляхом адаптації алгоритмів постійного донавчання алгоритму авторозмітки та розроблено механізм пріоритезації послідовності розмітки даних, що дозволяє розміти весь наявний набір даних в коротші терміни.

Запропонований механізм пріоретизації порядку розмітки даних базується на використанні агрегатів ознак згенерованих глибокою згортковою нейромережею, які використовуються для оцінки складності зображень. Ці оцінки використовуються для того, щоб відібрати найскладніші зображення, оскільки навчання на них буде відбуватися швидше в порівнянні з використанням випадкових зображень з того ж набору даних.

Розроблене алгоритмічне програмне забезпечення, яке реалізоване на мові програмування Python з використанням бібліотек mmdetection та CVAT.

З використанням розробленого алгоритмічного програмного забезпечення був проведений порівняльний аналіз швидкості розмітки даних при використанні класичних підходів до розмітки даних та запропонованого підходу. Проведена експериментальна перевірка показала, що запропонований підхід має на 26% вищу ефективність в порівнянні з класичними підходами до розмітки даних для задач детекції об’єктів та на 11% в порівнянні з іншими підходами до авторозмітки даних.

Наукова новизна отриманих результатів полягає в тому, що запропоновано механізм пріоритезації процесу розмітки даних як удосконалення механізму само-діагностики моделей детекції об’єктів, який встановлює порядок розмітки даних шляхом першочергового відбору для навчання алгоритму детекції об’єктів найскладніших зображень, вибір яких здійснюється на основі агрегатів ознак, згенерованих глибокою згортковою нейромережею для оцінки складності зображень, що пришвидшує процес навчання в порівнянні з використанням випадкових зображень з того ж набору даних.

Удосконалено алгоритм авторозмітки надвеликих об’ємів даних для задач детекції об'єктів за рахунок використання запропонованого механізму пріоритезації послідовності розмітки даних та адаптації підходів постійного донавчання алгоритмів детекції, що дозволяє збільшити швидкість розмітки даних та використати даний алгоритм для реалізації розмітки надвеликих даних для задач детекції як типових, так і нетипових об’єктів.

Таким чином, усі поставлені завдання магістерської роботи виконані, а поставлена мета – досягнута. Розроблене алгоритмічне програмне забезпечення є зручним інструментом для пришвидшення розмітки надвеликих об’ємів даних для задач детекції об’єктів.

СПИСОК ЛІТЕРАТУРИ

1) “AI Platforms”, [Електронний ресурс]. – 2021. – Режим доступу: https://www.predictiveanalyticstoday.com/category/reviews/ai-platforms

2) “Awash in “AI Platforms”, [Електронний ресурс]. – 2021. – Режим доступу: https://c3.ai/what-is-enterprise-ai/awash-in-ai-platforms

3) “Microsoft Azure AI Platform”, [Електронний ресурс]. – 2021. – Режим доступу: https://azure.microsoft.com/en-us/overview/ai-platform

4) “AI Platform”, [Електронний ресурс]. – 2021. – Режим доступу: https://cloud.google.com/ai-platform

5) “Amazon SageMaker”, [Електронний ресурс]. – 2021. – Режим доступу: https://aws.amazon.com/sagemaker

6) “IBM Watson”, [Електронний ресурс]. – 2021. – Режим доступу: https://www.ibm.com/watson

7) “Lionbridge AI”, [Електронний ресурс]. – 2020. – Режим доступу: https://lionbridge.ai/

8) “TensorFlow”, [Електронний ресурс]. – 2021. – Режим доступу: https://www.tensorflow.org

9) “PyTorch”, [Електронний ресурс]. – 2021. – Режим доступу: https://pytorch.org

10) “Scikit-learn”, [Електронний ресурс]. – 2021. – Режим доступу: https://scikit-learn.org/stable

11) “Machine learning attacks against Asirra Captcha”, [Електронний ресурс]. – 2008. – Режим доступу: https://crypto.stanford.edu/~pgolle/papers/dogcat.pdf

12) Dogs versus cats Kaggle competition on Asirra dataset [Електронний ресурс]. – 2019. – Режим доступу: https://www.kaggle.com/c/dogs-vs-cats

13) P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, p. 1627, 2010

14) K. K. Sung and T. Poggio, “Example-based learning for view-based human face detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 1, pp. 39–51, 2002

15) C. Wojek, P. Dollar, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 4, p. 743, 2012

16) H. Kobatake and Y. Yoshinaga, “Detection of spicules on mammogram based on skeleton analysis.” IEEE Trans. Med. Imag., vol. 15, no. 3, pp. 235–245, 1996

17) Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in ACM MM, 2014

18) Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in NIPS, 2012

19) Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in CVPR, 2017

20) Z. Yang and R. Nevatia, “A multi-scale cascade fully convolutional network face detector,” in ICPR, 2016

21) C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in ICCV, 2015

22) X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in CVPR, 2017

23) Dundar, J. Jin, B. Martini, and E. Culurciello, “Embedded streaming deep neural networks accelerator with applications,” IEEE Trans. Neural Netw. & Learning Syst., vol. 28, no. 7, pp. 1572–1583, 2017

24) R. J. Cintra, S. Duffner, C. Garcia, and A. Leite, “Low-complexity approximate convolutional neural networks,” IEEE Trans. Neural Netw. & Learning Syst., vol. PP, no. 99, pp. 1–12, 2018

25) Ziniu Wu, Rong Zhu, Andreas Pfadler, Yuxing Han, Jiangneng Li, Zhengping Qian, Kai Zeng, Jingren Zhou. “FSPN: A New Class of Probabilistic Graphical Model”. [Електронний ресурс]. – 2020. – https://arxiv.org/abs/2011.09020

26) S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri,“Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans. Neural Netw. & Learning Syst., vol. PP, no. 99, pp. 1–15, 2017

27) Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. In NIPS, pages 379–387, 2016

28) Tsung-Yi Lin, Piotr Dollar, Ross B. Girshick, Kaiming He, Bharath ´Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In CVPR, pages 936–944, 2017

29) Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In IEEE Conference on Computer Vision and Pattern Recognition, 2018

30) Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross B. Girshick. ´Mask R-CNN. In ICCV, pages 2980–2988, 2017

31) Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, 88(2):303–338, 2010

32) Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In ECCV, pages 740-755, 2014

33) R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in CVPR, 2014

34) R. Girshick, “Fast r-cnn,” in ICCV, 2015

35) J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in CVPR, 2016

36) S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015, pp. 91–99

37) Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. We don’t need no bounding-boxes: Training object class detectors using only human verification. In CVPR, pages 854–863, 2016

38) Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In ICCV, pages 4940–4949, 2017

39) “How Much Do Image Annotation Services Cost?”, [Електронний ресурс]. – 2019. – Режим доступу: https://lionbridge.ai/articles/how-much-do-image-annotation-services-cost/

40) “Что такое аннотация изображений: 5 основных видов”, [Електронний ресурс]. – 2020. – Режим доступу: https://senior.ua/articles/chto-takoe-annotaciya-izobrazheniy-5-osnovnyh-vidov

41) “Lionbridge AI”, [Електронний ресурс]. – 2020. – Режим доступу: https://lionbridge.ai/

42) “Pricing | Data Labelling Service”, [Електронний ресурс]. – 2020. – Режим доступу: https://cloud.google.com/ai-platform/data-labeling/pricing

43) “Как работает разметка данных”, [Електронний ресурс]. – 2020. – Режим доступу: https://data.korusconsulting.ru/press-center/blog/kak-rabotaet-razmetka-dannykh/

44) “Data Engineering, Preparation, and Labeling for AI 2019”, [Електронний ресурс]. – 2019. – Режим доступу: https://www.cognilytica.com/2019/03/06/report-data-engineering-preparation-and-labeling-for-ai-2019/

45) “Разметка данных”, [Електронний ресурс]. – 2019. – Режим доступу: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8

F:%D0%A0%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_(data_labeling)

46) “Best Practices for Managing Data Annotation Projects”, [Електронний ресурс]. – 2020. – Режим доступу: https://arxiv.org/pdf/2009.11654.pdf

47) “5 Approaches to Data Labeling for Machine Learning Projects”, [Електронний ресурс]. – 2020. – Режим доступу: https://lionbridge.ai/articles/5-approaches-to-data-labeling-for-machine-learning-projects/

48) “Як навчити машину бачити: розмітка зображень для штучного інтелекту (AI)”, [Електронний ресурс]. – 2020. – Режим доступу: https://evergreens.com.ua/ua/articles/image-annotation.html

49) “Computer Vision Annotation Tool: универсальный подход к разметке данных”, [Електронний ресурс]. – 2020. – Режим доступу: https://habr.com/ru/company/intel/blog/433772/

50) “Darwin”, [Електронний ресурс]. – 2021. – Режим доступу: https://www.v7labs.com/darwin

51) “Штучний інтелект, машинне навчання та нейронні мережі: в чому різниця і для чого їх використовують”, [Електронний ресурс]. – 2019. – Режим доступу: https://evergreens.com.ua/ua/articles/machine-learning-overview.html

52) “Wine Quality Data Set” [Електронний ресурс]. – 2009. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

53) “Вступ до машинного навчання” [Електронний ресурс]. – 2009. – Режим доступу: http://specials.kunsht.com.ua/machinelearning2

54) McCloskey, M. & Cohen, N. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (ed.) The Psychology of Learning and Motivation,24, 109-164

55) Ratcliff, R. (1990) Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. Psychological Review,97, 285-308

56) Hebb, D.O. (1949). “Organization of Behaviour”. New York: Wiley

57) Caroebterm G., & Grossberg, S. (1987) ART 2: Self-organization of stable category recognition codes for analog input patterns. “Applied Optics, 26”, 4919-4930

58) French, R. M. (1997) Pseudo-recurrent connectionist networks: an approach to the 'sensitivity-stability' dilemma. Connection Science, 9(4), 353–379

59) I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In International Conference on Learning Representations (ICLR), 2014

60) K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017

61) J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, page 201611835, 2017

62) A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In The European Conference on Computer Vision (ECCV), September 2018

63) J. Schwarz, J. Luketina, W. M. Czarnecki, A. GrabskaBarwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. Proceedings of the 35th International Conference on Machine Learning, 2018

64) R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In Proceedings of the European Conference on Computer Vision (ECCV), pages 139–154, 2018

65) F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 3987–3995. JMLR. org, 2017

66) Z. Li and D. Hoiem. Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017

67) G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In NIPS Deep Learning and Representation Learning Workshop, 2014

68) A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Proceedings of the European Conference on Computer Vision (ECCV), pages 67–82, 2018

69) A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7765–7773, 2018

70) J. Serra, D. Sur ` ´ıs, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Proceedings of the International Conference on Machine Learning (ICML), pages 4548–4557, 2018

71) J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In Proceedings of the International Conference on Learning Representations (ICLR), 2018

72) F. M. Castro, M. J. Marın-Jimenez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In The European Conference on Computer Vision (ECCV), September 2018

73) A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In Proceedings of the International Conference on Learning Representations (ICLR), 2019

74) K. Javed and F. Shafait. Revisiting distillation and incremental classifier learning. Asian Conference on Computer Vision (ACCV), 2018

75) D. Lopez-Paz et al. Gradient episodic memory for continual learning. In Advances in Neural Information Processing Systems, pages 6467–6476, 2017

76) C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In Proceedings of the International Conference on Learning Representations (ICLR), 2018

77) S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017

78) A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science, 7(2):123–146, 1995

79) R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. In International Conference on Learning Representations, 2018

80) H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, pages 2990–2999, 2017

81) Eden Belouadah, Adrian Popescu, Umang Aggarwal, Léo Saci. Active Class Incremental Learning for Imbalanced Datasets. Arxiv preprint, 2020

82) Belouadah, E., Popescu, A.: Il2m: Class incremental learning with dual memory. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 583–592 (2019)

83) Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 831–839 (2019)

84) Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Conference on Computer Vision and Pattern Recognition. CVPR (2017)

85) Quazi Marufur Rahman, Niko Sünderhauf, Feras Dayoub. Performance Monitoring of Object Detection During Deployment. arXiv:2009.08650, 2020

86) Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang,

Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin. “MMDetection: Open MMLab Detection Toolbox and Benchmark”, [Електронний ресурс]. – 2019. – Режим доступу: https://arxiv.org/abs/1906.07155

87) Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Jing Li, Xiangyu Zhang, Jian Sun. “Objects365: A Large-Scale, High-Quality Dataset for Object Detection”, [Електронний ресурс]. – 2019. – Режим доступу: https://www.objects365.org/overview.html

88) Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. “Aggregated Residual Transformations for Deep Neural Networks”, [Електронний ресурс]. – 2016. – Режим доступу: https://arxiv.org/abs/1611.05431

89) I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, Dhruv Mahajan. “Billion-scale semi-supervised learning for image classification”, [Електронний ресурс]. – 2019. – https://arxiv.org/abs/1905.00546

90) Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, Dahua Lin. “Region Proposal by Guided Anchoring”, [Електронний ресурс]. – 2019. – https://arxiv.org/abs/1901.03278

91) Michael R. Zhang, James Lucas, Geoffrey Hinton, Jimmy Ba. “Lookahead Optimizer: k steps forward, 1 step back”, [Електронний ресурс]. – 2019. – https://arxiv.org/abs/1907.08610

ДОДАТОК А.
ПРОГРАМНИЙ КОД

Dockerfile

```
ARG PYTORCH="1.6.0"
ARG CUDA="10.1"
ARG CUDNN="7"

FROM pytorch/pytorch:${PYTORCH}-cuda${CUDA}-cudnn${CUDNN}-devel

ENV TORCH_CUDA_ARCH_LIST="6.0 6.1 7.0+PTX"
ENV TORCH_NVCC_FLAGS="-Xfatbin -compress-all"
ENV CMAKE_PREFIX_PATH="$(dirname $(which conda))/../"

RUN apt-get update && apt-get install -y ffmpeg libsm6 libxext6 git ninja-build libglib2.0-0 libsm6 libxrender-dev libxext6 \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Install MMCV
RUN pip install mmcv-full==latest+torch1.6.0+cu101 -f https://openmmlab.oss-accelerate.aliyuncs.com/mmcv/dist/index.html

# Install MMDetection
RUN conda clean --all
RUN git clone https://github.com/open-mmlab/mmdetection.git /mmdetection
WORKDIR /mmdetection
```

```
ENV FORCE_CUDA="1"
RUN pip install -r requirements/build.txt
RUN pip install --no-cache-dir -e .
```

dist_test.

```
#!/usr/bin/env bash

CONFIG=$1
CHECKPOINT=$2
GPUS=$3
PORT=${PORT:-29500}

PYTHONPATH="$(dirname $0)/..":$PYTHONPATH \
python -m torch.distributed.launch --nproc_per_node=$GPUS --master_port=$PORT \
    $(dirname "$0")/test.py $CONFIG $CHECKPOINT --launcher pytorch ${@:4}
```

dist_train.sh

```
#!/usr/bin/env bash

CONFIG=$1
GPUS=$2
PORT=${PORT:-29500}

PYTHONPATH="$(dirname $0)/..":$PYTHONPATH \
python -m torch.distributed.launch --nproc_per_node=$GPUS --master_port=$PORT \
    $(dirname "$0")/train.py $CONFIG --launcher pytorch ${@:3}
```

train.py

```
import random
import warnings

import numpy as np
import torch
from mmcv.parallel import MMDataParallel, MMDistributedDataParallel
from mmcv.runner import (HOOKS, DistSamplerSeedHook, EpochBasedRunner,
                         Fp16OptimizerHook, OptimizerHook, build_optimizer,
                         build_runner)
from mmcv.utils import build_from_cfg

from mmdet.core import DistEvalHook, EvalHook
from mmdet.datasets import (build_dataloader, build_dataset,
                            replace_ImageToTensor)
from mmdet.utils import get_root_logger


def set_random_seed(seed, deterministic=False):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    if deterministic:
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = False
```

```
def train_detector(model,
                   dataset,
                   cfg,
                   distributed=False,
                   validate=False,
                   timestamp=None,
                   meta=None):
    logger = get_root_logger(cfg.log_level)

    # prepare data loaders
    dataset = dataset if isinstance(dataset, (list, tuple)) else [dataset]
    if 'imgs_per_gpu' in cfg.data:
        logger.warning('"imgs_per_gpu" is deprecated in MMDet V2.0. '
                       'Please use "samples_per_gpu" instead')
        if 'samples_per_gpu' in cfg.data:
            logger.warning(
                f'Got "imgs_per_gpu"={cfg.data.imgs_per_gpu} and '
                f'"samples_per_gpu"={cfg.data.samples_per_gpu}, "imgs_per_gpu"'
                f'={cfg.data.imgs_per_gpu} is used in this experiments')
        else:
            logger.warning(
                'Automatically set "samples_per_gpu"="imgs_per_gpu"='
                f'{cfg.data.imgs_per_gpu} in this experiments')
        cfg.data.samples_per_gpu = cfg.data.imgs_per_gpu

    data_loaders = [
```

```
    build_dataloader(
        ds,
        cfg.data.samples_per_gpu,
        cfg.data.workers_per_gpu,
        # cfg.gpus will be ignored if distributed
        len(cfg.gpu_ids),
        dist=distributed,
        seed=cfg.seed) for ds in dataset
]

# put model on gpus
if distributed:
    find_unused_parameters = cfg.get('find_unused_parameters', False)
    # Sets the `find_unused_parameters` parameter in
    # torch.nn.parallel.DistributedDataParallel
    model = MMDistributedDataParallel(
        model.cuda(),
        device_ids=[torch.cuda.current_device()],
        broadcast_buffers=False,
        find_unused_parameters=find_unused_parameters)
else:
    model = MMDataParallel(
        model.cuda(cfg.gpu_ids[0]), device_ids=cfg.gpu_ids)

# build runner
optimizer = build_optimizer(model, cfg.optimizer)
```

```
if 'runner' not in cfg:
    cfg.runner = {
        'type': 'EpochBasedRunner',
        'max_epochs': cfg.total_epochs
    }
    warnings.warn(
        'config is now expected to have a `runner` section, '
        'please set `runner` in your config.', UserWarning)
else:
    if 'total_epochs' in cfg:
        assert cfg.total_epochs == cfg.runner.max_epochs

runner = build_runner(
    cfg.runner,
    default_args=dict(
        model=model,
        optimizer=optimizer,
        work_dir=cfg.work_dir,
        logger=logger,
        meta=meta))

# an ugly workaround to make .log and .log.json filenames the same
runner.timestamp = timestamp

# fp16 setting
fp16_cfg = cfg.get('fp16', None)
if fp16_cfg is not None:
```

```
    optimizer_config = Fp16OptimizerHook(
        **cfg.optimizer_config, **fp16_cfg, distributed=distributed)
elif distributed and 'type' not in cfg.optimizer_config:
    optimizer_config = OptimizerHook(**cfg.optimizer_config)
else:
    optimizer_config = cfg.optimizer_config

# register hooks
runner.register_training_hooks(cfg.lr_config, optimizer_config,
                               cfg.checkpoint_config, cfg.log_config,
                               cfg.get('momentum_config', None))
if distributed:
    if isinstance(runner, EpochBasedRunner):
        runner.register_hook(DistSamplerSeedHook())

# register eval hooks
if validate:
    # Support batch_size > 1 in validation
    val_samples_per_gpu = cfg.data.val.pop('samples_per_gpu', 1)
    if val_samples_per_gpu > 1:
        # Replace 'ImageToTensor' to 'DefaultFormatBundle'
        cfg.data.val.pipeline = replace_ImageToTensor(
            cfg.data.val.pipeline)
    val_dataset = build_dataset(cfg.data.val, dict(test_mode=True))
    val_dataloader = build_dataloader(
        val_dataset,
        samples_per_gpu=val_samples_per_gpu,
```

```
        workers_per_gpu=cfg.data.workers_per_gpu,
        dist=distributed,
        shuffle=False)
    eval_cfg = cfg.get('evaluation', {})
    eval_cfg['by_epoch'] = cfg.runner['type'] != 'IterBasedRunner'
    eval_hook = DistEvalHook if distributed else EvalHook
    runner.register_hook(eval_hook(val_dataloader, **eval_cfg))

# user-defined hooks
if cfg.get('custom_hooks', None):
    custom_hooks = cfg.custom_hooks
    assert isinstance(custom_hooks, list), \
        f'custom_hooks expect list type, but got {type(custom_hooks)}'
    for hook_cfg in cfg.custom_hooks:
        assert isinstance(hook_cfg, dict), \
            'Each item in custom_hooks expects dict type, but got ' \
            f'{type(hook_cfg)}'
        hook_cfg = hook_cfg.copy()
        priority = hook_cfg.pop('priority', 'NORMAL')
        hook = build_from_cfg(hook_cfg, HOOKS)
        runner.register_hook(hook, priority=priority)

if cfg.resume_from:
    runner.resume(cfg.resume_from)
elif cfg.load_from:
    runner.load_checkpoint(cfg.load_from)
runner.run(data_loaders, cfg.workflow)
```

test.py

```
import os.path as osp
import pickle
import shutil
import tempfile
import time

import mmcv
import torch
import torch.distributed as dist
from mmcv.image import tensor2imgs
from mmcv.runner import get_dist_info

from mmdet.core import encode_mask_results


def single_gpu_test(model,
                    data_loader,
                    show=False,
                    out_dir=None,
                    show_score_thr=0.3):
    model.eval()
    results = []
    dataset = data_loader.dataset
    prog_bar = mmcv.ProgressBar(len(dataset))
    for i, data in enumerate(data_loader):
```

```
with torch.no_grad():
    result = model(return_loss=False, rescale=True, **data)

batch_size = len(result)
if show or out_dir:
    if batch_size == 1 and isinstance(data['img'][0], torch.Tensor):
        img_tensor = data['img'][0]
    else:
        img_tensor = data['img'][0].data[0]
    img_metas = data['img_metas'][0].data[0]
    imgs = tensor2imgs(img_tensor, **img_metas[0]['img_norm_cfg'])
    assert len(imgs) == len(img_metas)

    for i, (img, img_meta) in enumerate(zip(imgs, img_metas)):
        h, w, _ = img_meta['img_shape']
        img_show = img[:h, :w, :]

        ori_h, ori_w = img_meta['ori_shape'][:-1]
        img_show = mmcv.imresize(img_show, (ori_w, ori_h))

        if out_dir:
            out_file = osp.join(out_dir, img_meta['ori_filename'])
        else:
            out_file = None

        model.module.show_result(
            img_show,
```

```
                result[i],
                show=show,
                out_file=out_file,
                score_thr=show_score_thr)

        # encode mask results
        if isinstance(result[0], tuple):
            result = [(bbox_results, encode_mask_results(mask_results))
                      for bbox_results, mask_results in result]
        results.extend(result)

        for _ in range(batch_size):
            prog_bar.update()
    return results


def multi_gpu_test(model, data_loader, tmpdir=None, gpu_collect=False):
    """Test model with multiple gpus.

    This method tests model with multiple gpus and collects the results
    under two different modes: gpu and cpu modes. By setting 'gpu_collect=True'
    it encodes results to gpu tensors and use gpu communication for results
    collection. On cpu mode it saves the results on different gpus to 'tmpdir'
    and collects them by the rank 0 worker.

    Args:
        model (nn.Module): Model to be tested.
```

```
    data_loader (nn.Dataloader): Pytorch data loader.
    tmpdir (str): Path of directory to save the temporary results from
        different gpus under cpu mode.
    gpu_collect (bool): Option to use either gpu or cpu to collect results.

Returns:
    list: The prediction results.
"""
model.eval()
results = []
dataset = data_loader.dataset
rank, world_size = get_dist_info()
if rank == 0:
    prog_bar = mmcv.ProgressBar(len(dataset))
time.sleep(2)  # This line can prevent deadlock problem in some cases.
for i, data in enumerate(data_loader):
    with torch.no_grad():
        result = model(return_loss=False, rescale=True, **data)
        # encode mask results
        if isinstance(result[0], tuple):
            result = [(bbox_results, encode_mask_results(mask_results))
                      for bbox_results, mask_results in result]
    results.extend(result)

    if rank == 0:
        batch_size = len(result)
        for _ in range(batch_size * world_size):
```

```
            prog_bar.update()

    # collect results from all ranks
    if gpu_collect:
        results = collect_results_gpu(results, len(dataset))
    else:
        results = collect_results_cpu(results, len(dataset), tmpdir)
    return results


def collect_results_cpu(result_part, size, tmpdir=None):
    rank, world_size = get_dist_info()
    # create a tmp dir if it is not specified
    if tmpdir is None:
        MAX_LEN = 512
        # 32 is whitespace
        dir_tensor = torch.full((MAX_LEN, ),
                                32,
                                dtype=torch.uint8,
                                device='cuda')
        if rank == 0:
            mmcv.mkdir_or_exist('.dist_test')
            tmpdir = tempfile.mkdtemp(dir='.dist_test')
            tmpdir = torch.tensor(
                bytearray(tmpdir.encode()), dtype=torch.uint8, device='cuda')
            dir_tensor[:len(tmpdir)] = tmpdir
        dist.broadcast(dir_tensor, 0)
```

```
        tmpdir = dir_tensor.cpu().numpy().tobytes().decode().rstrip()
    else:
        mmcv.mkdir_or_exist(tmpdir)
    # dump the part result to the dir
    mmcv.dump(result_part, osp.join(tmpdir, f'part_{rank}.pkl'))
    dist.barrier()
    # collect all parts
    if rank != 0:
        return None
    else:
        # load results of all parts from tmp dir
        part_list = []
        for i in range(world_size):
            part_file = osp.join(tmpdir, f'part_{i}.pkl')
            part_list.append(mmcv.load(part_file))
        # sort the results
        ordered_results = []
        for res in zip(*part_list):
            ordered_results.extend(list(res))
        # the dataloader may pad some samples
        ordered_results = ordered_results[:size]
        # remove tmp dir
        shutil.rmtree(tmpdir)
        return ordered_results


def collect_results_gpu(result_part, size):
```

```
rank, world_size = get_dist_info()
# dump result part to tensor with pickle
part_tensor = torch.tensor(
    bytearray(pickle.dumps(result_part)), dtype=torch.uint8, device='cuda')
# gather all result part tensor shape
shape_tensor = torch.tensor(part_tensor.shape, device='cuda')
shape_list = [shape_tensor.clone() for _ in range(world_size)]
dist.all_gather(shape_list, shape_tensor)
# padding result part tensor to max length
shape_max = torch.tensor(shape_list).max()
part_send = torch.zeros(shape_max, dtype=torch.uint8, device='cuda')
part_send[:shape_tensor[0]] = part_tensor
part_recv_list = [
    part_tensor.new_zeros(shape_max) for _ in range(world_size)
]
# gather all result part
dist.all_gather(part_recv_list, part_send)

if rank == 0:
    part_list = []
    for recv, shape in zip(part_recv_list, shape_list):
        part_list.append(
            pickle.loads(recv[:shape[0]].cpu().numpy().tobytes()))
    # sort the results
    ordered_results = []
    for res in zip(*part_list):
        ordered_results.extend(list(res))
```

```
        # the dataloader may pad some samples
        ordered_results = ordered_results[:size]
        return ordered_results
```

inference.py

```
import warnings

import mmcv
import numpy as np
import torch
from mmcv.ops import RoIPool
from mmcv.parallel import collate, scatter
from mmcv.runner import load_checkpoint

from mmdet.core import get_classes
from mmdet.datasets import replace_ImageToTensor
from mmdet.datasets.pipelines import Compose
from mmdet.models import build_detector


def init_detector(config, checkpoint=None, device='cuda:0', cfg_options=None):
    """Initialize a detector from config file.

    Args:
        config (str or :obj:`mmcv.Config`): Config file path or the config
            object.
        checkpoint (str, optional): Checkpoint path. If left as None, the model
```

```
        will not load any weights.
    cfg_options (dict): Options to override some settings in the used
        config.

Returns:
    nn.Module: The constructed detector.
"""
if isinstance(config, str):
    config = mmcv.Config.fromfile(config)
elif not isinstance(config, mmcv.Config):
    raise TypeError('config must be a filename or Config object, '
                    f'but got {type(config)}')
if cfg_options is not None:
    config.merge_from_dict(cfg_options)
config.model.pretrained = None
config.model.train_cfg = None
model = build_detector(config.model, test_cfg=config.get('test_cfg'))
if checkpoint is not None:
    map_loc = 'cpu' if device == 'cpu' else None
    checkpoint = load_checkpoint(model, checkpoint, map_location=map_loc)
    if 'CLASSES' in checkpoint.get('meta', {}):
        model.CLASSES = checkpoint['meta']['CLASSES']
    else:
        warnings.simplefilter('once')
        warnings.warn('Class names are not saved in the checkpoint\'s '
                      'meta data, use COCO classes by default.')
        model.CLASSES = get_classes('coco')
```

```
    model.cfg = config  # save the config in the model for convenience
    model.to(device)
    model.eval()
    return model


class LoadImage(object):
    """Deprecated.

    A simple pipeline to load image.
    """

    def __call__(self, results):
        """Call function to load images into results.

        Args:
            results (dict): A result dict contains the file name
                of the image to be read.
        Returns:
            dict: ``results`` will be returned containing loaded image.
        """
        warnings.simplefilter('once')
        warnings.warn('`LoadImage` is deprecated and will be removed in '
                      'future releases. You may use `LoadImageFromWebcam` '
                      'from `mmdet.datasets.pipelines.` instead.')
        if isinstance(results['img'], str):
            results['filename'] = results['img']
```

```
            results['ori_filename'] = results['img']
        else:
            results['filename'] = None
            results['ori_filename'] = None
        img = mmcv.imread(results['img'])
        results['img'] = img
        results['img_fields'] = ['img']
        results['img_shape'] = img.shape
        results['ori_shape'] = img.shape
        return results


def inference_detector(model, imgs):
    """Inference image(s) with the detector.

    Args:
        model (nn.Module): The loaded detector.
        imgs (str/ndarray or list[str/ndarray] or tuple[str/ndarray]):
            Either image files or loaded images.

    Returns:
        If imgs is a list or tuple, the same length list type results
        will be returned, otherwise return the detection results directly.
    """

    if isinstance(imgs, (list, tuple)):
        is_batch = True
```

```
else:
    imgs = [imgs]
    is_batch = False

cfg = model.cfg
device = next(model.parameters()).device  # model device

if isinstance(imgs[0], np.ndarray):
    cfg = cfg.copy()
    # set loading pipeline type
    cfg.data.test.pipeline[0].type = 'LoadImageFromWebcam'

cfg.data.test.pipeline = replace_ImageToTensor(cfg.data.test.pipeline)
test_pipeline = Compose(cfg.data.test.pipeline)

datas = []
for img in imgs:
    # prepare data
    if isinstance(img, np.ndarray):
        # directly add img
        data = dict(img=img)
    else:
        # add information into dict
        data = dict(img_info=dict(filename=img), img_prefix=None)
    # build the data pipeline
    data = test_pipeline(data)
    datas.append(data)
```

```
    data = collate(datas, samples_per_gpu=len(imgs))
    # just get the actual data from DataContainer
    data['img_metas'] = [img_metas.data[0] for img_metas in data['img_metas']]
    data['img'] = [img.data[0] for img in data['img']]
    if next(model.parameters()).is_cuda:
        # scatter to specified GPU
        data = scatter(data, [device])[0]
    else:
        for m in model.modules():
            assert not isinstance(
                m, RoIPool
            ), 'CPU inference with RoIPool is not supported currently.'

    # forward the model
    with torch.no_grad():
        results = model(return_loss=False, rescale=True, **data)

    if not is_batch:
        return results[0]
    else:
        return results


async def async_inference_detector(model, imgs):
    """Async inference image(s) with the detector.
```

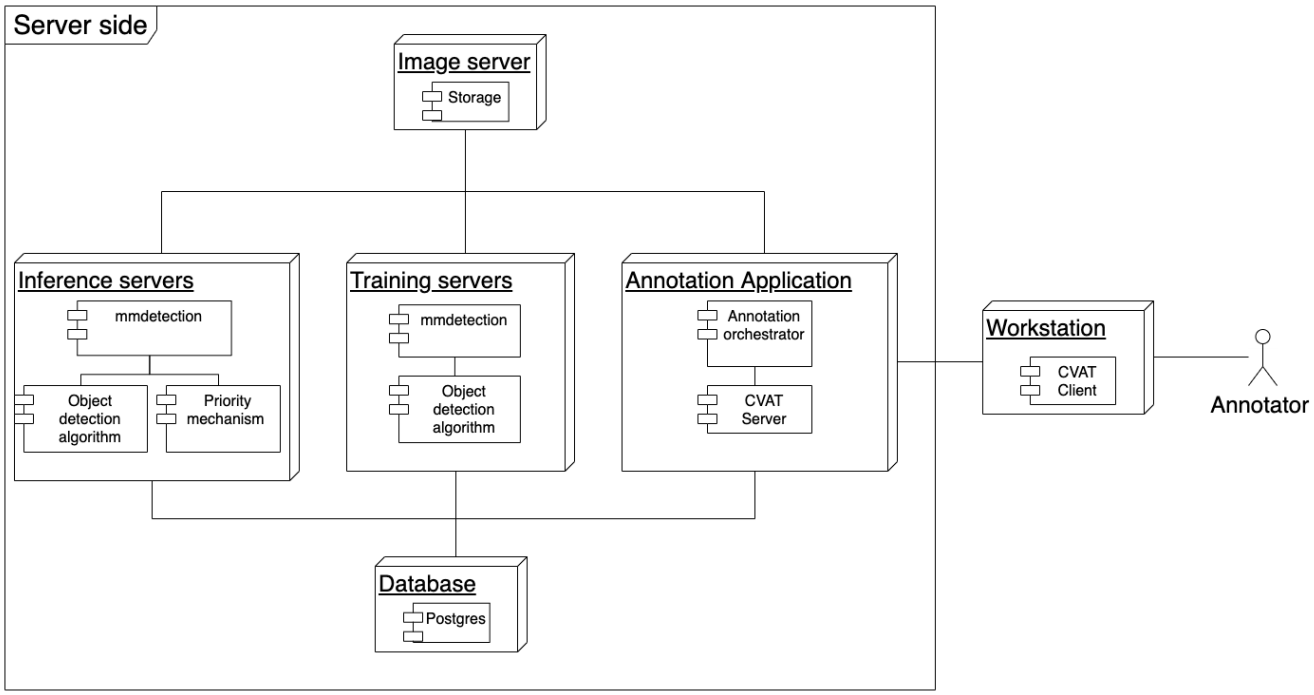
```
Args:
    model (nn.Module): The loaded detector.
    img (str | ndarray): Either image files or loaded images.

Returns:
    Awaitable detection results.
"""
if not isinstance(imgs, (list, tuple)):
    imgs = [imgs]

cfg = model.cfg
device = next(model.parameters()).device  # model device

if isinstance(imgs[0], np.ndarray):
    cfg = cfg.copy()
    # set loading pipeline type
    cfg.data.test.pipeline[0].type = 'LoadImageFromWebcam'

cfg.data.test.pipeline = replace_ImageToTensor(cfg.data.test.pipeline)
test_pipeline = Compose(cfg.data.test.pipeline)

datas = []
for img in imgs:
    # prepare data
    if isinstance(img, np.ndarray):
        # directly add img
        data = dict(img=img)
```

```
    else:
        # add information into dict
        data = dict(img_info=dict(filename=img), img_prefix=None)
    # build the data pipeline
    data = test_pipeline(data)
    datas.append(data)

data = collate(datas, samples_per_gpu=len(imgs))
# just get the actual data from DataContainer
data['img_metas'] = [img_metas.data[0] for img_metas in data['img_metas']]
data['img'] = [img.data[0] for img in data['img']]
if next(model.parameters()).is_cuda:
    # scatter to specified GPU
    data = scatter(data, [device])[0]
else:
    for m in model.modules():
        assert not isinstance(
            m, RoIPool
        ), 'CPU inference with RoIPool is not supported currently.'

# We don't restore `torch.is_grad_enabled()` value during concurrent
# inference since execution can overlap
torch.set_grad_enabled(False)
results = await model.aforward_test(rescale=True, **data)
return results
```

```
def show_result_pyplot(model,
                       img,
                       result,
                       score_thr=0.3,
                       title='result',
                       wait_time=0):
    if hasattr(model, 'module'):
        model = model.module
    model.show_result(
        img,
        result,
        score_thr=score_thr,
        show=True,
        wait_time=wait_time,
        win_name=title,
        bbox_color=(72, 101, 241),
        text_color=(72, 101, 241))
```

ДОДАТОК Б.
ГРАФІЧНИЙ МАТЕРІАЛ

ДОДАТОК Б. ЛИСТ 1 – ДІАГРАМА РОЗГОРТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



ДОДАТОК Б. ЛИСТ 2 – ДІАГРАМА ДІЯЛЬНОСТІ ПРОЦЕСУ РОЗМІТКИ ДАНИЗ З МЕХАНІЗМОМ ПРІОРИТЕЗАЦІЇ