Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів детектування та моніторингу людини у відеопотоці
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Миронюк С. А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник проф. Четвериков Г.Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Миронюку Сергію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів детектування та моніторингу людини у відеопотоці»

Затверджена наказом по університету від 29.03. 2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані Інтернет-джерел, відкрита програмна бібліотека для машинного навчання TensorFlow, відкрита нейромережна бібліотека Keras, мова програмування Python.

4. Перелік питань, що потрібно опрацювати в роботі

вступ, аналіз предметної області, виявлення проблем, дослідження існуючих методів детектування об'єктів, розробка та реалізація комбінованої нейромережі для детектування людини в режимі реального часу, тестування для оцінки ефективності розробленої моделі, висновки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області, постановка задачі, огляд літератури з обраної теми	31.03.2024	виконано
2	Огляд сучасних методів. Їх порівняльний аналіз	15.04.2024	виконано
3	Розробка комбінованої мережі	01.05.2024	виконано
4	Проведення тестування	15.05.2024	виконано
5	Підготовка пояснювальної записки	28.05.2024	виконано
6	Підготовка презентації та доповіді	08.06.2024	виконано
7	Нормоконтроль	12.06.2024	виконано
8	Рецензування	14.06.2024	виконано
9	Попередній захист	15.06.2024	виконано
10	Занесення в електронний архів	16.06.2024	виконано
11	Захист кваліфікаційної роботи	19.06.2024	виконано

Дата видачі завдання «28» лютого 2024 р.

Студент _____ Миронюк С.А.
(підпис)

Керівник роботи _____ проф. Четвериков Г.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Звіт: 91 сторінка, 38 рисунків, 2 таблиці, 35 посилань.

ВІДСЛІДКОВУВАННЯ ОБ’ЄКТІВ, ДЕТЕКТУВАННЯ ОБ’ЄКТІВ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, TRANSFER LEARNING.

Об'єктом дослідження є відеоматеріали із зображенням людей, що зафіксовані камерами відеонагляду.

Предмет дослідження – методи детектування та відслідковування людей у відеопотоці.

Мета дослідження полягає у підвищенні результативності детектування та відслідковування людини в режимі реального часу за допомогою оптимізації нейронної мережі YOLO.

В результаті роботи було здійснено аналіз предметної області, огляд технічної літератури за темою роботи, огляд існуючих технік та підходів до детекції та відслідковування об'єктів у відеопотоці, проведенно порівняльний аналіз продуктивності існуючих методів, розроблено комбіновану нейронну мережу для оперативного та точного детектування та відслідковування людини в відеопотоці.

COMPUTER VISION, OBJECT DETECTION, TRACKING OBJECTS, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS, TRANSFER LEARNING.

The object of the research is video footage of people recorded by video surveillance cameras.

The subject of research is methods of detecting and tracking people in a video stream.

The purpose of the study is to improve the effectiveness of human detection and tracking in real time by optimizing the YOLO neural network.

As a result of the work, an analysis of the subject area was carried out, a review of the technical literature on the topic of the work, a review of existing techniques and approaches to the detection and tracking of objects in the video stream, a comparative analysis of the performance of existing methods was carried out, a combined neural network was developed for operational and accurate human detection and tracking in the video stream.

Я, Миронюк Сергій Анатолійович, студент гр.ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів детектування та моніторингу людини у відео потоці», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату, і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CEPDOF – Challenging Events for Person Detection from Overhead Fisheye Images

CNN – Convolutional Neural Networks

HOG – Histogram of Oriented Gradients

FPN – Feature Pyramid Network

FPS – frames per second

mAP – mean average precision

IoU – intersection over union

NMS – non-maximum suppression

R-CNN – Region-based Convolutional Neural Network

ReLU – rectified linear unit

RoI – Region of Interest

RPN – Region Proposal Network

SORT – Simple Online and Realtime Tracking

SSD – Single Shot MultiBox Detector

YOLO – You Only Look Once

ВСТУП

За останній час відеоаналітика, яка використовує методи комп'ютерного зору для автоматизованого збору та аналізу інформації з відеокамер, стала популярною технологією. Застосовується вона у різних сферах, включаючи системи безпеки, відеоспостереження, торгівлю та транспорт.

Використання відеоаналітики широко поширюється як серед урядів, так і підприємств. За прогнозами компанії MarketsandMarkets [1], цей ринок значно зросте з 8,3 мільярда доларів у 2023 році до 22,6 мільярда доларів до 2028 року. Цей ріст прогнозується за середньорічним темпом зростання на рівні 22,3% протягом прогнозованого періоду. Основними факторами, що підтримують цей ринок, є зростаюча популярність відеоаналітики, розвиток інтелектуальних камер та передових технологій, таких як глибоке навчання. Різноманітні вертикалі, такі як критична інфраструктура, уряд і оборона, транспорт і логістика, вибирають відеоаналітику для підвищення безпеки. Очікується, що значні державні інвестиції в проєкти модернізації інфраструктури, такі як «розумне місто», стануть ключовими факторами для впровадження програмного забезпечення та послуг відеоаналітики в майбутньому.

У сьогоденні важливим завданням для відеоаналітики є детекція та моніторинг людини у відеопотоці. Системи розпізнавання та спостереження за людьми у відеопотоці матимуть широке застосування в різних галузях, включаючи автоматичне керування з метою виявлення пішоходів на вулицях, аналіз поведінки споживачів у торговельних закладах, клієнтська активність, визначення навантаження на персонал, моніторинг пасажиропотоку на громадському транспорті та в аеропортах, а також в системах «розумного будинку» та офісних комплексах.

Як правило рішення відеоаналітики включають у себе спеціалізовані програмні продукти для обробки даних, отриманих з вебкамер, разом з інтелектуальною оцінкою ситуації. Також наявні окремі відеокамери з інтегрованими можливостями відеоаналітики. Методи програмного детектування

та відслідковування осіб мають різноманітність, включаючи спеціальні додатки, веб-застосунки та ізольовані програмні модулі.

В останні часи нейронні мережі набули активного застосування у відеоаналітиці. Вони можуть бути вбудовані прямо в камери. Проте, цей підхід може стикатися з проблемами розпізнавання невеликих або часткових об'єктів, а також нестійкістю до зашумлених або з низьким рівнем освітлення зображень. Також навчання мереж та процес виявлення можуть займати значну кількість часу. Окремі нейромережі оброблюють відео з недостатньою швидкістю, що утруднює їх використання в реальному часі.

Отже актуальним є створення ефективної мережі для детектування та моніторингу людей, що сприятиме швидкішій обробці відеокадрів та дозволить проводити її більш точно.

Мета дослідження полягає у підвищенні ефективності процесу детектування та відслідковування людини в режимі реального часу за допомогою вдосконалення нейронної мережі моделі YOLO.

Для досягнення поставленої мети було сформульовано наступні завдання:

- дослідження актуальності обраної проблеми;
- проведення огляду технічної літератури в обраній тематиці;
- аналітичний огляд існуючих методів розпізнавання об'єктів в відеопотоці;
- проведення порівняльного аналізу продуктивності сучасних детекторів на відеодатасеті CEPDOF;
- вибір середовища програмування;
- розробка мережі на основі YOLO з додатковими функціями HOG-дескриптора;
- оцінка ефективності роботи розширеної моделі.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Комп’ютерний зір

У наш час наявні технології, які дозволяють ефективно аналізувати великі обсяги даних і зображень. Одним із найбільш вражаючих прикладів є комп'ютерний зір, що вміє розпізнавати та обробляти об'єкти на зображеннях та відео на рівні, подібному до людини [2]. До недавніх пір комп'ютерний зір мав обмежені можливості, але зараз він значно розвивається та удосконалюється.

Унаслідок прориву в галузі штучного інтелекту, нейронних мереж і передовим технологіям у глибокому навчанні, ця сфера зазнала значних змін у останні роки і тепер може випередити людину у деяких задачах щодо розпізнавання об'єктів.

У галузі комп'ютерного зору зазвичай розрізняють наступні основні типи завдань, які є найпоширенішими:

- класифікація об'єктів: система досліджує візуальний контент і розподіляє об'єкти на фотографіях або відео за визначеними категоріями;
- виявлення об'єктів: система досліджує вміст зображення і встановлює наявність конкретних об'єктів на фотографіях або відео;
- відслідковування об'єктів: система обробляє відео, виявляє об'єкти, що відповідають заданим критеріям, і стежить за їх рухом [3].

Комп'ютерний зір здатний створювати перетворене зображення чи перелік значень конкретних параметрів зображення, зокрема як розмір об'єкту, колір, положення камери, швидкість і т. д.

Розпізнавання образів – це галузь науки, присвячена розробці принципів і побудові програмних і апаратних систем, мета яких – визначити, якому з наперед визначених класів належать об'єкт, що аналізується.

Розпізнавання образів включає в себе класифікацію вихідних даних до певних категорій шляхом виділення ключових ознак або характеристик із загальної маси ознак. Це охоплює завдання комп'ютерного зору, такі як виявлення, локалізація та класифікація об'єктів. Виявлення об'єктів включає

визначення присутності об'єкта в просторі, його місцезнаходження та прив'язку до визначеного класу. Це завдання об'єднує у собі як локалізацію, та класифікацію.

Об'єкт – це конкретна область в просторі ознак, де ознака виступає як властивість або характеристика об'єкта чи явища у навколишньому середовищі. Простіром ознак є N-вимірним простіром, який визначається для конкретного завдання розпізнавання, де N – кількість ознак. Вектор ознак x, що використовується для розпізнавання об'єктів, є N-вимірним вектором із компонентами (x_1, x_2, ..., x_N). Кожна компонента цього вектора представляє значення певної ознаки для цього об'єкта[4].

Існують різноманітні методи детекції об'єктів на зображенні. Відео, як більш складний формат, має додатковий параметр – це час. Проте, відео можна розглядати як послідовність зображень. Для обробки його зазвичай потрібно пройтися по всіх кадрах у відеофайлі та задіяти відповідні алгоритми детекції об'єктів для кожного кадру. Проте використання одного й того ж методу для кожного кадру може бути неефективним з погляду обчислень, і не завжди гарантує, що об'єкт буде виявлено у кожному кадрі (адже не всі детектори стійкі до різноманітних позицій та ракурсів). Тому часто застосовуються алгоритми відслідковування об'єктів для додаткової обробки відео.

Трекінг або відслідковування об'єктів – це процес визначення розташування рухомого об'єкта або кількох об'єктів в просторі з плином часу. Це також включає прогнозування подальшого руху об'єкта, навіть якщо він на час зникає з області стеження (наприклад, заходить за перешкоду). Трекінг може бути спрямованим на один об'єкт або групу об'єктів.

Відомі різноманітні методи відслідковування:

Кореляційний метод: використовує навчання кореляційного фільтра для виділення об'єкту від фону на кадрі. Цей метод будує траєкторії лише для певних об'єктів, які визначені для пошуку.

Метод на основі графів: використовує графічні моделі для відстеження об'єктів та їх зв'язків.

Метод мультикамерного спостереження: передбачає застосування синхронізованих камер, які спостерігають за зв'язаними областями. Об'єкт переміщується з області спостереження однієї камери до області спостереження іншої.

Метод відслідковування на основі детектуваня (tracking-by-detection): де спершу застосовується алгоритм виявлення об’єкта, потім його координати з попереднього кадру застосовуються для обчислення в наступному кадрі. Алгоритм SORT (Simple Online and Realtime Tracking) є методом відслідковування на основі детектування, оскільки він спочатку використовує алгоритм виявлення об'єктів для отримання координат, а потім використовує ці координати для відстеження об'єктів у наступних кадрах.

Методи конкретно розпізнавання об'єктів включають в себе використання колірних фільтрів, виокремлювання контурів, методи зіставлення з шаблоном, використання ознак Хаара для виділення об'єкта на зображенні, метод опорних точок, нейронні мережі тощо. З метою відслідковування можна використовувати алгоритми, основані на мірі Жаккара, фільтрі Калмана або вилучення ознак. Вибір певного методу обумовлений поставленою задачею. Далі будуть розглянуті такі класичні алгоритми як Віоли-Джонса і гістограм напрямлених градієнтів, а також нейронні мережі для детектування об'єктів. Відслідковування об'єктів буде здійснюватися на основі детектування.

1.2 Основні проблеми детекції та моніторингу людей у режимі реального часу

Труднощі і проблеми, які можуть виникати при детекції та моніторингу людини у режимі реального часу, включають:

Накладання об'єктів: ситуації, коли люди перетинаються чи накладаються друг на друга, можуть ускладнити визначення меж та руху окремих осіб.

Зміна масштабу: об'єкти можуть знаходитися на різних відстанях від камери, що призводить до їхнього різного розміру та вигляду, що ускладнює відслідковування людей [5] [6].

Недостатнє освітлення: В умовах недостатнього освітлення камера може має недостатню інформацію для коректного визначення обличчя або руху людини, що може призвести до помилок у детекції та моніторингу.

Перепади освітлення: різка зміна освітлення, така як тінь від рухомих предметів або мінливе природне освітлення, може спричинити зміну в якості та контрастності зображення, що також може ускладнити процес детекції.

Зашумленість також може стати проблемою при детекції та моніторингу людини у режимі реального часу. У великих містах або на переповнених вулицях, може бути багато рухомих об'єктів, які створюють зашумленість на зображенні, ускладнюючи процес виявлення та відслідковування людини.

Рух камери: нестабільність або рух камери може призвести до зміни позиції об'єктів та ускладнити відслідковування їхнього руху.

Обробка великого обсягу даних: велика кількість кадрів у відеопотоках вимагає високої обчислювальної потужності та швидкості обробки для трекінгу кожного об'єкта.

Необхідність реального часу: Вимога відстеження об'єктів у реальному часі потребує швидкої обробки даних та ефективного використання ресурсів для забезпечення точності та оперативності системи моніторингу.

З метою подолання цих труднощів використовуються різноманітні підходи, такі як комп'ютерний зір, глибинне навчання та інші методи. Продовження досліджень і розвиток нових підходів допоможуть покращити ефективність детектування і моніторингу людей у відеопотоках та вирішити ці проблеми.

1.3 Метод Віола-Джонса

За останні два десятиліття еволюція методів виявлення об'єктів умовно ділиться на два основні історичні періоди. Перший період, що тривав до 2014 року, характеризувався використанням традиційних методів виявлення, які базувалися на ручному створенні ознак. Другий період, що розпочався з 2014 року, відзначається переходом до методів виявлення, що ґрунтуються на глибокому навчанні.

У 2001 році Пол Віола та Майкл Джонс представили метод Віоли-Джонса [7], який визначив новий стандарт продуктивності при обробці зображень у реальному часі. Попри те, що спочатку головним призначенням алгоритму було розпізнавання облич, його можна застосовувати для розпізнавання різних видів об'єктів. Існує численні варіації цього алгоритму, включаючи ті, що доступні у бібліотеці комп'ютерного зору OpenCV. Цей алгоритм має багато різних реалізацій, включаючи ті, що можна знайти у бібліотеці OpenCV.

Основні принципи цього методу включають:

- принцип інтегрального представлення образа дозволяє швидко обчислювати потрібні об'єкти. Інтегральне представлення зображення – це матриця, що відповідає розмірам вхідного зображення у пікселях, де кожен елемент представляє суму інтенсивностей всіх пікселів, які розміщених ліворуч та на більш високому рівні даного елемента;
- використання знаків, аналогічних знакам Хаара [7]. Це включає в себе використання знаків Хаара для виявлення об'єкта, що шукається;
- вибір ознак проводиться через алгоритм бустінгу – послідовного представлення композиції алгоритмів машинного навчання, де кожен наступний алгоритм старається компенсувати помилки у композиції всіх минулих алгоритмів;
- класифікатор визначає, чи належить об'єкт до конкретного класу. Класифікатор, що отримує символи як вхідні дані, повертає відповідь «true» або «false», що дозволяє визначити;
- каскади функцій застосовуються під час виявлення. Ця концепція полягає в створенні послідовності класифікаторів, яку називають каскадом: кожен наступний класифікатор старається усунути помилки попереднього;
- для аналізу зображень застосовується метод віконного сканування: зображення обробляється вікном пошуку, потім класифікатор використовується до кожної позиції.

Переваги цього методу включають його популярність, високу швидкість розпізнавання та точність порівняно з іншими алгоритмами. Однак для його ефективності необхідна велика дослідницька вибірка та час на навчання. Також існують обмеження на точне положення об’єкта під час виявлення.

AdaBoost (Adaptive Boosting), запропонований Йоавом Фройндом і Робертом Шапіро в 1999 році [8] – алгоритм, який можна використовувати разом із методом Віоли-Джонса для покращення його продуктивності. AdaBoost – адаптивний алгоритм, оскільки кожна подальша композиція базується на неправильно класифікованих об'єктах попередніми комітетами класифікаторів.

Алгоритм AdaBoost працює послідовно, застосовуючи функції до зображень та вибираючи ті, які мають найменші помилки. Кожне зображення спочатку має однакову вагу, яка збільшується після кожної неправильної класифікації. Цей процес триває до досягнення необхідної точності або певної кількості характеристик.

Для ефективної обмеження кількості використовуваних ознак та збереження високої точності виявлення застосовується каскадна класифікація. Це дозволяє швидко скорочувати кількість ознак, зберігаючи точність.

Переваги методу включають його адаптованість до складних елементів навчальної вибірки, здатність досягнення нульової помилки навчання за обмежену кількість ітерацій, а також високу швидкість роботи та простоту реалізації.

Серед недоліків можна відзначити чутливість алгоритму до викидів та шуму. Також, навчання вимагає значного часу, що залежить від кількості класифікаторів та розміру навчальної вибірки.

Як працює метод Віоли-Джонса. Щодо розрахунку яскравості прямокутної області на зображенні, метод Віоли-Джонса використовує інтегральне представлення [9]. Цей підхід широко застосовується в численних алгоритмах комп'ютерного зору і надає можливість ефективно обчислювати загальну яскравість будь-якого прямокутника на заданому кадрі, при цьому час обчислення не залежить від площі прямокутника.

Інтегральне представлення зображення визначається матрицею, та її розміри відповідають розмірам вхідного зображення. Кожен елемент цієї матриці містить суму інтенсивностей пікселів, розташованих ліворуч та вище від цього елемента. Формула для розрахунку елементів матриці виглядає наступним чином (формула 1.1):

$$I(x, y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{1.1}$$

де $I(x, y)$ – визначає значення точки (x, y) в інтегральному зображенні;

$i(x', y')$ – представляє собою значення інтенсивності зображення вхідного.

Використання цього підходу дозволяє одночасно обчислювати ознаки одного типу з різними геометричними параметрами. Однак розрахунок матриці інтегрального представлення вимагає лінійного часу, пропорційного кількості пікселів на фотографії.

Підхід до використання функцій виявлення об'єктів за допомогою ознак Хаара був вперше представлен Папагоргіу у 1998 році. Віола та Джонс вдало адаптували цю концепцію у своїй роботі, створивши прямокутні ознаки, відомі як ознаки Хаара, зовнішній вигляд яких показано на рисунку 1.1 [10].

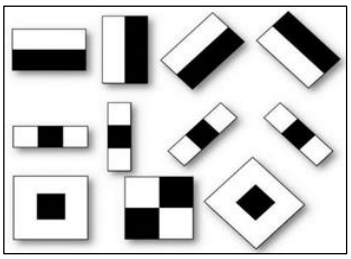


Рисунок 1.1– Ознаки Хаара

Вдосконалений метод Віоли-Джонса, що включений до бібліотеки комп'ютерного зору OpenCV також використовує додаткові ознаки, які представлені на рисунку 1.2.

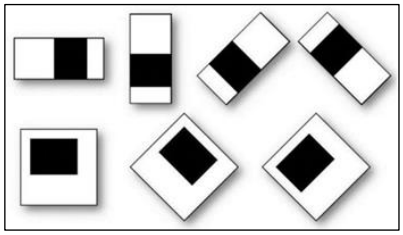


Рисунок 1.2 – Додатково розгорнуті ознаки Хаара

Відображення результатів розрахунку цих ознак у формі інтегрального представлення буде виглядати наступним чином (формула 1.2):

$$F = U - V| \tag{1.2}$$

де U – сума яскравості точок, закритих світлою частиною ознаки;

V – сума значень яскравості точок, покритих темною частиною ознаки.

Ці ознаки визначають зміну яскравості по двом осям зображення.

Виявлення об'єкта виконується за допомогою вікна сканування, розмір якого в оригінальному алгоритмі становить 24×24 пікселі. Це вікно переміщується по зображенню з кроком у 1 піксель, і для кожної його позиції обчислюються ознаки Хаара з різним масштабом і положенням у вікні.

Сам процес сканування відбувається аналогічним чином незалежно від різних масштабів скануючого вікна. Знайдені ознаки Хаара передаються класифікатору, який використовує їхні значення для визначення того, чи зображення в області вікна є шуканим об’єктом – чи ні.

Структура класифікатора у вигляді каскаду пришвидшує процес розпізнавання об’єктів, фокусуючись на найбільш цікавих зонах зображення. Каскад створює організацію слабких класифікаторів у структурну послідовність, яку вони вивчили за допомогою процедури підсилення. Таким чином, за незначних обчислювальних витратах можна високо ймовірно відкидати зображення на ранніх етапах, які не містять шуканого об'єкта. Каскадна структура класифікатора наведена на рисунку 1.3.

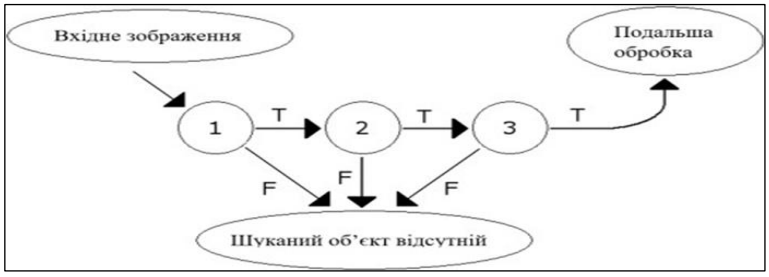


Рисунок 1.3 – Каскадний класифікатор

На кожному рівні каскаду використовується алгоритм навчання AdaBoost, про який було згадано раніше [10]. Кількість застосованих ознак поступово збільшується, поки виявлення цільового об'єкта та помилки 1-го типу не досягнуть заданих значень.

Рівні формуються через тестування детектора на тестовому наборі. Додатковий рівень до каскаду додається у випадку, коли загальна помилка першого типу для всього об'єкта не відповідає заданому значенню.

Негативний набір для наступних шарів формується з усіх помилкових виявлень при використанні поточного каскаду.

Після проведеної класифікації отримується набір зон зображення, які містять об'єкт пошуку. Під час наступної обробки усуваються внутрішні повтори, що виникають в результаті масштабування скануючого вікна та виявлення одного й того ж об'єкта. Об’єкт перетворюються в градації сірого та масштабуються до розміру 128×128 пікселів для подальшого аналізу.

1.4 Метод гістограм напрямлених градієнтів

Метод гістограм напрямлених градієнтів (Histogram of Oriented Gradients, HOG) є ефективним засобом екстрагування характеристик зображень для подальшої їх класифікації. Представлений у 2005 році дослідниками INRIA Далалом та Тріггсом [11], цей метод виявився дуже корисним у великому спектрі завдань комп'ютерного зору та розпізнавання образів, особливо у реальному масштабі часу. Хоча HOG може використовуватися для виявлення різних об'єктів, але основним каталізатором його розвитку став саме процес виявлення пішоходів.

Основний принцип HOG в тому, щоб вимірювати локальні градієнти яскравості (або кольору) в різних частинах зображення. Це відбувається за допомогою розбиття зображення на малий набір клітин, а потім обчисленням градієнтів у кожній з цих клітин. Градієнти можуть бути виміряні в різних напрямках, що дозволяє захоплювати інформацію про текстуру та форму об'єктів на зображенні.

У порівнянні з методом масштаб-незалежного перетворення ознак (Scale-Invariant Feature Transform, SIFT), HOG відрізняється тим, що використовує щільну сітку рівномірно розміщених клітин, що дозволяє захоплювати більше локальної інформації. Щоб підвищити точність, HOG використовує метод, який адаптивно коригує значення градієнтів у кожній області зображення відносно їх сусідів. Цей процес називається локальною нормалізацією контрасту. Він забезпечує більш однорідне та стабільне представлення градієнтів, що допомагає у зменшенні впливу змін яскравості на класифікаційні результати. Іншими словами, цей метод створює умови для більш надійного виявлення важливих особливостей на зображенні, що підвищує ефективність класифікації об'єктів.

Щоб реалізувати алгоритм HOG, спочатку необхідно піддати зображення попередній обробці, включаючи зміну розміру та нормалізацію кольору.

Після цього для кожного пікселя зображення обчислюють вектор градієнта, його величину та напрямок. Це відбувається шляхом фільтрації зображення горизонтальними і вертикальними ядрами (рис. 1.4). За формулами, що зображені на рисунку 1.4 визначаються величина і напрямок градієнтів:

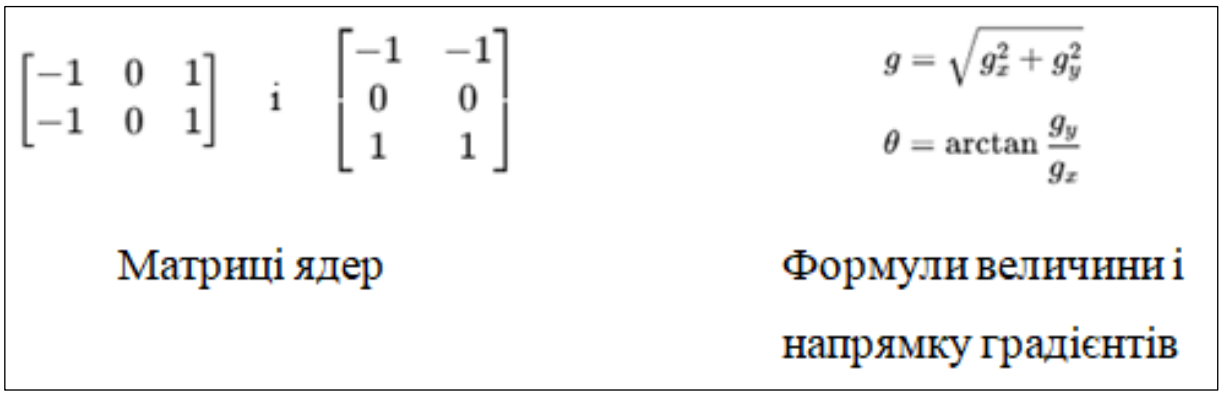


Рисунок 1.4 – Методи обчислення градієнтів за допомогою матриць ядер

Де g – величина градієнта, θ – направлення, g_x, g_y – градієнти по осі X і Y відповідно (рис. 1.5).

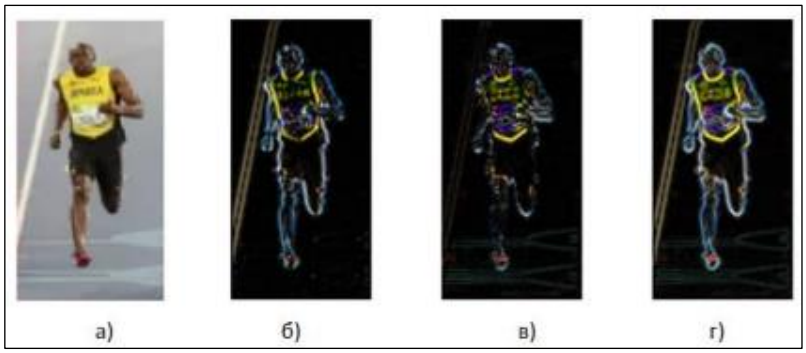


Рисунок 1.5 – а) початкове зображення, б) градієнт по осі X, в) градієнт по осі Y, г) величина градієнта [12]

Градієнт надає нам інформацію про місця на зображенні, де інтенсивність різко змінюється. Він використовується для виділення контурів об'єктів, залишаючи осторонь менш важливу інформацію, таку як кольори.

Для кольорових зображень аналізуються градієнти у трьох каналах. При кольорових зображеннях аналізуються градієнти у трьох каналах (R, G, B). Величина градієнта в пікселі обчислюється як максимальне значення серед градієнтів у каналах R, G та B, а напрямок градієнта визначається за каналом, де цей градієнт є максимальним.

Після цього зображення розділяється на менші області – комірки, для яких розраховується гістограма градієнтів, що являє собою вектор з 9 бінів. Ці біни (числа) у випадку беззнакових градієнтів відповідають кутам від 0˚ до 160˚, а у випадку зі знаковими градієнтами – від 0˚ до 360˚.

Процес розподілу значень магнітуди градієнту в гістограмі напрямлених градієнтів полягає в тому, що для кожного значення напрямлення кута θ визначається, в яку позицію гістограми записується відповідне значення g. Якщо значення кута θ не відповідає жодному з фіксованих значень кутів у гістограмі, то значення градієнту розподіляється пропорційно поміж 2-ма сусідними кутами гістограми [12]. На рисунку 1.6 можна спостерігати, що для кута 10° значення

рівномірно розподілене між нульовою та 1-ю коміркою, відповідно до кутів 0° та 20°.

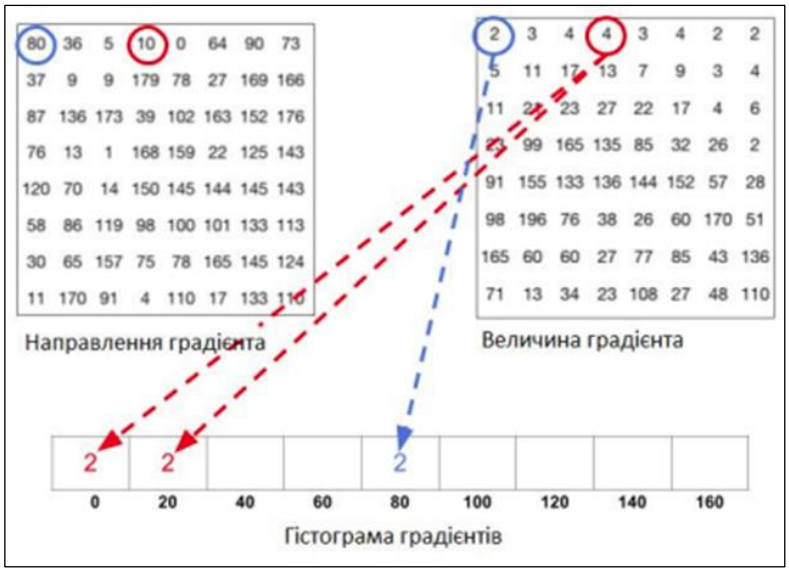


Рисунок 1.6 – Формування гістограми градієнтів

Коли кут перевищує 160˚, він розглядається як еквівалентний куту від 0 до 20˚ градусів. Таким чином, наприклад, піксель з кутом 165˚ градусів буде мати внесок як у відрізок від 0 до 20˚, так і у відрізок від 160˚ до 180˚ (рис. 1.7).

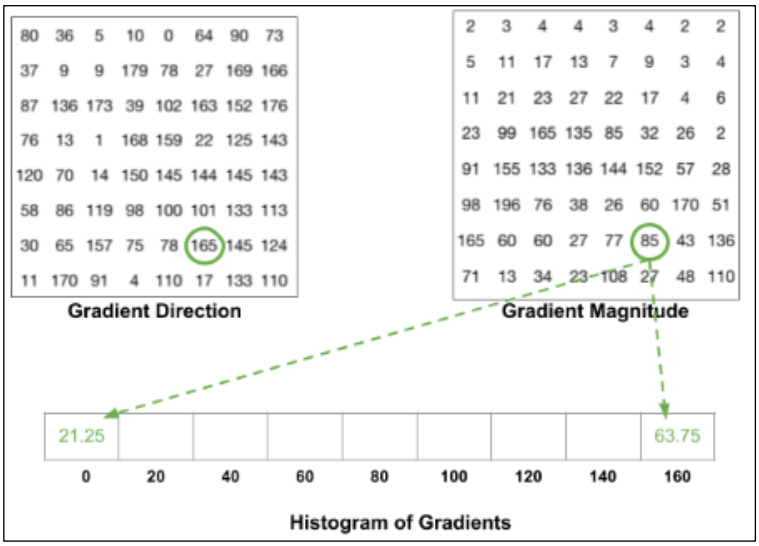


Рисунок 1.7 – Розрахунок гістограми градієнтів

Здійснюється нормалізація обчислених гістограм градієнтів використанням двох типів норм (формула 1.3):

$$h_{L2} = \frac{h}{\sqrt{|h|_2^2 + e^2}}, h_{L1} = \frac{h}{|h|_1 + e}, \sqrt[k]{L_1} = \sqrt{h_{L_1}}, \quad (1.3)$$

де h – відповідає цим гістограмам;

e – є малою константою.

Нормалізація L_2 враховує суму квадратів всіх значень в гістограмі, тоді як нормалізація L_1 враховує суму абсолютних значень. Ця процедура сприяє покращенню стійкості до змін у рівні освітлення, затінення і контрасту на зображенні.

На фінальному кроці, локальні гістограми, які визначені у всіх блоках, що перекривають вікно виявлення об'єкта, об'єднуються у зведений вектор ознак. Цей зведений вектор ознак використовується для подальшої класифікації.

Метод HOG виявляється потужним і продуктивним засобом для вилучення ознак із зображень, що сприяє підвищенню точності та ефективності систем розпізнавання об'єктів, особливо в реальному часі.

Отже, хоча метод Віоли-Джонса досяг значного успіху у завданні розпізнавання об'єктів, він навчається повільніше відносно інших алгоритмів, тому що вимагає великої кількості ознак. HOG використовує локальні комірками, що робить його інваріантним до геометричних та фотометричних спотворень об'єкта. Однак класичні методи виявлення об'єктів найкраще підходять для сценаріїв, де доступний обмежений обсяг тренувальних даних. У той час як для обробки великих обсягів даних, зокрема датасетів з більш ніж 1000 зображень, ефективніше застосовувати нейронні мережі, які, імітуючи роботу людського мозку, краще впораються з завданнями обробки зображень.

1.5 Штучні нейронні мережі

Штучна нейронна мережа представляє собою математичну модель, яка імітує роботу біологічних нейромереж в людському мозку. Ця модель складається з великої кількості простих елементів, які називаються нейронами, і має структуру, де зв'язки між нейронами організовані у вигляді шарів. Спосіб, яким

нейрони пов'язані між собою, визначається типом мережі [13]. Кожен нейрон має вхідні сигнали, а його вплив на інші нейрони визначається за допомогою зв'язків, які мають вагу. Ваги цих зв'язків визначають важливість кожного входу для нейрона. Для створення нейронної мережі з метою вирішення певного завдання потрібно визначити, як встановити взаємозв'язки між нейронами та оптимізувати значення вагових параметрів для цих зв'язків.

В залежності від того, скільки шарів має мережа, вона може бути або одношаровою, або багатошаровою (БНМ). В БНМ нейрони кожного шару виступають як вхідні сигнали для нейронів наступного шару. Повнозв'язні мережі мають зв'язки між всіма нейронами кожного шару та всіма нейронами сусідніх шарів, тоді як у неповнозв'язних мережах деякі зв'язки можуть бути відсутніми (рис. 1.8)

Повнозв'язна нейронна мережа Неповнозв’язна нейронна мережа

Рисунок 1.8 – Порівняння структур обох типів нейронних мереж

1.5.1 Багатошаровий персептрон

Штучний нейрон, який відомий як персептрон, представляє собою одну з простих моделей штучного інтелекту. Персептрон отримує вхідні дані, зважує їх відповідно до їх ваги, і використовує функцію активації для генерації вихідного сигналу, який передається на вихідний шар (рис. 1.9).

Багатошаровий персептрон – це тип нейронної мережі, де вхідний сигнал проходить через кілька послідовних шарів нейронів, перетворюючись на вихідний сигнал. Відсутність зворотних зв'язків, які є характерними для рекурентних

мереж, означає, що інформація рухається лише в одному напрямку, без зворотнього впливу на попередні шари.

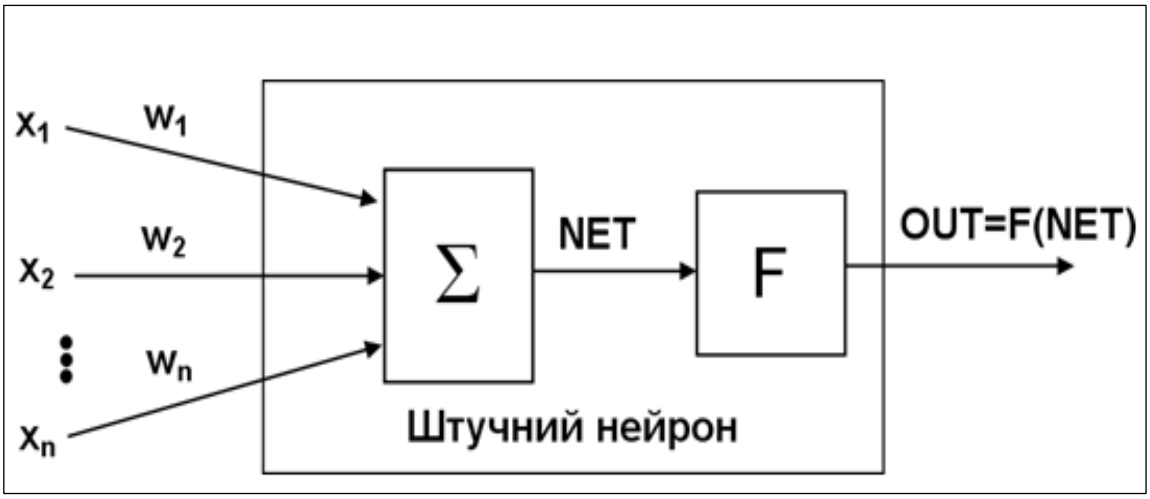


Рисунок 1.9 – Сруктура штучного нейрону

Кожен елемент у одному шарі мережі з'єднаний з кожним елементом у наступному шарі, створюючи мережу з повними зв'язками. Ця архітектура широко використовується в завданнях, таких як розпізнавання мови та машинний переклад.

1.5.2 Функція активації НМ

Функція активації є ключовим елементом, що визначає поведінку нейронної мережі. Функція активації додає нелінійність в роботу нейрона, що дає можливість виявляти складніші відносини у даних та надає гнучкість у функціонуванні нейронної мережі. Функції активації можуть приймати вхідні значення та перетворювати їх у вихідні сигнали, що відображають активаційний стан нейрона. Деякі з найпоширеніших функцій активації включають в себе логістичну (сигмоїдну), гіперболічний тангенс, ReLU (Rectified Linear Unit), і сигмоїду.

Кожна з цих функцій має свої переваги та недоліки, і їх вибір залежить від конкретного завдання та архітектури мережі. Наприклад, ReLU часто використовується через свою простоту та ефективність у навчанні глибоких

мереж, тоді як сигмоїда може бути корисною для задач класифікації з великими вихідними діапазонами.

Сигмоїда (Sigmoid) – це функція активації, яка стискає будь-яке вхідне значення до діапазону між 0 та 1, дозволяючи нейрону генерувати вихід, що завжди перебуває в цьому діапазоні.

ReLU – це функція активації, яка обнуляє від'ємні значення вхідних даних, залишаючи додатні без змін, що сприяє ефективному опрацюванню нелінійних задач нейронами.

Tanh (гіперболічний тангенс) – це функція, аналогічна сигмоїді, але вона перетворює вхідні дані до значень від -1 до 1, що робить її універсальнішою за сигмоїду.

Особливості та недоліки більш поширених функцій активації представлені на рисунку 1.10.

Функція активації	Формула	Використання	Недоліки
Порогова	$f(x) = 1$, *якщо* $x >= t$, *інакше* 0, *де* t *— порогове значення*	Рідко	Безперервність, відсутність градієнта
Лінійна	$f(x) = x$	Прості моделі	Обмежене уявлення нелінійностей
Гіперболічний тангенс	$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Приховані шари	Згасаючий градієнт
Сигмоїда	$f(x) = \frac{1}{1+exp(-x)}$	Задачі класифікації	Згасаючий градієнт
ReLU	$f(x) = max(0, x)$	Глибокі мережі	«Мертві нейрони» з негативним входом
Leaky ReLU	$f(x) = max(0,01x, x)$	Глибокі мережі	Додатковий параметр нахилу може бути неоптимально налаштований, що може впливати на продуктивність моделі.
Parametric ReLU (PReLU)	$f(x) = max(ax, x)$, *де* a *— параметр*	Глибокі мережі	Обчислювальна складність
ELU (Exponential ReLU)	$f(x) = x$ *за умови* $x >= 0$, $a(exp(-1))$ *за умови* $x < 0$	Глибокі мережі	Наявність негативних значень може спричинити збільшення складності оптимізації моделі.
Maxout	$f(x) = max(w1 * x + b1, w2 * x + b2)$, *де* $w1$, $w2$ *— вагові коефіцієнти,* $b1$, $b2$ *— зміщення*	Глибокі мережі	Збільшення обчислювальної складності та вимог більшого обсягу даних для навчання

Рисунок 1.10 – Порівняння поширених функцій активації та окремих модифікацій ReLU [14]

Важливо розуміти, що функція активації – це просто правило, яке визначає, як нейрон реагує на інформацію, яку він отримує. Вона допомагає зробити роботу нейрона більш гнучкою та адаптивною, що, в свою чергу, дозволяє нейромережі навчатися і робити більш точні прогнози.

1.5.3 Навчання нейронної мережі

Процедура навчання нейронної мережі включає в себе реалізацію змін в параметрах нейронів для досягнення визначеної цілі в завданні. Зазвичай для цього застосовується навчання «з учителем»: системі надаються дані разом зі знанням очікуваних відповідей. Це дозволяє поступово адаптувати мережу до роботи з реальними даними, де правильні відповіді не завжди відомі наперед. Суть завдання полягає в тому, щоб мережа визначила загальний принцип, який перетворює вхідні дані на вихідні.

З метою підвищення ефективності мережі виконується процес зменшення певної неточності, яка визначає різницю між передбаченим та реальними виходами. Тоді, прогнозований вихід $(f(x))$ мережі знаходиться в межах від $f(x) - e$ до $f(x) + e$.

Поширений спосіб покращення навчання мережі – метод зворотнього розповсюдження помилки. Цей метод передбачає передачу сигналу про помилку у зворотньому напряму через мережу для регулювання ваг та поліпшення точності її роботи. Метод діє ітеративно з метою поетапної зведення до мінімума помилки, що відомо як епохи навчання мережі. Розглянемо поетапний алгоритм методу, де вихідні ваги мережі, як правило, встановлюються на випадкових, але невеликих значеннях.

Під час кожної ітерації надаємо вхідні дані x_i і отримуємо відповідні виходи y_i. Помилка обчислюється за допомогою формули 1.4:

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - t_i)^2 \tag{1.4}$$

де t_i – очікувані значення;

y_i – виходи мережі;

x_i – вхідні дані;

η – швидкість навчання.

За отриманням виходу мережі на кожній ітерації відбувається регулювання ваг. З цією ціллю визначається (формула 1.5):

$$h_{L2} = \frac{h}{\sqrt{|h|_2^2 + e^2}}, h_{L1} = \frac{h}{|h|_1 + e}, \sqrt[h]{L_1} = \sqrt{h_{L_1}}, \tag{1.5}$$

$$\delta_i = y_i(1-y_i)(t_i - y_i)$$

де δ_i – градієнт (похідна) ваги i у вхідному (прихованому) шарі;

y_i – вихідне значення активації нейрона i у цьому шарі;

t_i – очікуване (цільове) значення вихідного нейрона i.

Щодо вузлів внутрішнього шару (формула 1.6):

$$\delta_i = y_i(1-y_i)\sum_{j \in I(j)} \delta_j \omega_{i,j} \tag{1.6}$$

де δ_j – градієнт ваги j у вихідному шарі;

$\omega_{i,y}$ – вага.

Ваги актуалізуються відповідно до формули1.7:

$$\omega_{i,j} = \omega_{i,j} + \Delta\omega_{i,j} \tag{1.7}$$

Вага $\omega_{i,y}$ коригується пропорційно до добутку швидкості навчання, градієнта помилки та виходу нейрона з попереднього шару. Знак мінус вказує, що ваги оновлюються в напрямку, що зменшує помилку (формула 1.8).

$$\Delta\omega_{i,j} = -\eta\delta_j x_i \tag{1.8}$$

де $\Delta\ \omega_{i,\ y}$ – зміна ваги ω між нейроном i у попередньому шарі та нейроном y у поточному шарі;

η – швидкість навчання (learning rate), яка визначає, наскільки великою буде зміна ваги в кожному кроці;

δ_j – похибка (градієнт помилки) для нейрона j у поточному шарі. Це значення вказує, наскільки поточний вихід нейрона відрізняється від очікуваного значення;

x_i – вихід (або активація) нейрона i у попередньому шарі.

Вищезазначені дії здійснюються підряд при кожній епосі, доки не буде досягнено потрібної точності, встановленого ліміту епох або ж доки помилка не перестане змінюватись значущим чином, що свідчить про те, що мережа практично зупинила свій процес «навчання».

Інна стратегія навчання нейронних мереж – це метод перенесеного навчання (transfer learning), де застосовується заздалегідь навчена модель щодо розв'язання наступної задачі. Цей метод може включати як навчання всієї мережі з нуля, так і застосування готової мережі, яку донавчають реалізовувати певну задачу. В той же час заздалегідь навчена база залишається незмінною, навчаються тільки приєднані шари, що дозволяє економити час і ресурси. Більш того, transfer learning сприяє акумулюванню знань: застосовуються знання, набуті попередньо навченою моделлю, та здобуваються нові знання під час донавчання.

Важливо відзначити, що під час навчання мережі можуть виникати труднощі недостатнього навчанням або перенавчанням. Недонавчання стає проблемою, якщо мережа не може відтворити основні закономірності, що присутні у даних. Точність на валідаційном наборі даних відчутно вища, ніж на тренувальном. Перенавчання мережі відбувається, коли вона занадто точно підлаштовується під конкретний набір навчальних прикладів, що призводить до втрати здатності до узагальнення [13]. Ця складність може з’явитися через тривале навчання мережі або недостатній обсяг навчальних зразків. Також можливі проблеми недо- або перенавчання по причині невірно вибраної архітектури мережі.

Нейромережі, що є надто простими, не в змозі відповідно уявляти залежності у реальних завданнях, тоді як дуже складні мережі мають надлишкову кількість параметрів, що в результаті може призвести до навчання не лише на встановлення залежностей, так і на віднову зашумленості.

1.5.4 Згорткові нейронні мережі

Використання багатошарового персептрону для розв'язання завдань детектування об'єктів на зображеннях має деякі обмеження. Основна проблема полягає в тому, що зображення на вході, як правило, мають великі розмірності, що призводить до значного збільшення кількості нейронів та синаптичних зв’язків. Це в свою чергу призводить до зростання потреби у великій навчальній вибірці, збільшення часу, необхідного для навчання, а також до збільшення обчислювальної складності процесу навчання. Зважаючи на це, для обробки зображень, загалом, використовуються згорткові нейронні мережі (ЗНМ), які є одним з типів штучних нейромереж і використовують операцію згортки для обробки зображень.

Цей вид нейронних мереж був створений на основі вивчення зорової кори головного мозку з уважністю до її деталей [15]. Індивідуальні нейрони реагують лише на стимули в обмеженій області поля зору, відомій як локальне рецептивне поле. Такі рецептивні поля утворюють набір, що перекривається, охоплюючи весь візуальний простір. У ЗНМ подібні рецептивні поля забезпечують локальну двовимірну взаємодію між нейронами.

Характеристики згорткових нейромереж:

- у згорткових мережах взаємодія між ядром і вхідними даними зазвичай є розрідженою, оскільки обираються лише більш вагомі пікселі вхідного зображення;
- використання одного параметру в декількох функціях моделі, у звичайних мережах будь-який зв'язок має свою неповторну вагу, а у згорткових мережах октемий елемент ядра застосовується до окремого елементу входу;

– можливість використання входів з різними розмірами;

– стійкість до зсуву.

Обмежувальний фактор згорткових мереж полягає в тому, що на будь-який нейрон вхідний сигнал поступає від обмеженої території рецепторів у попередньому шарі, тому вони витягують локальні ознаки. Коли ознака вже виявлена, її конкретне положення вважається не настільки важливим, оскільки воно встановлюється відносно інших ознак приблизно.

Згорткові нейромережі включають три основны типи шарів: шари згортки, шари пулінгу (або субдискретизації), повнозв’язні шари (рис. 1.11).

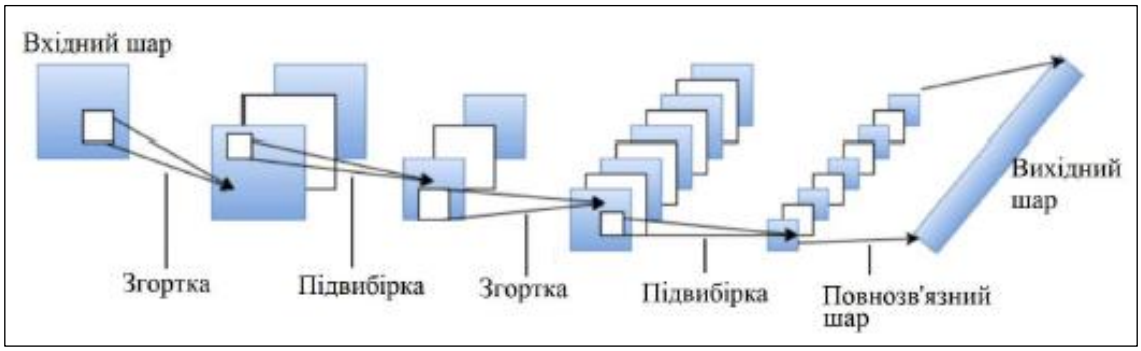


Рисунок 1.11 – Стандартна структура загорткової нейромережі

1.5.5 Архітектура згорткової нейромережі

Згорткові шари використовують операцію згортки для обробки вхідних даних в мережі. Згортка (або конволюція) може бути описана як операція, що застосовується до двох функцій, що мають один аргумент, і є специфічною формою лінійного перетворення цих функцій.

Інше представлення – операція, що плавно переміщує одну функцію вздовж другої і обчислюється як інтеграл від їх поточкового множення. У інших термінах, згортка здійснює локальне перетворення функції, враховуючи її взаємодію з іншими функціями (формула 1.9).

$$s(t) = \int u(a) \cdot w(t-a)da = (u \cdot w)(t), \tag{1.9}$$

де u – вхід;

w – є ядром згортки.

Мета операції згортки полягає у зменшенні розмірів карти ознак таким чином, щоб мережа прямого поширення, як багатошаровий персептрон, могла діяти із повним набіром ознак. У розглянутому прикладі, вхідне зображення представлене як матриця в розмірі 3×4, а ядро – розмірністю 2×2. Зображення розбивається на вікна розмірності, як ядро, і для кожного вікна проводиться операція згортки. Ця операція полягає у витягуванні векторів з кожного вікна та ядра, а потім їхньому покомпонентному перемноженні. Ці кроки повторюються для кожного вікна зображення.

Фільтри є матрицями ваг зв'язків між нейронами попереднього шару і нейронами шару згортки. Кожен фільтр виділяє певну ознаку на зображенні, таку як прямі лінії під певним кутом. У момент навчання мережі ваги фільтрів змінюються, щоб вони краще визначали ознаки на зображенні. Як правило застосовуються квадратні фільтри, а кількість таких фільтрів встановлює глибину згорткового шару. Кожен фільтр невеликий за розміром, наприклад, 3x3 чи більш. Він переміщається по зображенню з певним інтервалом, приміром, 1 піксель чи більш (рис. 1.12). На кожному етапі фільтр обробляє ту частину зображення, яка розташована під ним.

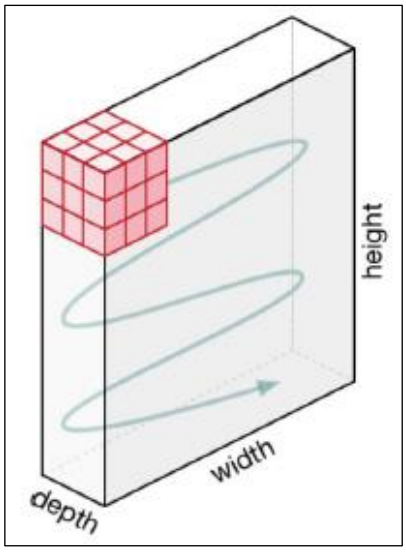


Рисунок 1.12 – Переміщення фільтра по зображенню з 3-ма колірними каналами [16]

Результати згортки, які визначають місце розташування ознак на вихідному зображенні, називаються картами ознак. Карти ознак мають форму площини, на якій всі нейрони використовують одну і ту саму множину ваг.

Додаткові параметри згорткового шару включають зміщення (stride), яке визначає відстань між рецептивними полями, та додавання нулів (padding) для управління розміром виходу. Способи доповнення нулями включають коректну, конгруентну та повну згортки, кожна з яких має свої особливості та вплив на вихідні сигнали.

Рахунок виходу нейрона шару згортки проводиться шляхом обчислення суми зважених значень вхідних даних, які проходять через функцію активації. Це включає в себе згортку вхідного зображення з ядром згортки, додавання зсуву (bias), та застосування функції активації до результату. У кінцевому результаті отримується значення, яке вказує на активацію нейрона в згортковому шарі (формула 1.10).

$$z_{i,j,k} = b_k + \sum_{u=0}^{a-1} \sum_{v=0}^{b-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k}, \tag{1.10}$$

де $i' = i \cdot s_h + u$; $j' = j \cdot s_w + v$;

$z_{i,j,k}$ – значення (активація) нейрона в положенні (i, j)(i, j) в шарі k;

b_k – зсув (bias) для нейронів у шарі k;

потрійний підсумок виконується за всіма вхідними значеннями x і вагами w;

u, v, k' – індекси, що використовуються для ітерацій по просторових розмірах і глибині вхідних даних;

a, b, f – розміри ядра згортки (kernel size) по відповідних вимірах;

$x_{i-u,j-v,k'}$ – значення вхідних даних у позиції $(i-u, j-v)$ для каналу k';

$w_{u,v,k',k}$ – значення ваги для ядра згортки в позиції (u,v) для каналу k' та шару k.

У згорткових мережах для формування вихідного сигналу нейронів згорткового шару використовуються різні функції активації, такі як ReLU, сігмоїдна та гіперболічний тангенс (див. п. 1.5.2). Шар пулінгу (або

субдискретизації) використовується для зменшення розмірності зображення, що допомагає знизити складність мережі та запобігти перенавчанню.

Шар пулінгу перетворює сигнали згорткового шару, виділяючи найбільш значущі за певними критеріями області. Узагальнені ознаки зберігають значущу інформацію, хоча втрачається частина даних про їхнє місцерозташування, але зменшується розмірність зображення.

Для створення сигналу у шарі пулінгу застосовуються два основних підходи: максимальне значення серед сигналів попереднього шару (maxpooling) або розрахунок середнього значення (averagepooling). Обраний підхід залежить від конкретної задачі та характеру даних. Процес утворення сигналів у шарі субдискретизації зображено на рисунку 1.13.

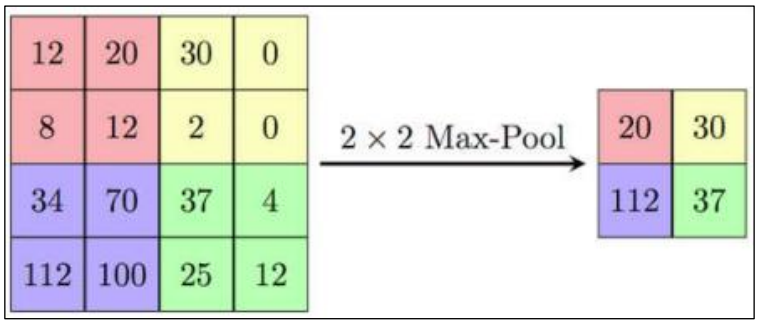


Рисунок 1.13 – Операція підвибірки (Max Pooling)

Додатково до максимального та середнього значення, використовується проміжна варіація, при якій максимум обирається не за всім вікном, а за деякою довільною підмножиною. Це може бути виконано, наприклад, за допомогою L_2-норми в прямокутному оточені або використовуючи середнє значення з вагами, які залежать від дистанції к центральному пікселю.

Повнозв'язний шар відповідає за формування виходу нейромережі. У цьому шарі кожен нейрон наступного ярусу з'єднаний із кожним нейроном попереднього ярусу на усіх ярусах, якщо присутній параметр глибини. Його основна роль полягає в перетворенні сигналів, отриманих на попередніх рівнях згорткових шарів мережі, у одновимірне поданні та виявлення ознак на одновимірному рівні. У випадку завдання класифікації повнозв'язний шар визначає, до якого класу належить вхідне зображення.

1.5.6 Алгоритм роботи мережі

Алгоритм роботи згорткової мережі можна описати наступним чином.

Вхідні дані містять інформацію про зображення, такі як ширина 32 пікселів, висота 32 пікселів та 3 кольорові канали (R, G, B).

Згортковий шар використовує фільтри для перемноження значень фільтра на значення пікселів вхідного зображення (множення поелементне). Далі йде сумування всі отриманих добутків.

Шар субдискретизації зменшує розмір до 16×16×12. Він використовує фільтр для об'єднання інформації з входу, і його основна відмінність від згорткового шару – відсутність ваг. Натомість, ядро використовує функцію агрегації, яка заповнює вихідний масив. Існують два види пулінгу: максимальний (обирається піксель з найбільшим значенням) і середній (обчислюється середнє значення).

Повнозв'язний шар генерує N-мірний вектор для N класів, аналізуючи вихід попереднього шару (карту ознак), і встановлюючи показники, які є найбільш характерними для кожного класу.

1.5.7 Аналіз типових архітектур загорткових нейромереж

LeNet, зокрема LeNet5, є простою архітектурою згорткової нейронної мережі, яка була вперше розроблена Яном ЛеКаном та іншими для розпізнавання простих рукописних символів, букв і цифр (рис. 1.14).

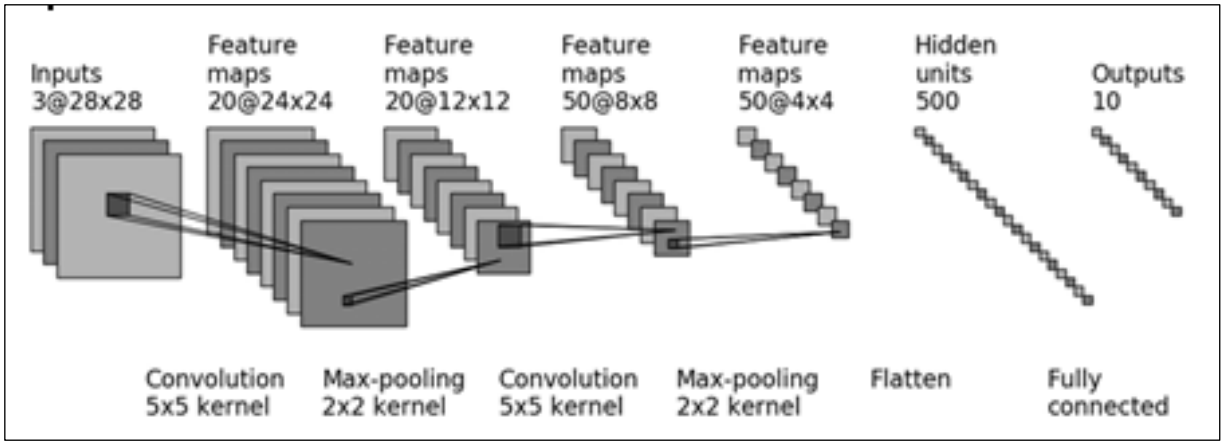


Рисунок 1.14 – Архітектура LeNet5

Ця класична мережа містить п'ять шарів, що навчаються – три з них згорткові (карти ознак (feature map) і два повнозв'язні. Звідси й така назва LeNet5. Вхідне зображення збільшується нулями до формату 32×32 та нормалізується. У шарах пулінгу використовується метод об'єднання за середнім значенням, де кожен нейрон розраховую середнє значення вхідних даних, помножається на коефіцієнт, додає до нього зміщення, застосовує функцію активації та інші відповідні операції.

Вихідний шар LeNet 5 користується квадратом евклідової відстані для порівняння вектора входів з вектором ваг. Замість множення вхідних значень на вагові коефіцієнти, нейрон рахує квадрат відстані між вектором входу і ваговим вектором. У якості функції активації використовується перехресна ентропія, тому що вона сприяє швидкому збіженню до мінімуму, породжуючи великі градієнти.

Розвитком ідеї, що лежить в основі LeNet, стала мережа AlexNet, яку назвали ім’ям Алекса Крижевського, який розробив її в 2012 році. Мережа стала популярною завдяки значній перевазі у точності на змаганнях по розпізнаванню на наборі даних ImageNet (датасет, що містить понад 14 мільйонів зображень, які належать до 1000 класів) , отримавши на 10,8% кращий результат, ніж її конкуренти. Модель AlexNet містить вісім шарів, в тому числі – п'ять згорткових, причому лише деякі з цих згорткових шарів мають слідом після себе шар агрегації (рис. 1.15).

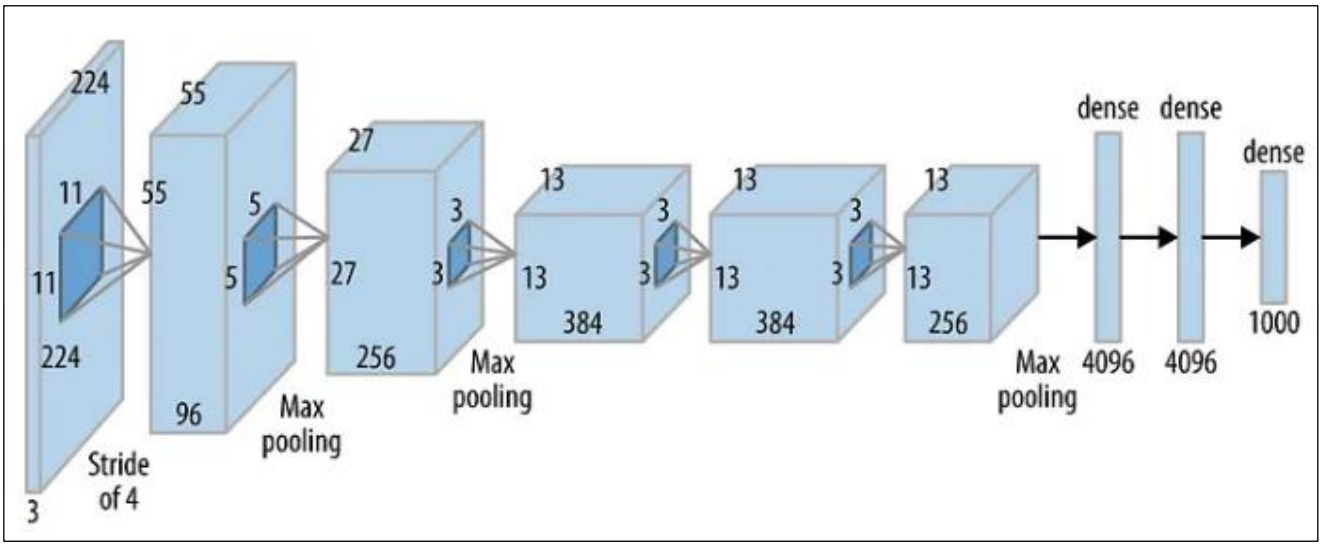


Рисунок 1.15 – Архітектура AlexNet

Одна з важливих переваг AlexNet – це використання функції активації ReLU, яка продемонструвала покращені показники під час навчання та впоралася з труднощами, що виникали при застосуванні традиційних функцій активації, таких як сигмоїда та гіперболічний тангенс. Ці проблеми включали обчислювальну складність, втрату сили активації під час глибокого навчання та недостатню нелінійність на великому діапазоні значень, що ускладнювало побудову ефективних глибоких мереж. Тож за такі переваги оригінальна публікація AlexNet вважається однією з ключових подій в історії еволюції згорткових нейронних мереж, оскільки вона викликала з'яву численних глибоких архітектур [17].

GoogLeNet – це глибока згорткова нейронна мережа, розроблена командою дослідників з Google у 2014 році для класифікації зображень. «Le» у назві посилається на нейронну мережу LeNet 5. Основна особливість GoogLeNet полягає в тому, що вона використовує спеціальний модуль первинної обробки даних, названий Inception [18]. Цей модуль застосовує згортки з різними розмірами ядра (1×1, 3×3, 5×5) одночасно, після цього об'єднує вектори ознак (рис. 1.16). Такий підхід дозволяє краще виявляти як локальні, так і глобальні ознаки зображення.

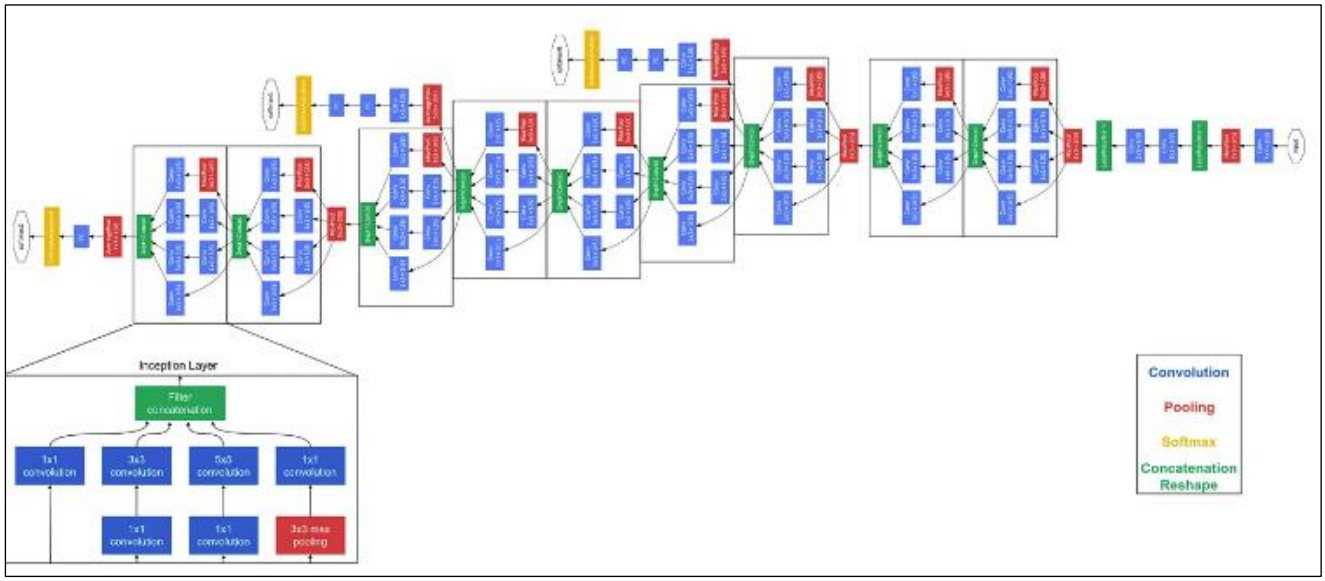


Рисунок 1.16 – Архітектура GoogLeNet та модуля inception

Незважаючи на складну конструкцію мережі, що складається з 22 шарів, кількість налаштовуваних параметрів у GoogLeNet лишається невеликою завдяки застосуванню згортки з розміром ядра 1×1, яка власне діє як простий фільтр і допомагає зменшити кількість параметрів у наступних шарах мережі. Таким чином, GoogLeNet виявляє менші вимоги до обсягу пам'яті на відеокарті, порівняно з архітектурами, такими як AlexNet, які не використовують модуль Inception.

Завдяки збалансованості GoogLeNet демонструє велику точність класифікації зображень різних розмірів. У 2014 році вона стала переможцем у змаганні ImageNet. З того часу більш сучасні нейромережі розробляються на її базі, використовуючи глибоку згорткову архітектуру та ідею модуля Inception.

Моделі загорткових нейромереж типу VGG розроблені в 2013 році для розв'язання завдань виявлення ознак на зображеннях дослідниками групи Оксфордського університету К. Сімоняном та А. Зіссерманом [19]. (Абревіатура VGG – скорочення назви групи Visual Geometry Group). Ця модель показала один із найкращих на той момент результат у точності розпізнавання об'єктів на наборі даних ImageNet. VGGNet універсальна і пропонує різні конфігурації в залежності від їхньої глибини, найбільш популярними з яких є VGG16 і VGG19. VGG16 складається з 16 вагових шарів, включаючи 13 згорткових шарів і три повнозв'язкові шари (рис. 1.17). VGG19 розширює цю архітектуру до 19 рівнів, покращуючи її здатність захоплювати складні шаблони.

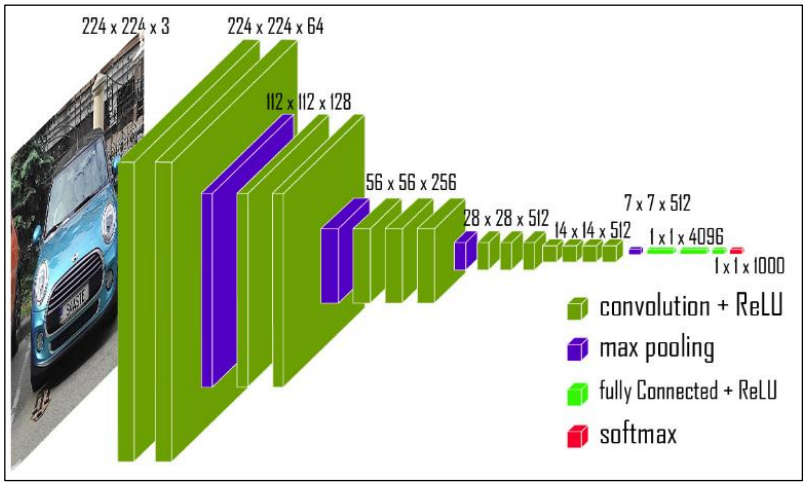


Рисунок 1.17 – Архітектура мережі VGG-16

Поглиблення моделі привело до підвищення точності порівняно з попередньою версією. Обидві мережі використовують функцію активації ReLU та softmax після повнозв’язних шарів для класифікації. Незважаючи на покращення точності, архітектури VGG мають недоліки, такі як обмежене використання пам'яті та низька швидкість, що призводить до переваги у виборі менших архітектур у деяких практичних задачах.

З'яву «залишкових» загорткових нейромереж (ResNet – Residual Networks) можна вважати значним проривом у класифікації зображень. Основним мотивом виникнення побідних мереж було те, що елементарне збільшення кількості шарів згортки в мережі швидко приводило до насичення, коли додаткові шари призводили до того, що точність мережі переставала зростати, а в деяких випадках навіть починала різко знижуватися. Введенням «залишкових» блоків мережа може ефективно навчитися «залишати» важливі інформаційні ознаки, що дозволяє збільшувати глибину мережі без негативного впливу на точність.

Модель ResNet була побудована на основі архітектури VGG, але вона більш швидкісна порівняно із VGG, при цьому менш складна в обчисленні і вимагає менш ресурсів. Це робить її більш придатною для практичних завдань, зокрема для розпізнавання в режимі реального часу.

У мережі ResNet, подібно до мережі VGG, застосовуються фільтри згортки розміром 3×3, і число цих фільтрів зростає при зменшенні розмірів вхідного зображення. Проте, головна особливість – в використанні з'єднань «shortcut connections» таких сценаріях. Коли розміри вхідного та вихідного шарів однакові, це з'єднання використовується прямо, по формулі F(x) + x (рис 1.18).

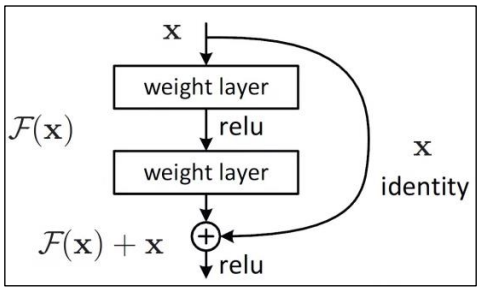


Рисунок 1.18 – Структура з'єднання «shortcut connections» та отримання результату такого з'єднання

У випадку ResNet 34, коли розмірності міняються, ідентифікатори шарів відповідно кооперують із нулями, які приєднуються для підвищення розмірності. Але для моделі ResNet 50 і вище застосовується проєкція з'єднання за допомогою згорток розміром 1×1 з обох сторін. Це означає використання блока із трьох шарів, включаючи згортки розміром 1× 1, 3×3 та ще одну згортку розміром 1×1.

Агрегаційні шари застосовуються тільки перед та після всіх шарів згортки. На початку вони реалізують максимальну агрегацію, а перед останнім повнозв'язним шаром для класифікації використовується усереднювальна агрегація. Рисунок 1.19 демонструє порівняння архітектур звичайної мережі VGG-19, базової 34-шарової моделі та реальної архітектури ResNet 34 з використанням з'єднань «shortcut connections».

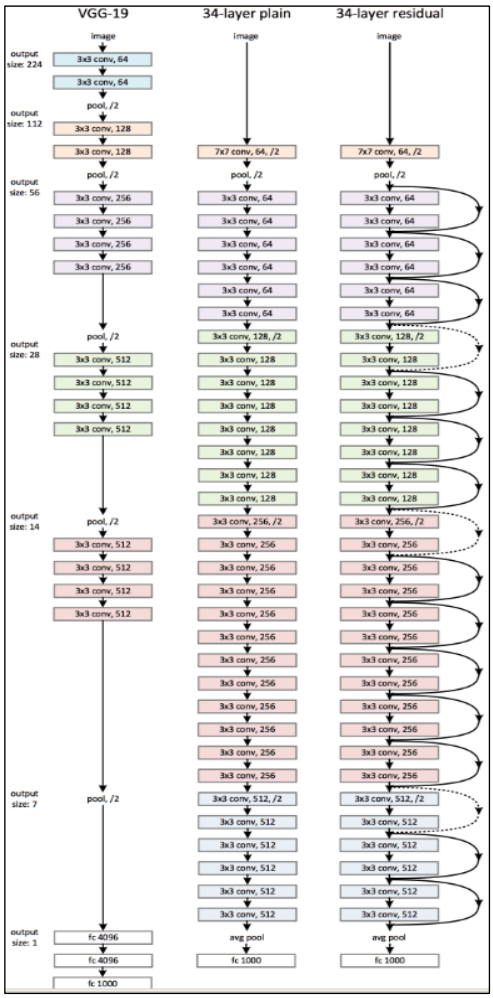


Рисунок 1.19 – Модель VGG-19 як стандарт (зліва), базова мережа з 34 шарів (у середині), ResNet з 34 шарами (праворуч). Пунктирні швидкі з'єднання збільшують розмірність

Згадані класичні методи, які використовувалися 15–20 років тому, забезпечували результати, які було б несправедливо не порівняти з досягненнями сучасних нейронних мереж за якістю. Однак, вони відставали за своєю «технологічністю» в порівнянні із сучасними методами розпізнавання об’єктів. Отже, хоча метод Віоли-Джонса досяг значного успіху у завданні розпізнавання об'єктів, його навчання відбувається повільніше відносно інших алгоритмів, тому що вимагає великої кількості ознак. HOG використовує локальні комірками, що робить його інваріантним до геометричних та фотометричних спотворень об'єкта. Класичні методи виявлення об'єктів найкраще підходять для сценаріїв, де доступний обмежений обсяг тренувальних даних. У той час як для обробки великих обсягів даних, зокрема датасетів з більш ніж 1000 зображень, ефективніше застосовувати нейронні мережі, які, імітуючи роботу людського мозку, краще впораються з завданнями обробки зображень.

Переваги нейронних мереж у виявленні об'єктів полягають у наступному:

Паралельна обробка інформації: Нейронні мережі можуть паралельно обробляти інформацію у всіх своїх шарах, що дозволяє швидше і ефективніше виконувати завдання виявлення об'єктів, зменшуючи час обробки та ресурси.

Спроможність до навчання і узагальнення: Нейронні мережі можуть навчатися на великій кількості даних та узагальнювати отримані знання, що дозволяє їм адаптуватися до різних умов та сценаріїв виявлення об'єктів.

Аналіз різних частин зображення: Нейронні мережі, зокрема згорткові, можуть аналізувати різні частини зображення та їх взаємозв'язки, що дозволяє їм ефективно виявляти об'єкти та ознаки в складних візуальних даних, наприклад, на зображеннях.

Зважаючи на це, для розв'язання завдання детектування об'єктів було застосовано згорткові нейромережі.

2 АНАЛІЗ СУЧАСНИХ ГЛИБОКИХ ЗГОРТКОВИХ МЕРЕЖ У КОНТЕКСТІ ДЕТЕКЦІЇ ОБ'ЄКТІВ

2.1 Сучасні детектори об'єктів

На сучасний момент існує широкий спектр нейромереж для пошуку об'єктів на зображенні та відеокадрах, які використовують два основних підходи: одноетапне та двоетапне виявлення. Мережі з двоетапним виявленням спочатку ідентифікують області (RoI – регіони інтересів), де можуть знаходитися об'єкти, а потім здійснюють пошук цих об'єктів у визначених регіонах. Ці мережі часто відзначаються високою точністю детектування. З іншого боку, мережі з одноетапним виявленням одночасно визначають місцезнаходження і типи об'єктів на зображенні в рамках одного обчислювального циклу, що забезпечує швидке виконання цієї задачі.

До двоетапних детекторів належать мережа R-CNN (Region-based Convolutional Neural Networks) та її варіації (FastR-CNN, FasterR-CNN, MaskR-CNN, MeshR-CNN), а також RepPoints [20], яка не використовує якорі. У той час до одностадійних детекторів, що використовують якорі, відносяться: RetinaNet, SSD і YOLO. Такі мережі як CornerNet, CenterNet, та FCOS не використовують якорі.

Для вирішення задачі objeckt detection у реальному часі двоступеневі детектори R-CNN і FastR-CNN не є оптимальним вибором. Тільки FasterR-CNN стала першою моделлю, яка практично досягла роботи в режимі реального часу, здатної обробляти 17 кадрів на секунду. Отже, для порівняння в цьому дослідженні використовуються популярні одноступеневі детектори для розпізнавання в реальному часі YOLO, SSD, RetinaNet та двоступеневий FasterR-CNN.

2.1.1 Двоступеневий детектор FasterR-CNN

Відмінність традиційних завдань класифікації зображень від задач детекції складається з того, що в алгоритмах пошуку об’єктів пробують виокремити деяке візуальне обрамлення, що охоплює об’єкт, щоб знайти його на зображенні.

Об'єкти можуть бути в різній кількості: від одного до кількох, та можуть належати до різних класів. Зазвичай такі завдання вирішуються для ситуацій реального часу. Тому важливо не лише забезпечити точність мереж, які можуть виявляти, класифікувати та маркувати об’єкти зображень, також їхню швидкість, простоту та ефективність використання в умовах, коли обмежені обчислювальні ресурси.

Прикладом мереж, які спрямовані на прискорення завдання детекції, є мережі R-CNN. Такі моделі призначені для розв’язання задач детекції великої кількості областей інтересу (region of interest, RoI) на зображенні одночасно. Для того, щоб шукати велике число потенційних регіонів, пропонується вибирати тільки деяку кількість, приміром, 2000 регіонів (регіонів пропозицій) через селективний пошук [21].

Архітектуру R-CNN можна описати як послідовність дій: спочатку зображення надходить на вхід моделі, після чого проводиться виявлення областей інтересу в зображенні, потім вираховуються характеристики цих областей за допомогою згорткової нейронної мережі (візьмемо, AlexNet), і нарешті, проводиться класифікація знайдених областей з використанням попередньо навченої моделі (рис. 2.1).

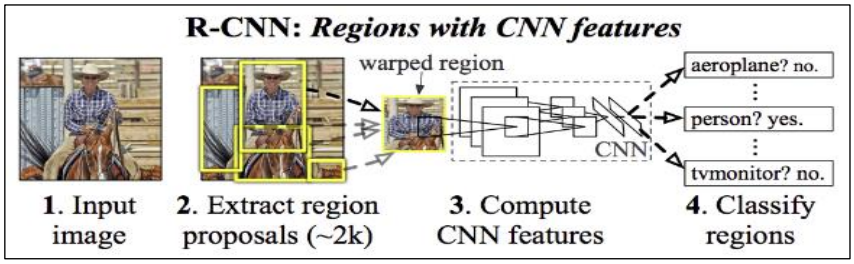


Рисунок 2.1 – Процес визначення та аналізу областей інтересу в R-CNN

Звичайно, такий метод поки має обмеження для використання в режимі реальному часі з причини потреби виявлення і класифікації великого числа регіонів. До певної міри цю задачу вдалося розв’язати за допомогою методу, впровадженого у FastR-CNN, розробленої у 2015 році [22]. У цій новій архітектурі оброблювання зображення розпочинається з введення його до шарів згортки з метою виявлення ключових ознак, що дозволяє виконати цей процес одноразово

для одного зображення, а не для кожного індивідуального регіону інтересу. Потім застосовується шар агрегації RoI pooling (Region of Interest pooling), який передає ці регіони на повнозв'язні шари для класифікації. Метод обробки регіонів в мережі FastR-CNN представлено на рисунку 2.2.

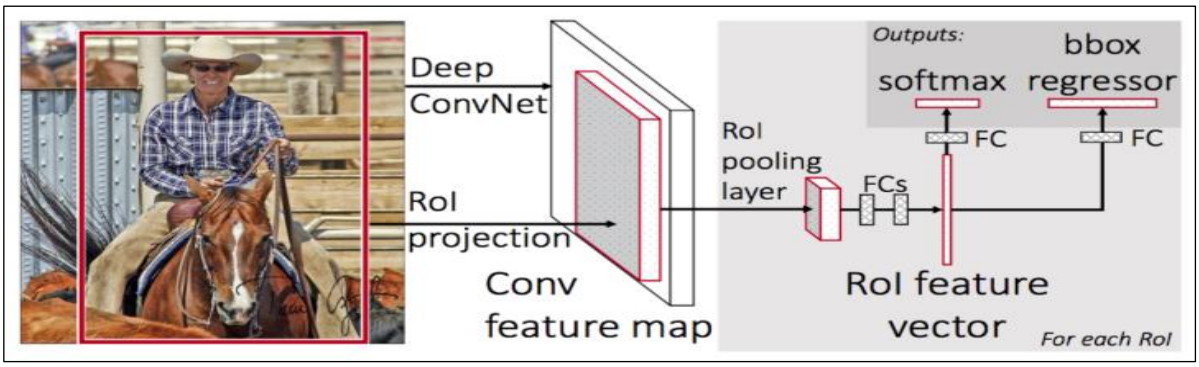


Рисунок 2.2 – Приклад аналізу зображення за допомогою FastR-CNN

FastR-CNN виявляється більш точним та швидким у порівнянні з R-CNN, оскільки до згорткового шару додаються не всі регіони інтересів. Модернізацією FastR-CNN є мережа FasterR-CNN, в якій впроваджено новий метод локалізації об'єкта за допомогою Region Proposal Network (RPN). Цей підхід базується на використанні системи якорів (anchor boxes), до цієї комбінації входять: центр ковзаючого вікна, масштаб, співвідношення сторін. Процес визначення та класифікації регіонів пропозицій в Faster R-CNN представлен на рисунку 2.3:

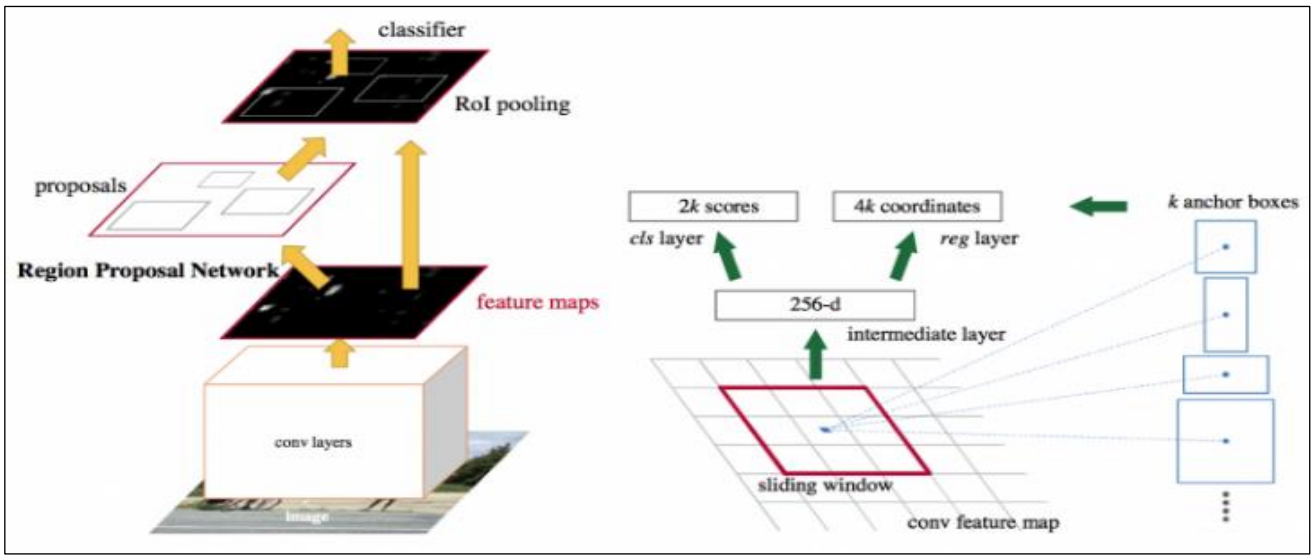


Рисунок 2.3 – Архітектура FasterR-CNN та RPN

Для кожного вхідного зображення створюється карта ознак, яка подальшим чином оброблюється шаром RPN (рис. 2.3). Ковзаюче вікно переміщується по карті ознак, і його центр зв'язаний з центром якоря. Застосовується метрика IoU (Intersection over Union, друга назва –індекс Жаккара) для визначення ступеня перетину областей (як у FasterR-CNN, де ці області є якорями) і порівняння схожості серед двох об'єктів. В контексті завдання object detection метрика IoU визначається як відношення площі перетину 2-х обмежувальних рамок до їх загальної площі. За допомогою метрики та прямокутних рамок вирішується, чи присутній об'єкт у вказаній області зображення. Після цього карта ознак з виявленими об'єктами, надсилається до шару RoI для наступної класифікації та визначення зсуву місцезнаходження можливих об'єктів.

FasterR-CNN взагалі менш успішна у точності локалізації об'єктів, проте її робота відбувається швидше, ніж у FastR-CNN. Для вирішення задач сегментації зображень використовується її модифікація – мережа MaskR-CNN.

2.1.2 Одноетапний детектор YOLO

Мережа R-CNN та її варіації забезпечують високу точності при детекції об'єктів на зображеннях (наприклад, FastR-CNN досягає середнього показника точності у 66% на наборі даних PASCAL VOC 2012 [22]). Однак ці моделі мають невелику швидкість роботи, і лише FasterR-CNN може бути використана в режимі реального часу. Мережі, які оперували шляхом виділення областей інтересу на зображенні, поступово збільшували свою швидкість та підходили до можливостей реального часу для розпізнавання та детекції. Проте, досі стикалися з певними обмеженнями у швидкості, головним чином через необхідність виділення, аналізу та класифікації безлічі окремо взятих регіонів інтересу.

Одностадійні детектори, зокрема YOLO (You Only Look Once – «Дивишся лише раз»), більш швидкі, але менш точні, ніж мережі R-CNN. Основна ідея YOLO полягає в тому, що вона трактує виявлення об'єктів як задачу регресії, де просторово розділені обмежувальні рамки та ймовірності класів об'єктів

прогнозуються одночасно. Вхідне зображення проходить через нейронну мережу лише один раз, що дозволяє виявити об'єкти негайно.

Архітектура YOLO була вдосконалена за зразком моделі GoogLeNet для розпізнавання зображень, але вона має свої відмінності. Замість складних модулів, подібних до тих, що в GoogLeNet, в архітектурі YOLO застосовується послідовність редукційних шарів розміром 1×1, які допомагають зменшити кількість каналів або фільтрів, а потім використовуються згорткові шари розміром 3×3 для подальшої обробки зображення.

Алгоритм YOLO базується на розбитті зображення на сітку із комірок, що виступають як базові точки (так звані – якоря) для обмежувальних рамок. Для кожної комірки мережа генерує кілька обмежувальних рамок, що мають потенційно містити об'єкти. Для кожної рамки мережа виводить п'ять параметрів (рис. 2.4):

– координати центру рамки (x, y) щодо меж комірки (c_x, c_y);
– ширина і висота рамки (w, h) щодо усього зображення;
– показник упевненості (confidence), який відображає ймовірність знаходження у комірці об'єкта (формула 2.1):

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o) \tag{2.1}$$

де *Pr(object)* – ймовірність наявності об'єкта у даному кадрі;

IOU(b, object) – обчислюється як відношення площі перетину рамки детекції та реальної рамки об'єкта до площі об'єднання цих двох рамок;

$\sigma(t_0)$ – значення якості детекції об'єкта у момент часу t_0.

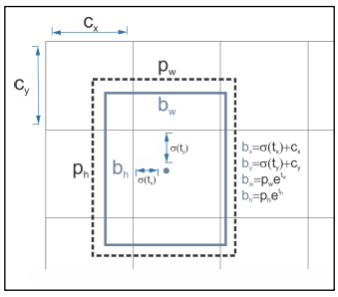


Рисунок 2.4 – Характеристики обмежувальної рамки

де t_x, t_y, t_w, t_h – те, що передбачила YOLO;

C_x,C_y – координати верхнього лівого кута, що відповідають комірці сітки;

P_w, P_h – ширина та висота конкретної обмежувальної рамки;

b_x,b_y, b_w, b_h – параметри для вказаного обмежувального прямокутника;

$\sigma(t_o)$ – це конкретна оцінка упевненості.

YOLO використовує метод немаксимального придушення (NMS), щоб відфільтрувати дублікати рамок, які мають високу імовірність класифікації об'єктів. Це означає, що лише рамки з найвищими значеннями показника упевненості залишаються, інші ігноруються (формула 2.2).

$$
\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(x_i-\hat{x}_i\right)^2+\left(y_i-\hat{y}_i\right)^2\right]\\
&+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]\\
&+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2\\
&+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2\\
&+\sum_{i=0}^{S^2}1_i^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2
\end{aligned}
\tag{2.2}
$$

де λ_{coord} – ваговий коефіцієнт для координатної втрати;

S^2 – кількість комірок сітки (наприклад, якщо сітка 7×7, то $S^2 = 49$);

B – кількість рамок (bounding boxes) на кожну комірку;

1_{ij}^{obj} – індикатор наявності об'єкта в j-тій рамці i-ї комірки;

x_i, y_i – координати центру передбаченої рамки;

w_i, h_i – ширина та висота передбаченої рамки;

$\hat{x}_i$, $\hat{y}_i$, $\hat{\omega}_i$, $\hat{h}_i$ – справжні координати центру та розміри рамки;

C_i – впевненість в тому, що рамка містить об'єкт;

$\hat{C}_i$ – справжня впевненість (1 для об'єкта, 0 для фону);

1_{ij}^{noobj} – індикатор відсутності об'єкта в j-тій рамці i-ї комірки;

λ_{noobj} – індикатор наявності об'єкта в i-тій комірці;

$p_i(c)$ – імовірність передбачення класу с для і-ї комірки;

$\hat{p}_i(c)$ – справжня ймовірність класу с (1 для правильного класу, 0 для інших).

Багатокомпонентна функція втрат у YOLO включає в себе такі складові:

- втрата класифікації: ця складова різницю між передбаченими ймовірностями наявності кожного класу об'єктів та їх справжніми значеннями для кожної комірки, у випадку, якщо об'єкт на зображенні було виявлено;
- втрата локалізації: ці дві складові враховують квадратичну похибку визначення координат та розмірів рамок відносно справжніх значень;
- втрата упевненості: ця складова враховує різницю між передбаченим значенням показника впевненості та справжнім значенням, якщо об'єкт виявлено. Якщо об'єкт не виявлено, враховується також допоміжний коефіцієнт λ_{noobj} (дорівнює 0,5), щоб компенсувати дисбаланс між класами.

YOLOv2, випущена у 2017 році, покращила точність моделі за рахунок використання мережі Darknet-19 та впровадження пакетної нормалізації [23]. Одним з важливих додатків до оригінального YOLO було додавання блоків якорів, що дозволило прогнозувати п'ять обмежувальних рамок для кожної комірки, уникаючи обмеження на один об'єкт у комірці.

YOLOv3, представлена у 2018 році, базується на мережі Darknet-53 і містить 106 згорткових шарів [24]. Вона здатна точно виявляти невеликі об'єкти на зображеннях завдяки прогнозуванню обмежувальних рамок у трьох різних масштабах. YOLOv3 використовує до дев'яти опорних рамок, розподілених між трьома шарами карт ознак. Також, на відміну від попередніх версій, YOLOv3 передбачає наявність трьох обмежувальних рамок на комірку, тільки у різних «масштабах», що дозволяє компенсувати недоліки у виявленні дрібних об'єктів.

У 2020 році була представлена модель YOLOv4, яка базується на архітектурі SPDarknet-53 та впроваджує інноваційні концепції, такі як «зважені залишкові з’єднання» (WRC), «міжстадійні часткові з’єднання» (CSP), «крос-міні-

пакетну нормалізацію» (CmBN) і «самонавчання» (SAT). У порівнянні з YOLOv3 вищої середньої точності нововведення дозволили YOLOv4 досягти на 10% та на 12% кращої продуктивності.

Версія YOLOv5 зазнала подальших оптимізацій, зокрема покращень у навчанні та швидкості виведення порівнюючи з YOLOv4. Було запроваджено різні методи аугментації даних при навчанні, такі як обрізання, обертання, масштабування. Також адаптовано функцію Focal Loss. YOLOv5 був випущений Гленном Джохером через кілька місяців після YOLOv4 у 2020 році. Він розроблений на PyTorch, а не на Darknet.

Усі ці зміни зробили YOLO дуже зручною, гнучкою та універсальною системою детекції об'єктів, яка переймала найкращі методи для одно стадійних детекторів протягом свого розвитку.

YOLO відрізняється від методів, що використовують ковзне вікно або пропозиції регіонів, оскільки він може оцінювати всі зображення цілком під час прогнозування. Це дозволяє YOLO враховувати контекстну інформацію про класи та їхній зовнішній вигляд, що сприяє його узагальненню на нові області або несподівані вхідні дані. Однак, хоча YOLO відзначається високою швидкістю ідентифікації об'єктів, він може мати проблеми з точністю локалізації деяких об'єктів, особливо маленьких.

Процес навчання YOLO може бути складним через складність його функції втрат, що враховує різні фактори, як координати та розмірність рамки, ймовірність наявності об'єкту та ймовірності його класу.

Проблеми, що виникали з YOLO, поступово вирішувалися при його модернізацій до різних версій. Моделі YOLO є найбільш популярними серед систем object detection завдяки їхній гнучкості і можливості вибору з декількох конфігурацій із різною швидкодією, що дає можливість настроювати їх для виконання різних завдань.

2.1.3 Одноетапний детектор SSD

Мережа для детекції об'єктів у режимі реального часу, яка подібна до YOLO, – SSD (Single Shot MultiBox Detector). Модифікація «Single Shot» підразуміває виконання завдання локалізації і класифікації у єдиному проході мережі (рис. 2.5). SSD відзначається тим, що він став одним з перших одноєтапних детекторів, які досягли високої точності, порівняно з двоетапними моделями, при цьому забезпечуючи можливість роботи в режимі реального часу.

SSD відзначається більшою точністю детекції невеликих об'єктів порівнюючи з іншими детекторами одного етапу, така як 77% на наборі даних Pascal VOC 2007, завдяки здатності детектувати об'єкти різних масштабів. Ця мережева архітектура особливо ефективна у використанні для відеоспостереження та виявлення ключових точок.

SSD включає у себе дві основні компоненти: заздалегідь навчену мережу для вилучення ознак та низку згорткових шарів для детекції об'єктів. Базою архітектури SSD є VGG-16, але може використовувати інші мережі як основу (див.рис.2.5).

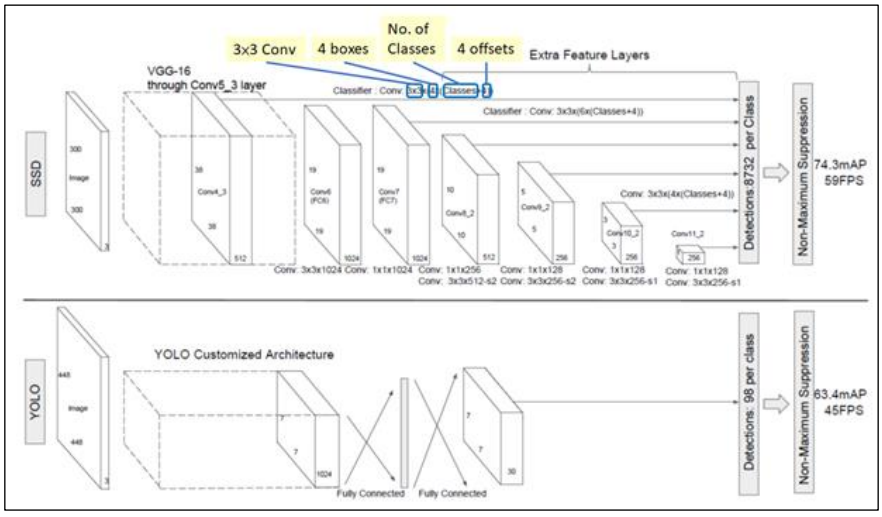


Рисунок 2.5 – Схема архітектури SSD порівняно зі схемою архітектури YOLO

VGG-16 використано за її високу ефективність у класифікації зображень, які мають високу роздільну здатності. Замість повнозв'язних шарів VGG додано допоміжні згорткові шари для витягування об'єктів у різних масштабах та

зменшення розміру вхідних даних до кожного наступного шару. Для зменшення кількості виявлених обмежувальних рамок та усунення дублікатів використовується метод немаксимального придушення.

Подібно до YOLO: зображення розбивається на сітку, і в кожному сегменті цієї сітки здійснюється розпізнавання об'єктів. SSD використовує концепцію рецептивного поля для виявлення об'єктів різних масштабов та застосовує стандартні обмежувальні рамки, подібні анкорам FasterR-CNN. Натомість кластеризації k-середніх, як у YOLO, SSD прогнозує рамки та їх надійність, використовуючи картки ознак різних масштабів.

Всього SSD використовує 8732 обмежувальні рамки за замовчуванням, порівнюючи їх з реальними рамками за розмірами, положенням і масштабом, і обирає ті, які мають щонайбільше перекриття (IoU понад 0,5).

Застосовується функція втрат, яка представляє собою комбінацію втрат локалізації (L_{loc}) та втрати впевненості в класифікації (L_{conf}), приваженої згідно їх вагам (формула 2.3):

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha \cdot L_{loc}(x,l,g)) \quad (2.3)$$

де L – загальна функція втрат;

x – вхідне зображення або об'єкт;

c – класифікація об'єктів;

l – позиція (координати) об'єкта;

g – істинні значення позиції об'єкта;

N – кількість прикладів у партії.

Існують дві версії SSD: SSD 300 та SSD 512. SSD 300 працює з зображеннями в розмірі 300×300 в якості вхідних даних. Ця версія з меншою роздільною здатністю, але її швидкодія вище. SSD300 може мати точность на рівні 74,3% mAP при швидкості 59 кадрів в секунду (FPS) на датасеті VOC 2007 [25]. Мережа SSD 512 працює з зображеннями в розмірі 512×512 в якості вхідних

даних, що забезпечує вищу роздільну здатність. Ця модель працює з більшою точністю, ніж SSD 300. Зокрема, SSD 512 досягає точності на рівні 76,9% mAP при швидкості 22 кадри в секунду (FPS). Це перевищує результати моделей YOLOv1 (63,4% mAP, FPS = 45) та FasterR-CNN (73,2% mAP, FPS = 7) [25].

2.1.4 Одностадійний детектор RetinaNet

RetinaNet – ще один одностадійний детектор, популярний для застосування в режимі реального часу. Активно застосовується для обробки аерофотознімків та зображень, отриманих з супутників [26]. Щоб уникнути негатичного впливу на точність через недостатню увагу при навчанні до відмінностей між фоном і самими об'єктами автори впровадили новий метод оцінки помилок, відомий як «фокусна втрата», яка акцентує увагу на складніших зразках під час навчання нейромережі. RetinaNet використовує архітектуру ResNet для витягування ознак та впроваджує мережу Feature Pyramid Network (FPN) для отримання ознак на різних рівнях зображення. Вона також має мережу класифікації блоків якорів за допомогою сигмоїдної активації, мережу регресії, яка перетворює ці блоки якорів на реальні об'єкти (рис. 2.6).

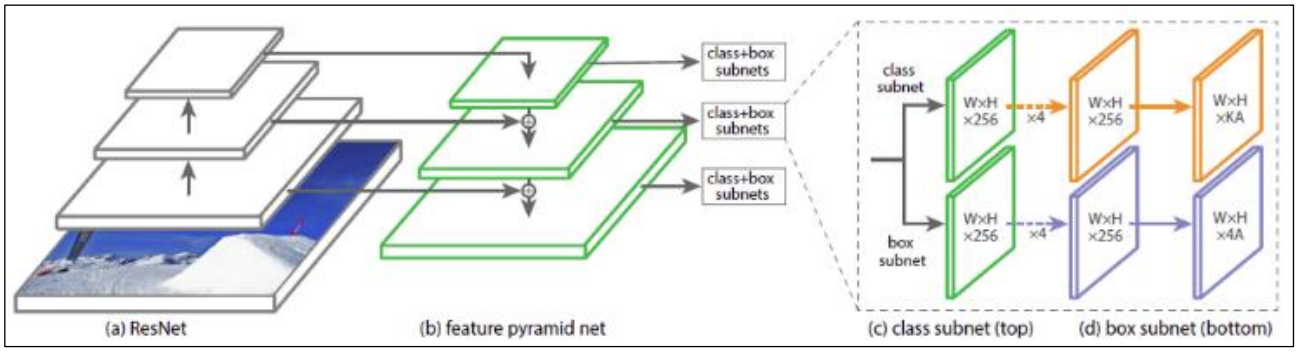


Рисунок 2.6 – Структура RetinaNet

До складу Feature Pyramid Network (FPN) входять три головні компоненти: підйомний шлях (bottom-up pathway), спусковий шлях (top-down pathway) та бічні зв'язки (lateral connections) (рис. 2.7).

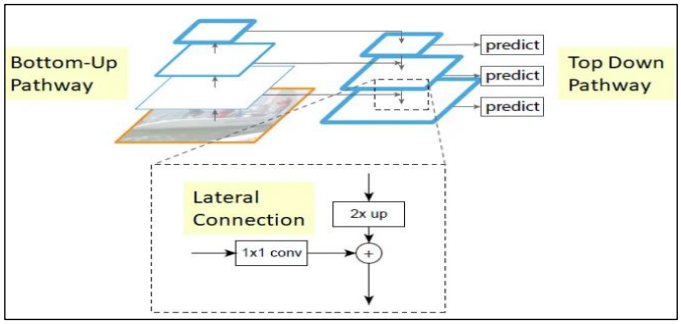


Рисунок 2.7 – Структура піраміди ознак

Підйомний шлях відображає послідовність згорткових шарів, які поступово зменшують розмір карти ознак. У цій послідовності верхні шари мають вищий ступінь семантичності, проте вони меншої роздільної здатності, ніж нижні шари. У вихідній структурі RetinaNet цю роль реалізує базова підмережа ResNet. Спусковий шлях складається з послідовності шарів, де розміри карт ознак на вищих рівнях плавно зростають до розмірів карт ознак на нижчих рівнях. Внаслідок бічних зв'язків карти ознак відповідних шарів пірамід «знизу вгору» і «зверху вниз» поелементно з'єднуються. Це злиття шарів згорткових забезпечує масштабну інваріантність архітектури RetinaNet.

Фокусна втрата (focal loss) – це перехресна ентропія, яка вирішує питання диспропорції класів, надаючи більшу вагу складним і невірно класифікованим прикладам, таким як зашумлений фон або частково перекриті об'єкти, і меншу вагу для простих прикладів, наприклад, фоновим об'єктам [26].

Додаванням коефіцієнта модуляції до перехресної ентропії реалізована функція фокусної втрати. У формулі (2.4) представлений вираз, де γ є параметром фокусування, що регулюється за допомогою перехресної перевірки. За допомогою цього коефіцієнта модуляції фокусна втрата може змінювати своє значення в залежності від значення γ:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{2.4}$$

де p_t – ймовірність, присвоєна моделлю правильному класу. Якщо об'єкт належить до позитивного класу (наприклад, об'єкт присутній), $p_t = p$; якщо об'єкт належить до негативного класу (наприклад, об'єкт відсутній), $p_t = 1 - p$;

p – ймовірність, присвоєна класу позитивного результату;

γ – параметр фокусування. Коли $\gamma = 0$, формула стає стандартною крос-ентропією.

Рисинок 2.8 демонструє, як значення фокусної втрати змінюється в залежності від значень γ. При збільшенні ймовірності коректного вибору класу коефіцієнт масштабування зменшується до нуля. Це означає, що фокусна втрата спрямована на зменшення впливу на добре класифіковані приклади і зосереджується на важких та неправильно класифікованих прикладах.

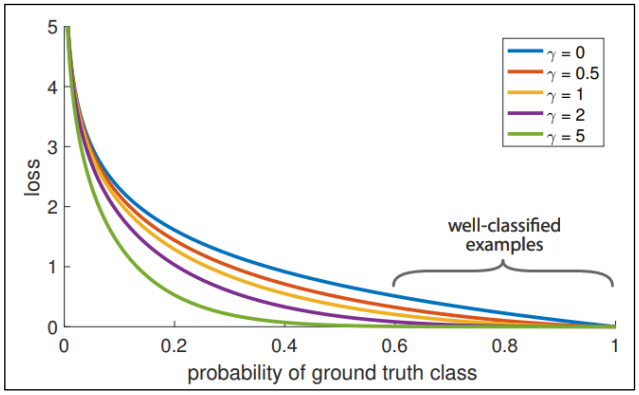


Рисунок 2.8 – Графічне зображення динаміки фокусної втрати

Особливості фокусної втрати:

- у випадку неправильно класифікованого прикладу з малим значенням його ймовірності p_t, коефіцієнт модуляції наближається до 1, і вважається що втрати немає;
- вимір фокусування γ поступово контролює темп, з яким зменшується вага легких прикладів;
- підвищення значення γ призводить до збільшення коефіцієнта модуляції. Оптимальне значення γ, виходячи з експериментальних результатів розробників, становить 2.

Використання α-збалансованого варіанту фокусної втрати (формула 2.9) в RetinaNet, де параметри α = 0,25 і γ = 2, допомогло розв’язати питання диспропорції класів.

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log (p_t) \tag{2.9}$$

де p_t – ймовірність, присвоєна правильному класу.

γ – параметр фокусування.

α – коефіцієнт балансування, що компенсує дисбаланс класів. Зазвичай:

$\alpha_t = \alpha$ – для позитивного класу.

$\alpha_t = 1 - \alpha$ – для негативного класу.

Збалансована фокальна втрата комбінує переваги фокальної втрати і збалансованої крос-ентропії, надаючи більше ваги рідкісним класам та важким для класифікації прикладам. Це робить її потужним інструментом для задач з дисбалансом класів, покращуючи здатність моделі розпізнавати рідкісні об'єкти та зменшуючи вплив легких для класифікації прикладів.

2.2 Оцінка продуктивності моделей

2.2.1 Обрання дата сету

Обрання певного датасету залежить від конкретної задачі, можливостей доступу до даних і потреб в обсязі, різноманітності сцен та точності анотацій.

Оцінка продуктивності моделей буде здійснюватися на датасете CEPDOF[27]. Набір даних CEPDOF (Challenging Events for Person Detection from Overhead Fisheye Images), розроблений у лабораторії обробки візуальної інформації (VIP) Бостонського університету та опублікований у квітні 2020 року, містить записи з камер «Риб’яче око», розташованих зверху, в одній кімнаті. Камери кругового огляду, відомі також як «омнідирекційні камери», масово застосовуються для систем відеоспостереження. Одним з їх типів і є камери із «Риб'ячим оком», які можуть охоплювати область огляду до 360˚. Ці камери

застосовуються у різних місцях, таких як термінали аеропортів, магазини роздрібної торгівлі, готелі, лікарняні заклади та офіси.

CEPDOF містить 8 відеороликів, на яких одночасно можна побачити до 13 осіб. У наборі даних більше 25 000 анотованих кадрів і кілька складних сценаріїв (приміщення переповнено, є сильні перешкоди для візуалізації тіл, різноманітні та часто незвичні пози тіл, а також камуфляж голови, такий як капюшони та капелюхи. Особливі умови включають зображення людей на проекційному екрані та обмежене освітлення, яке може бути інфрачервоним). CEPDOF анотується у просторі і часі, тобто обмежувальні прямокутники для кожної особи мають однаковий ідентифікатор в послідовності кадрів. Це також може бути корисним для додаткових завдань зору, таких як відслідковування об'єктів у відео та ідентифікація людей. Це забезпечує реалістичність та підвищує рівень складності завдань, пов'язаних із детекцією та стеженням за людьми. Опис кожного відео представлено на рисунку 2.9 в табличній формі.

Відеоряд	Сценарії	Опис/Завдання	Максимальна кількість людей	Кількість кадрів	Роздільна здатність
Lunch meeting 1	Спільна діяльність	Люди йдуть і сидять	11	1,201	2048 x 2048 x 1 Гц
Lunch meeting 2	Багатолюдна сцена	Більше 10 людей сидять і обідають	13	3000	2048 x 2048 x 10 Гц
Lunch meeting 3	Спільна діяльність	Люди йдуть і сидять	10	900	2048 x 2048 x 1 Гц
Edge cases	Крайові корпуси	Люди, що ходять і сидять, екстремальні пози тіла, камуфляж голови, сильні оклюзії тіла	8	4,201	2048 x 2048 x 10 Гц
High activity	Прогулянкова діяльність	Люди заходять через одні двері, а виходять через інші	9	7,202	1080 x 1080 x 10 Гц
All-off	Слабке освітлення	Люди ходять і сидять, верхнє освітлення вимкнене, ІЧ-фільтр камери знято, ІЧ-підсвічування відсутнє	7	3000	1080 x 1080 x 10 Гц
IRfilter	Слабке освітлення	Люди, що йдуть і сидять, верхнє освітлення вимкнене, з ІЧ-фільтром камери, без ІЧ-підсвічування	8	3000	1080 x 1080 x 10 Гц
IRil	Слабке освітлення	Люди, що йдуть і сидять, верхнє освітлення вимкнене, ІЧ-фільтр камери видалений, ІЧ-підсвічування	8	3000	1080 x 1080 x 10 Гц

Рисунок 2.9 – Особливості відеоматеріалів з набору відеоданих CEPDOF [27]

Кожне відео збережено в самостійній папці, а для кожного кадру створюється окремий файл JPEG. Анотації кожного відео розміщуються у єдиному файлі JSON, формат якого загалом відповідає стандартам датасету MS COCO, проте є деякі відмінності.

Кожна обмежувальна рамка описується 5-ма числами: [cx, cy, w, h, d], координати центру рамки (cx, cy) в пікселях відносно верхнього лівого кута зображення на [0, 0], ширина (w) і висота (h) рамки в пікселях, а також «d» – кут повороту рамки по годинниковій стрілці відносно вертикальної осі, спрямованої нагору, у градусах (від -90 до +90). З метою уникнення неоднозначності: (w<=h).

В датасете використовується лише один клас – «особа». Кожному анотованому об’єкту – людині надається «person_ID», який призначається послідовно і залишається незмінним для цієї особи у всіх послідовних кадрах. Немає надання масок сегментації.

Через обмеження в обчислювальних ресурсах використовувався підхід transfer learning, який дозволяє використовувати попередньо натреновані моделі. Для навчання використовувався відеодатасет «Lunch meeting2», для тестування – «Edge cases». Тривалість тренування для всіх моделей становила 30 епох.

2.2.2 Обрання метрик для порівняння моделей

Оцінка продуктивності алгоритмів детектування об'єктів є ключовим кроком, який дозволяє порівнювати різні моделі та визначати, яка з них найкраще відповідає конкретній задачі.

Основним і найважливішим показником оцінки ефективності моделі в задачах object detection є середня точність (mAP, Mean Average Precision). Цей показник включає в себе метрики: точність (precision) та повноту (recall) [28]. Точність показує частку правильних детекцій серед усіх детекцій, зроблених моделлю. Визначається як (формула 2.10):

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.10}$$

де *TP* (True Positives) – кількість правильних детекцій;

FP (False Positives) – кількість неправильних детекцій.

Повнота показує частку правильних детекцій серед усіх наявних об'єктів. Визначається як (формула 2.11):

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.11}$$

де *FN* (False Negatives) – кількість пропущених об'єктів.

Ці метрики визначають кількість правильно виявлених у всьому наборі обмежувальних рамок. Іншими словами, точність вимірює, наскільки відповідні результати детекції є правильними, а повнота вказує на те, наскільки ефективно модель виявляє всі відповідні об'єкти. Крім того, mAP узагальнює ці метрики, враховуючи їх середнє значення по всім класам або об'єктам, що дозволяє отримати більш комплексне уявлення про продуктивність моделі (формула 2.12):

$$mAP = \frac{1}{n}\sum_{i=1}^{N} AP_i \tag{2.12}$$

де *N* – кількість класів;

AP (Average Precision) – розраховується як середнє значення точності при різних значеннях повноти. Використовується для оцінки ефективності детектора при різних порогах впевненості.

Другий важливий показник – пропускна здатність (Throughput), вимірюється в кадрах на секунду (FPS). Він визначає кількість кадрів, які можуть бути оброблені або відтворені за одну секунду. Це метрика, яка використовується для визначення швидкості обробки або передачі даних. У контексті моделей глибокого навчання, FPS визначається як кількість кадрів, які можуть бути оброблені за одну секунду під час прогнозування моделлю. Для роботи моделі в

режимі реального часу, бажано мати значення FPS більше 24, щоб забезпечити плавну обробку даних і відповідь на запити в реальному часі.

Для моделі FasterR-CNN було необхідно змінити формат анотацій для зображень з .txt на XML. Для роботи з даними та моделлю використовувалася відкрита бібліотека PyTorch. Натренована модель, яку завантажили, була torchvision.models.detection.fasterrcnn_resnet50_fpn [29].

Після цього було проведено тренування моделі YOLOv3, яку завантажили з вказаного репозиторію [30]. Для попереднього тренування використовувалася модель SSD, яку також завантажили з репозиторію. Модель SSD була завантажена з репозиторію [31] і тренувалася з використанням відкритих бібліотек TensorFlow та Keras. Анотаційні файли з датасету були конвертовані у формат .csv. У завершенні було використано PyTorch для реалізації моделі RetinaNet, зокрема torchvision.models.detection.retinanet_resnet50_fpn [32]. Аналогічно до FasterR-CNN, ця модель вимагає анотацій у форматі .xml. Приклад оброблення відеокадрів тестової вибірки за допомогою YOLO наведено на рисунку 2.10.

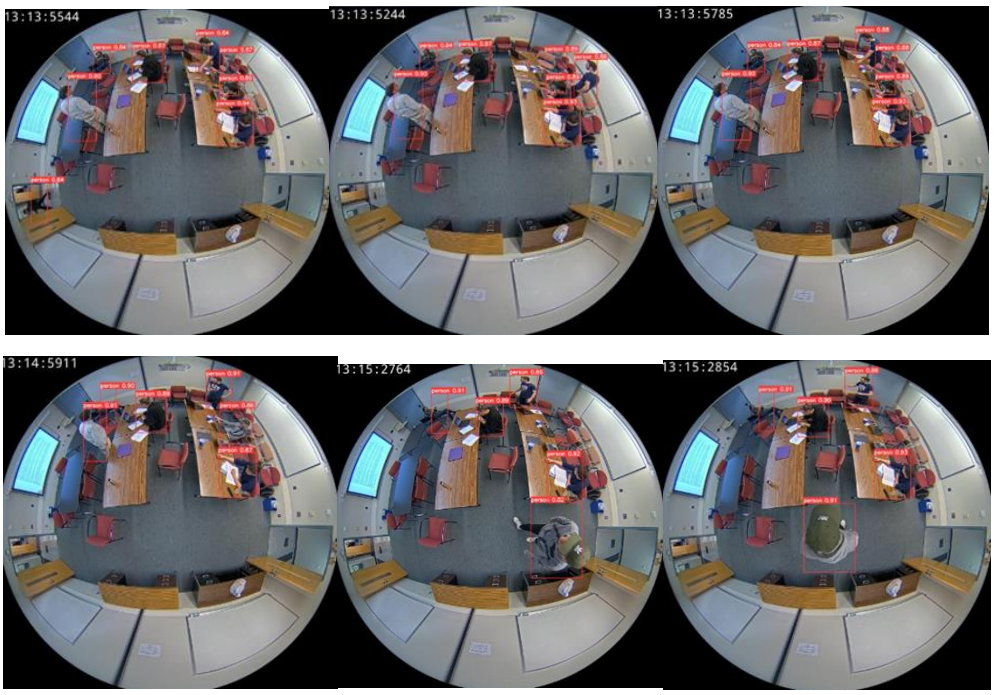


Рисунок 2.10 – Обробка відеокадрів за допомогою YOLO

Отримані результати порівняльного аналізу продуктивності детекторів представлені в таблиці 2.1.

Таблиця 2.1 – Порівняння продуктивності моделей детектування об'єктів на наборі даних CEPDOF

Показники ефективності	FasterR-CNN	YOLO	SSD	RetinaNet
mAP, %	51,1	37,7	38,1	42,1
Recall, %	31,6	26,5	27,7	33,2
FPS	9	39	36	27

Узагальнюючи, двостадійний детектор FasterR-CNN показав високу точність детекції об'єктів, але через низьку швидкість (лише 9 кадрів в сек.) не є оптимальним варіантом для використання з відео. SSD та RetinaNet підходять для завдань, де ключовою є точність детекції об'єктів, хоч вони повільніші за YOLO. SSD враховує об'єкти у різних масштабах, тоді як високий показник повноти детектора RetinaNet (33,2%) підтверджує високу точність у визначенні класів об'єктів.

Отже, оскільки для вирішення поставленої проблеми важливі точність детектування та можливість функціонування в режимі реального часу, була обрана мережа YOLO для подальшого вдосконалення. Зараз існує кілька варіацій CNN-YOLO. В цій роботі планується використовувати YOLO третьої версії. Незважаючи на те, що YOLOv3 не є із останніх, проте досить оптимально збалансована за точністю та швидкістю.

3 СТВОРЕННЯ ТА ОЦІНКА ПРОДУКТИВНОСТІ КОМБІНОВАНОЇ НЕЙРОННОЇ МЕРЕЖІ

З метою покращання детектування людини на відео в реальному часі пропонується розробити нейронну мережу, яка включатиме в себе додаткові функції HOG-дескриптора. На основі результатів порівняння якісних характеристик сучасних CNN в задачі детектування людини на відеокадрах, вирішено інтегрувати алгоритм HOG в мережу YOLO. YOLOv3 була обрана серед усіх доступних, оскільки вона відзначається оптимальним поєднанням точності та швидкості.

Гіпотетично метод «YOLO з доповненням HOG» використовує переваги обох підходів: нейронної мережі YOLO для швидкої та точної детекції об'єктів на зображеннях, а також алгоритму HOG для використання текстурної та градієнтної інформації для підвищення точності детектування.

HOG може бути корисним у ситуаціях, коли візуальна інформація обмежена, наприклад, у слабкій освітленості або коли контраст між об'єктом та фоном низький. Очікується, що використання такого підходу може покращити продуктивність і надійність системи детекції та відстежування людини в режимі реального часу.

Для трекінгу людини використовується алгоритм SORT, що є простим і ефективним підходом до відслідковування об'єктів в реальному часі. Він часто використовується в системах відстеження об'єктів на відео та реалізований з дотриманням швидкодії та невеликої складності обчислень.

3.1 Опис інструментарію та середовища для розробки

Вибір засобів програмування визначається потребами і вимогами конкретного проєкту.

Більшість дослідників, які працюють у сфері машинного навчання, використовують мову програмування Python для реалізації своїх проєктів. Це через те, що завдання, пов'язані з нейронними мережами та машинним навчанням, відрізняються від звичайних програм і вимагають певних можливостей від мови

програмування та середовища реалізації. Python відповідає цим вимогам і є популярним вибором для таких проектів.

Python – це високорівнева інтерпретована об’єктно-орієнтована мова програмування з динамічною типізацією. Мова програмування Python підтримує модулі та пакети модулів, що сприяє повторному використанню коду. Крім того, Python та його стандартні бібліотеки відкриті у скомпільованій та вихідній формі на всіх основних платформах. Саме Python було використано для вирішення поставленої в роботі задачі.

Обравши мову програмування Python, як середовище розробки було вибрано Google Colaboratory, відомий також як Colab – це безкоштовне інтерактивне середовище для розробки програмного забезпечення, особливо зорієнтоване на машинне навчання та аналіз даних. Основні можливості Colab включають:

Інтерактивні блокноти: Використання блокнотів Colab (заснованих на Jupyter Notebook) для написання та виконання коду Python у веб-браузері без необхідності налаштування середовища розробки на локальному комп'ютері.

Широкий вибір бібліотек: Попередньо встановлені популярні бібліотеки Python для роботи з машинним навчанням та аналізом даних, такі як TensorFlow, PyTorch, Pandas, NumPy тощо.

Використання ресурсів Google: Можливість використовувати потужні обчислювальні ресурси Google, такі як графічні процесори (GPU) та тензорні процесори (TPU), для прискорення обчислень. В якості апаратного прискорювача використовувався T4 GPU.

Спільна робота та обмін кодом: Зручна можливість спільно працювати над проектами та обмінюватися кодом з іншими користувачами, а також використовувати різноманітні файли, зокрема тексти, зображення та дані.

Збереження та завантаження даних: Можливість зберігати дані на Google Drive або завантажувати їх з різних джерел, таких як GitHub, Google Cloud Storage тощо.

Keras – це високорівнева бібліотека для глибокого навчання, розроблена з метою спростити процес побудови та навчання нейронних мереж. Ось деякі характеристики Keras:

- простота використання: Keras надає простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати, навчати та валідувати моделі глибокого навчання;
- портативність: Keras можна використовувати як надбудову над різними обчислювальними бекендами, такими як TensorFlow, Theano або Microsoft Cognitive Toolkit (CNTK). Це дозволяє зручно переносити моделі між різними середовищами та платформами;
- модульність: Keras реалізований як набір модулів, що дозволяє використовувати лише ті частини, які потрібні для конкретного завдання. Він має просту структуру, що спрощує зміну та розширення моделей;
- розширеність: Keras має велику кількість вбудованих функцій та можливостей, таких як різні види шарів, функцій втрат та оптимізаторів.

OpenCV – це потужна бібліотека комп'ютерного зору, яка надає широкі можливості для роботи з зображеннями і відео. Вона має широкий функціонал, включаючи алгоритми детекції об'єктів та відстеження, а також аугментацію даних. OpenCV оптимізована для роботи в реальному часі і підтримується на різних платформах. Це безкоштовна бібліотека з відкритим вихідним кодом, що робить її доступною для широкого кола розробників.

3.2 Проєктування НМ

Проєктування програмного продукту допомагає забезпечити ефективне та структуроване створення програмного забезпечення, що відповідає потребам користувачів та вимогам бізнесу.

Алгоритм роботи моделі демонструється за допомогою діаграми діяльності, що складається з блоків, які представляють конкретні дії або етапи, та стрілок, які показують потік від одного етапу до іншого (Додаток А). Переваги використання діаграми діяльності полягають у її здатності чітко та доступно відображати

послідовність дій та взаємозв'язки між ними, що полегшує розуміння та використання моделі.

3.3 Підготовка набору даних

В цілому, підготовка датасету на етапі проєктування комбінованої нейронної мережі є важливим кроком, який впливає на якість та ефективність моделі.

Для систем відеоспостереження активно застосовуються панорамні камери, такі як камери типу «Риб’яче око». Вони забезпечують широкий кут огляду – до 360 градусів, і експлуатуються у різних громадських місцях, таких як аеропорти, готелі, лікарні, магазини, офісні приміщення. Тренування комбінованої НМ відбувалося за допомогою відеодатасету CEPDOF, який містить записи саме з камер «Риб'яче око» (див. п. 2.2.1).

Був написан код, який змінює розміри зображень з 2048×2048 на забраження розміром 640×640 та відповідно масштабує їх анотації, зберігаючи результати в нових папках (Додаток В).

Дані розподілено на навчальну та валідаційну вибірки. Навчання мережі проводилося на відео «Lunch meeting2», яке мало 1500 кадрів. Тоді як відео «Edge cases», що складається з 1500 кадрів було використано для тестування.

Реалізація попередньої обробки даних для тренування моделі машинного навчання з використанням бібліотеки KerasCV. Основні кроки обробки даних включають:

Визначення методів аугментації – RandomFlip та RandAugment. Аугментація даних – це процес створення нових прикладів даних шляхом застосування різноманітних трансформацій до наявних даних. Це може включати в себе обертання, зміщення, масштабування, зміну яскравості та контрастності зображень, а також інші операції, які допомагають розширити набір даних для тренування моделі. Аугментація даних є ефективним способом запобігання перенавчанню моделі та поліпшення її здатності до узагальнення, оскільки вона робить модель менш чутливою до варіацій у вхідних даних і допомагає їй вчитися

розпізнавати об'єкти в різних умовах та позиціях. Саме RandomFlip випадковим чином обертає зображення горизонтально і вертикально, вказавши mode="horizontal and vertical". RandAugment застосовує рандомізовані покращення до зображень, такі як зміна яскравості, контрасту тощо.

Визначення функції аугментації: Функція augment використовує визначені методи аугментації для обробки вхідних даних. Вона застосовує RandAugment до зображень, а потім застосовує RandomFlip.

Мапування даних: Функція augment використовується для мапування аугментованих даних до навчального та валідаційного наборів даних (train ds та val ds) з використанням методу map. Після цього вони конвертуються в кортежі (зображення, bounding boxes).

Попереднє завантаження даних: Завантажені дані попередньо завантажуються для ефективного тренування та оцінки моделі за допомогою методу prefetch. Це дозволяє асинхронно завантажувати та підготовляти наступний пакет даних під час навчання моделі, щоб уникнути затримок під час тренування (Додаток В).

3.4 Структура НМ

YOLO виступає базовою моделлю для розв'язання підзадачі object detection. Для уникнення необхідності у великої кількості обчислювальних ресурсів використано готову, попередньо навчену модель yolo3-tiny, яка вже має знання про об'єкти з набору даних COCO [33][35].

Вхідне зображення: початкове зображення подається на вхід комбінованої мережі.

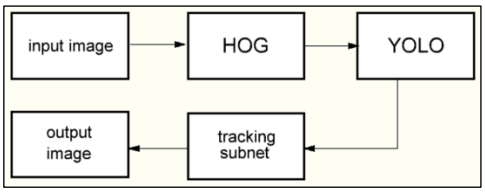
HOG-дескриптор: зображення перетворюється за допомогою методу HOG для вилучення ознак, що описують його текстуру та форму.

YOLO: ознаки, отримані від HOG-дескриптора, подаються на вхід YOLO для детектування об'єктів. YOLO аналізує зображення та визначає прямокутники, які містять об'єкти та класифікує їх.

Підмережа відслідковування: SORT працює шляхом поєднання детекції об'єктів з трекінгом. Він починає з детекції об'єктів у кожному кадрі відео, створює трекери для кожного об'єкта, прогнозує їхнє майбутнє положення, а потім здійснює асоціацію між виявленими об'єктами та трекерами, щоб оновити їхнє стан. Трекери, які не збігаються з виявленнями, можуть бути видалені, і нові трекери можуть бути створені для нових об'єктів. Цей процес повторюється для кожного кадру відео.

Вихідне зображення: на основі інформації, отриманої в результаті детекції та відслідковування об'єктів, створюється вихідне зображення, на якому об'єкти позначені та відстежені.

Загальна стуктура комбінованої моделі надана на рисунку 3.1



Рисунку 3.1 – Загальна стуктура комбінованої НМ

Якщо розглянути алгоритм роботи НМ у більш детальному ключі. Починаючи з вхідного відео, воно перетворюється у послідовність кадрів, які становлять набір зображень. Кожен кадр проходить попередню обробку та подається на вхід мережі. Кожен кадр обробляється за допомогою алгоритму HOG, що дозволяє виділити ознаки шляхом обчислення напрямків та величин градієнтів для невеликих областей. Ці ознаки збираються у вектор, який потім подається на вхід попередньо навченої моделі YOLO для об'єктного виявлення і класифікації.

Відслідковування об'єктів відбувається у функції update класу Sort:

```
    def update(self, dets= np.empty((0,6))):
        """
        Параметри:
        'dets' - масив numpy у форматі [[x1, y1, x2, y2, score],
[x1,y1,x2,y2,score],...]
```

```
        Обов'язково викликати цей метод навіть, якщо на кадрі немає
виявлених об'єктів. (передайте np.empty((0,5)))

        Повертає подібний масив, де останній стовпець - ідентифікатор
об'єкта (замість рівня впевненості)

        ЗАМІТКА: Кількість об'єктів, які повертаються, може відрізнятися
від кількості наданих об'єктів.
        """
        self.frame_count += 1

        # Отримати передбачені місця з існуючих трекерів
        trks = np.zeros((len(self.trackers), 6))
        to_del = []
        ret = []
        for t, trk in enumerate(trks):
            pos = self.trackers[t].predict()[0]
            trk[:] = [pos[0], pos[1], pos[2], pos[3], 0, 0]
            if np.any(np.isnan(pos)):
                to_del.append(t)
        trks = np.ma.compress_rows(np.ma.masked_invalid(trks))
        for t in reversed(to_del):
            self.trackers.pop(t)
        matched,         unmatched_dets,         unmatched_trks         =
associate_detections_to_trackers(dets, trks, self.iou_threshold)

        # Оновити відповідні трекери, призначені виявленим об'єктам
        for m in matched:
            self.trackers[m[1]].update(dets[m[0], :])

        #  Створити  і  ініціалізувати  нові  трекери  для  непризначених
об'єктів
        for i in unmatched_dets:
            trk = KalmanBoxTracker(np.hstack((dets[i,:], np.array([0]))))
            self.trackers.append(trk)

        i = len(self.trackers)
        for trk in reversed(self.trackers):
            d = trk.get_state()[0]
            if  (trk.time_since_update  <  1)  and  (trk.hit_streak  >=
self.min_hits or self.frame_count <= self.min_hits):
                ret.append(np.concatenate((d,  [trk.id+1])).reshape(1,-1))
#+1'd because MOT benchmark requires positive value
            i -= 1
            # Видалити "мертві" треки
            if(trk.time_since_update >self.max_age):
                self.trackers.pop(i)
        if(len(ret) > 0):
            return np.concatenate(ret)
        return np.empty((0,6))
```

Цей метод приймає масив виявлених об'єктів dets, де кожен об'єкт представлений у форматі [x_1, y_1, x_2, y_2, score]. Він викликає метод predict для кожного існуючого трекера, оновлює відповідні трекери для об'єктів, які були

відсортовані, і створює нові трекери для об'єктів, які не мають відповідностей. Нарешті, він повертає масив трекерів.

3.5 Навчання мережі

В навчанні підмережі для детекції об'єктів використовується метод оптимізації, який називається стохастичним градієнтним спуском, момент якого 0,9. Цей метод полягає в ітеративному оновленні параметрів мережі за допомогою градієнту функції помилок, причому кроки оновлення пропорційні протилежному градієнту. Функція помилок визначається як квадратична різниця між передачуванними і фактичними значеннями (див. п. 2.1.2). Експериментально було визначено, що найбільш оптимальне число епох навчання – 30. У цьому випадку модель мала точності на рівні 38,9%, а швидкість обробки складала 41 кадр в секунду. Порівняно з результатами детектора YOLO, поєднання YOLO з дескриптором HOG підвищило якість і швидкість мережі. Проте, тривалість навчання збільшилася на понад 11 хвилин, що, можливо, пов'язано зі складністю розширеної архітектури.

Згідно з графіком дінамики точності (рис. 3.3), можна припустити, що мережа може страждати від проблеми недостатнього навчання. В результаті було прийнято рішення доопрацювати навчання мережі, застосувавши більшу кількість епох. Після 100епох навчання мережа досягла точності 39,5% при обробці 43кадрів за секунду. Оглядаючи графік (рис. 3.3), можна зробити висновок, що проблему недостатнього навчання моделі було вирішено.

протягом 30 епох тренування протягом 100 епох тренування

Рисунок 3.3 – Графіки динаміки точності

Отже, проведені дослідження підтвердили, що поєднання hog-дескриптора з YOLO на датасете CEPDOF призвело до підвищення точності на 1,8% і збільшення швидкості обробки на чотири кадри в сек., попри те, що тривалість навчання зросла до 50 хвил. (табл. 3.1)

Таблиця 3.1 – Оцінка ефективності детекторів YOLO та YOLO з додаванням функцій HOG на відеодатасеті CEPDOF

Показники ефективності мережі	YOLO	YOLO з додатковими функціями HOG
mAP, %	37,7	39,5
Recall, %	26,5	27,5
FPS	39	43

3.6 Проведення тестування

В результатах дослідження важливу роль відіграє емпіричний аналіз. Проведено аналіз роботи нейронної мережі в різних умовах, таких як перекриття об'єктів на відео та низька освітленість кадрів.

Спочатку було використано відеофайл в форматі .mp4 із залу засідання (рис. 3.4), де програма вдало розпізнала та відстежила присутніх людей, де деякі були неповні та перекриті.

Рисунок 3.4 – Демонстрація функціонування моделі

У другому експерименті нейромережа була протестована на стійкість до спотворень вхідних даних за умови використання відео з пішоходами на вулицях у вечірній час в м. Київ (рис. 3.5) та м. Харків (рис. 3.6), де об'єкти відстеження мали нечіткі контури, були перекриті.

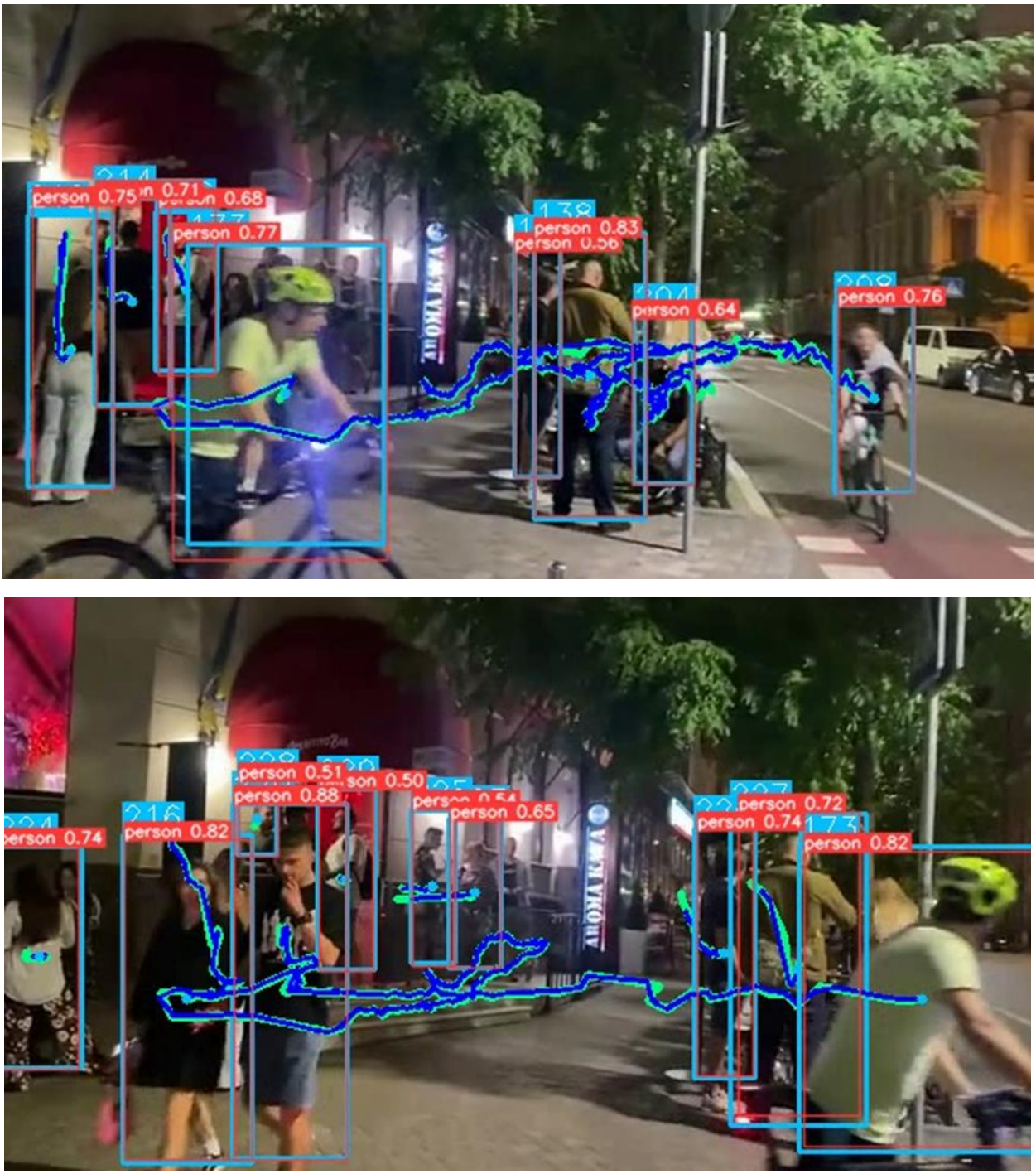


Рисунок 3.5 – Результати функціонування моделі при перекритті об’єктів

Програма коректно розпізнала та відстежила об'єкти, навіть ті, які знаходилися далеко від камери.

Рисунок 3.6 – Результати функціонування мережі в умовах недостатнього освітлення

Також було зауважено, що збільшення близькості об'єктів призвело до підвищення точності їх розпізнавання.

Далі було проведено аналіз роботи мережі на відео з камер спостереження торгівельного центру (рис. 3.7), де було важливо виявити об'єкти, які були перекриті. Хоча мережа успішно впоралася з цим завданням, проблема детектування невеликих об'єктів залишається відкритою.

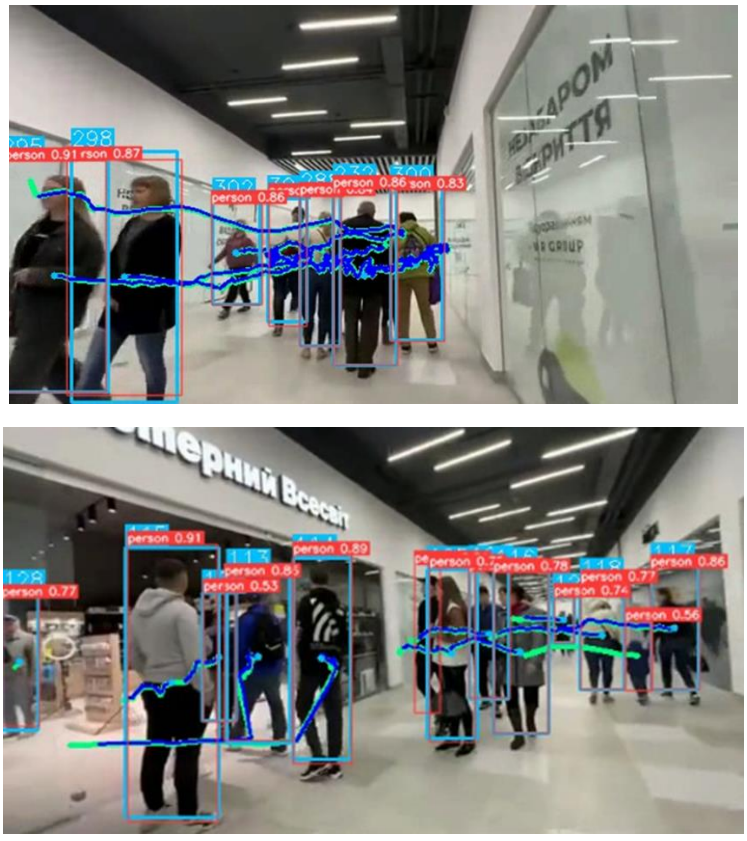


Рисунок 3.7 – Приклад роботи мережі з відео файлом

ВИСНОВКИ

У межах кваліфікаційної роботи досліджено існуючі методи і підходи до детектування та моніторингу людини в реальному часі. Метою роботи було підвищення результативності детектування та моніторингу людей в відеопотоці.

Для вирішення намічених завдань використовувалися теоретичні та емпіричні методи дослідження. Вирішені наступні завдання. Під час дослідження існуючих методів розпізнавання об'єктів на зображеннях, включаючи традиційні техніки (алгоритм Віоли-Джонса та гістограми напрямлених градієнтів), став виявлений ряд переваг згорткових нейронних мереж у вирішенні задач детектування об'єктів. Вмотивовано використання нейромережевого підходу до вирішення задачі детекції людини в режимі реального часу.

Були розглянуті принципи функціонування, слабкі та сильні сторони сучасних детекторів – YOLO, SSD, RetinaNet, FasterR-CNN при вирішенні завдань детектування. Через порівняльний аналіз їх продуктивності за швидкістю та точністю розпізнавання на датасеті CEPDOF з'ясовано наступне. Двоетапний детектор FasterR-CNN показав високу точність, але його обробка відео відбувається повільно. RetinaNet і SSD також забезпечують точне виявлення об'єктів, проте менш ефективні за швидкістю у порівнянні з YOLO. SSD здатний працювати з об'єктами різних масштабів, тоді як RetinaNet відзначається високим показником повноти, що підтверджує його точність у класифікації. Серед всіх розглянутих мереж, YOLO видається найшвидшою і може бути застосована в режимі реального часу.

Для отримання кращих результатів детекції людини у реальному часі була створена модель нейронної мережі шляхом комбінування двох потужних методів комп'ютерного зору: класичного дескриптора HOG і нейронної мережі YOLO. Застосування HOG, що збагачує інформацію про текстури та форми об'єктів на зображенні, дозволяє удосконалити підхід до попередньої обробки даних перед використанням мережею YOLO.YOLO – це швидкий та точний алгоритм для об'єктного виявлення, тому він ідеально підходить для використання після

передобробки кадрів за допомогою HOG. Використання попередньо натренованої моделі YOLO дозволяє використовувати загальні знання про об'єкти, що допомагає підвищити точність детекції. Для навчання мережі було підготовлено обраний відеодатасет.

Було визначено, що комбінація нейронної мережі YOLO із додатковими функціями HOG-дескриптора для екстрагування ознак перед передачею на вхід детектора дозволила підвищити точність та збільшити швидкодію обробки відеокадрів.

За результатами проведеного тестування доповненої мережі в різних умовах можна зробити висновки, що мережа ефективно функціонує за умов обмеженого освітлення, але існують питання щодо детекції дрібних об'єктів, які залишаються відкритими для подальших досліджень.

Для проведення дослідженя використовувалися мова програмування Python, бібліотеки Keras, OpenCV, а також платформа Google Colaboratory для розробки.

У майбутньому можна випробувати покращення ефективності моделі шляхом зміни розмірності кадру, який подається на вхід моделі. Також можна додати додаткові функції – систему аналізу кількості людей в відео потоці.

Результати роботи апробовано у вигляді тез доповідей під час IX Міжнародної науково-технічної конференції «Поліграфічні, мультимедійні та web-технології» (Україна, 14-18 травня 2024 року)[34].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Video Analytics Market worth $22.6 billion by 2028 // Markets and Markets. URL: http://www.marketsandmarkets.com/PressReleases/iva.asp.

2. A Gentle Introduction to Computer Vision / Machine Learning Mastery. URL: https://machinelearningmastery.com/what-is-computer-vision/.

3. Everything You Ever Wanted To Know About Computer Vision / Towards Data Science. URL: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.

4. Довбиш А.С., Шелехов І.В. Основи теорії розпізнавання образів: навч. посіб. у 2 ч. Суми: Сумський державний університет. 2015. ч. 1. 109 с.

5. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.

6. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R- CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

7. P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features / 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. 1. 8–14 December 2001 / The Institute of Electrical and Electronics Engineers, Inc. C. 511–518.

8. H.A. Rowley, S. Baluja, T.Kanade. Neural Network-Based Face Detection // PAMI, January 1998.

9. Р. Гонсалес, Р. Вудс. Цифровая обработка изображений, М.: Техносфера, 2005. – 1072 с.

10. R.V. Hattem, Mastering Python: master the art of writing beautiful and powerful Python by using all of the features that Python 3.5 offers, Packt Publishing, 2016.

11. Dalal N., Triggs B. Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition Conference (CVPR). 2005.

12. Histogram of oriented gradients [Електронний ресурс] – Режим доступу до ресурсу: https://www.learnopencv.com/histogram-oforiented-gradients/.

13. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі. Харків: Компанія СМІТ. 2006.

14. Функції активації: ступінчаста, лінійна, сигмоїда, ReLU та Tanh// robot_dreams [Електронний ресурс]. URL: https://robotdreams.cc/uk/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh.

15. Hubel D., Wiesel T. Receptive fields of single neurones in the cat's striate cortex. J. Physiol. 148 (3): 574-91. 1959.

16. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog. [Електронний ресурс] – режим доступу: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

17. Liu Y., Zhang Y.-M., Zhang X.-Y., Liu C.-L. Adaptive spatial pooling for image classification. Pattern Recognition. 2016. Vol. 6. P. 58–67.

18. Szegedy C., Liu W., Jia Y. et al. Going Deeper with Convolutions. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

19. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015: Proceedings of the 3rd International Conference on Learning Representations. San Diego, California, USA, 7 May – 9 May, 2015.

20. Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, Stephen Lin. RepPoints: Point Set Representation for Object Detection. In: International Conference on Computer Vision (ICCV). 2019.

21. Ren S., et al. FasterR-CNN: Towards real-time object detection with region proposal networks. NIPS 2015: Proceedings of Advances in neural information processing systems. Montreal, Quebec, Canada. 7 December – 12 December, 2015. P.91–99.

22. Girshick R. FastR-CNN. In: International Conference on Computer Vision (ICCV). 2015.

23. Redmon J., Farhadi A. YOLO9000: better, faster, stronger. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

24. Redmon J., Farhadi A. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767. 2018.

25. Liu W. et. al. SSD: Single shot multibox detector. In European conference on computer vision. 2016. pp. 21-37.

26. Lin T.-Y. et. al. Focal Loss for Dense Object Detection. In: International Conference on Computer Vision (ICCV). 2017. pp. 2980-2988.

27. Li S., Tezcan M. O., Ishwar P., Konrad J. Supervised people counting using an overhead fisheye camera. In: International Conference on Advanced Visual and Signal-Based Surveillance (AVSS). 2019.

28. Vavassori L. SSC: Single-Shot Multiscale Counter: Counting Generic Objects in Images. 2019.

29. FasterR-CNN model with a ResNet-50-FPN backbone. URL: https://pytorch.org/vision/stable/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html.

30. Ultralytics. YOLOv3. URL: https://github.com/ultralytics/yolov3.

31. SSD: Single-Shot MultiBox Detector implementation in Keras. URL: https://github.com/pierluigiferrari/ssd_keras.

32. RetinaNet model with a ResNet-50-FPN backbone. URL: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.retinanet_resnet50_fpn.html.

33. YOLO: Real-Time Object Detection. URL: https://pjreddie.com/darknet/yolo/.

34. Миронюк С.А., Четвериков Г.Г. Аналіз методів детекції та моніторингу людини у відео потоках // IX Міжнародної науково-технічної конференції «Поліграфічні, мультимедійні та web-технології». 2024.

35. Smelyakov, K., Bohomolov, O., Kizitskyi, M., Chupryna, A. Identification of Modern Facial Emotion Recognition Models CEUR Workshop Proceedings, 2022, 3171, pp. 1267–1281.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

35. Smelyakov, K., Bohomolov, O., Kizitskyi, M., Chupryna, A. Identification of Modern Facial Emotion Recognition Models CEUR Workshop Proceedings, 2022, 3171, pp. 1267–1281.