НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.93’1

До захисту допущено
В. о. зав. кафедри ШІ
_____О. І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 122 «Комп’ютерні науки»

на тему: «Розпізнавання та відстеження людини в режимі реального часу»

Виконала:
студентка ІІ курсу, групи КІ-11мп
Русакова Лариса Олексіївна _____

Керівник:
к.т.н., старший викладач
Шаповал Наталія Віталіївна _____

Рецензент:
к.т.н., доцент кафедри ІКТС ІТС НТУУ «КПІ»
Астраханцев Андрій Анатолійович _____

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

Студент _____

Київ 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки »

ЗАТВЕРДЖУЮ

В. о. зав. кафедри

_____ О. І. Чумаченко

«___» _____ 2022 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Русаковій Ларисі Олексіївні

1. Тема дисертації: «Розпізнавання та відстеження людини в режимі реального часу», науковий керівник роботи Шаповал Наталія Віталіївна, к.т.н., старший викладач, затверджено наказом по університету від «03» листопада 2022 р. № 4046-с.

2. Термін подання студентом дисертації: 15.12.2022.

3. Об'єкт досліджень: відео з людьми, зібрані з камер відеоспостереження.

4. Предмет досліджень: методи виявлення та відстеження людини у відеопотоці.

5. Перелік завдань, які потрібно зробити: здійснити огляд технічної літератури за темою роботи; дослідити актуальність обраної теми; ознайомитись з існуючими методами виявлення і відстеження об'єктів у відео; здійснити порівняльний аналіз

наявних методів; розробити та реалізувати гібридну нейронну мережу (дескриптор HOG і YOLO) для виявлення і відстеження людини в режимі реального часу; провести експеримент з метою оцінки ефективності розробленої мережі; здійснити маркетинговий аналіз стартап-проекту; оформити пояснювальну записку.

6. Орієнтовний перелік графічного матеріалу: архітектура гібридної мережі, діаграма класів, діаграма діяльності.

7. Орієнтовний перелік публікацій: публікації тез доповідей в матеріалах I Міжнародної науково-практичної конференції «Scientific Research In The Modern World» (Канада, 9-11 листопада 2022 р.), I Всеукраїнської науково-практичної конференції «Системні науки та інформатика» (Україна, 22-29 листопада 2022 р.) та XXII Міжнародної науково-технічної конференції «Artificial Intelligence And Intellegent Systems (AIIS'2022)» (Україна, 8-9 грудня 2022 р.).

Календарний план

№	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Аналіз предметної області, постановка задачі, опрацювання літератури за обраною тематикою	05.09.2022 – 23.09.2022	Виконано
2	Огляд існуючих рішень, їх порівняльний аналіз	23.09.2022 – 17.10.2022	Виконано
3	Розроблення гібридної мережі	17.10.2022 – 02.11.2022	Виконано
4	Проведення експериментів зі створеною мережею	02.11.2022 – 14.11.2022	Виконано
5	Підготовка стартап-проекту	14.11.2022 – 23.11.2022	Виконано
6	Оформлення пояснювальної записки	23.11.2022 – 28.11.2022	Виконано

Студент Русакова Л. О.

Керівник Шаповал Н. В.

РЕФЕРАТ

Магістерська дисертація: 100 сторінок, 36 рисунків, 25 таблиць, 41 посилання, 2 додатки.

Актуальність теми полягає у необхідності створення ефективної мережі для виявлення і відстеження людини, яка дозволить значно точніше і швидше обробляти відеокадри.

Об’єкт досліджень – відео з людьми, зібрані з камер відеоспостереження.

Предмет – методи виявлення та відстеження людини у відеопотоці.

Метою роботи є підвищити ефективність виявлення і відстеження людини у відеопослідовностях шляхом удосконалення згорткової нейромережі типу YOLO.

Завдання: огляд переваг і недоліків сучасних підходів до розпізнавання об’єктів у відеопотоці; порівняння швидкості й точності нейронних мереж Faster R-CNN, YOLO, SSD, RetinaNet на датасеті HABBOF; розробка гібридної нейронної мережі (дескриптор HOG і YOLO) для виявлення і відстеження людини в режимі реального часу; проведення дослідження ефективності розробленої мережі та порівняти з існуючими аналогами на обраному наборі даних.

Основні результати: розроблено мережу для виявлення і одночасного відстеження людини у відеопотоці на основі YOLO; підвищено швидкість і точність виявлення об’єктів мережі YOLO завдяки використанню HOG для витягу ознак.

Практичне значення дослідження полягає у створенні нейронної мережі для швидкого і точного виявлення і відстеження людини у системах відеоаналітики.

Ключові слова: виявлення об’єктів, відстеження об’єктів, згорткова нейронна мережа, гістограма орієнтованих градієнтів.

ABSTRACT

Master’s thesis: 100 pages, 36 figures, 25 tables, 41 references, 2 appendices.

Relevance of the research topic is the need to create an effective human detection and tracking network that will allow much more accurate and faster processing of video frames.

The object of the study is videos with people collected from video surveillance cameras.

The subject of the study is real-time human detection and tracking method.

Research purpose is to improve existing methods for solving the detection and tracking problems by improving the convolutional neural network of the YOLO type.

Tasks: an overview of the advantages and disadvantages of modern approaches to object recognition in a video stream; comparison of speed and accuracy of Faster R-CNN, YOLO, SSD, RetinaNet neural networks on the HABBOF dataset; development of a hybrid neural network (HOG and YOLO descriptor) for the real-time human detection and tracking; conduct a study of the effectiveness of the developed network and compare it with existing analogues on the selected dataset.

The main results: a network for the real-time human detection and tracking based on YOLO was developed; the speed and accuracy of YOLO network object detection was improved by using HOG for feature extraction.

The practical value of the research lies in the creation of a neural network for fast and accurate human detection and tracking in the video analytics systems.

Key words: object detection, object tracking, convolutional neural network, histogram of oriented gradient.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

FPN – Feature Pyramid Network

FPS – frames per second

HABBOF – Human-Aligned Bounding Boxes from Overhead Fisheye cameras

HOG – Histogram of Oriented Gradients

IoF – intersection over union

mAP – mean average precision

NMS – non-maximum suppression

R-CNN – Region-based Convolutional Neural Network

ReLU – rectified linear unit

RoI – Region of Interest

RPN – Region Proposal Network

SSD – Single Shot MultiBox Detector

SPP – spatial pyramid pooling

SVM – support vector machine

UML – Unified Modeling Language

YOLO – You Only Look Once

ВСТУП

На сьогоднішній день системи відеоспостереження використовуються майже у всіх сферах людської діяльності. Дані, зібрані з відеокамер, потребують обробки. Аналіз відеоконтенту (Video content analysis, VCA) або відеоаналітика (VA) – це можливість автоматичного аналізу відео для виявлення та визначення часових і просторових подій [1]. Відеоаналітика знайшла застосування у охоронних системах, роздрібній торгівлі, автомобільній промисловості, технологіях розумного будинку тощо.

Однією із задач, які вирішує відеоаналітика, є саме виявлення і відстеження людей у відеопотоці. Системи розпізнавання, відстеження осіб у відео застосовуються у системах автоматичного керування для виявлення пішоходів на дорогах, у магазинах для підрахунку кількості та аналізу поведінки клієнтів чи розрахунку навантаження на робочий персонал, на зупинках громадського транспорту і в аеропортах для аналізу пасажиропотоку, у офісних приміщеннях і системах розумного будинку тощо.

Зазвичай системи відеоаналітики являють собою спеціалізоване програмне забезпечення для аналізу відеоданих, зібраних з вебкамер, й інтелектуальної оцінки ситуації. Існують і окремі камери відеоспостереження із вбудованими функціями відеоаналітики. Програмні підходи до виявлення і відстеження людини досить різноманітні, створюються спеціальні застосунки і веб-додатки чи окремі програмні модулі. Останнім часом широкого поширення набули нейронні мережі. Їх можна навіть безпосередньо приєднувати до камер. Але при такому підході можуть виникнути проблеми з виявленням дрібних чи неповних об’єктів, стійкістю до зашумлених чи погано освітлених зображень. Багато часу займає навчання мереж як і сам процес виявлення. Деякі мережі обробляють відео зі швидкістю менше 10 кадрів за секунду, що не прийнятно для застосування у режимі реального часу. Тому актуальність роботи полягає у необхідності створення ефективної мережі для

виявлення і відстеження людини, яка дозволить значно точніше і швидше обробляти відеокадри.

Об’єкт досліджень – відео з людьми, зібрані з камер відеоспостереження. Предмет – методи виявлення та відстеження людини у відеопотоці.

Метою роботи є підвищити ефективність виявлення і відстеження людини у відеопослідовностях шляхом удосконалення згорткової нейромережі типу YOLO.

Для досягнення мети було поставлено наступні завдання:

а) огляд переваг і недоліків сучасних підходів до розпізнавання об’єктів у відеопотоці;

б) порівняння швидкості й точності нейронних мереж Faster R-CNN, YOLO, SSD, RetinaNet на датасеті HABBOF;

в) створення нейронної мережі (дескриптор HOG і YOLO для виявлення та алгоритм відстеження на основі виявлення) для виявлення і відстеження людини в режимі реального часу;

г) проведення дослідження ефективності розробленої мережі та порівняння з існуючими аналогами на обраному наборі даних.

Перший розділ роботи присвячений огляду задачі виявлення і відстеження об’єктів у режимі реального часу та існуючих методів їх розв’язання. Обґрунтований вибір нейромережевого підходу до вирішення поставленої у роботі задачі.

У другому розділі проведено порівняльний аналіз нейронних мереж Faster R-CNN, YOLO, SSD, RetinaNet, натренованих на датасеті HABBOF.

У третьому розділі описано процес розробки гібридної нейронної мережі на основі дескриптора HOG і детектора YOLO.

Четвертий розділ роботи відведений для маркетингового аналізу стартап-проекту.

Основні результати досліджень було представлено на I Міжнародній науково-практичній конференції «Scientific Research In The Modern World» [2], I Всеукраїнській науково-практичній конференції «Системні науки та інформатика» [3] та XXII Міжнародній науково-технічній конференції «Artificial Intelligence And Intellegent Systems (AIIS’2022)» [4].

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Об’єкт – певна область у просторі ознак. Ознака – характеристика властивості об’єкта або явища навколишнього середовища. Простір ознак – N-вимірний простір, визначений для даної задачі розпізнавання (N – число ознак). Вектор з простору ознак x, відповідний об’єкту задачі розпізнавання, – N-вимірний вектор з компонентами (x_1, x_2, …, x_N), що утворюють значення ознак для даного об’єкта [5]. Тобто розпізнавання образів характеризується віднесенням вихідних даних до певного класу за допомогою виділення істотних ознак або властивостей цих даних із загальної сукупності ознак. Сюди можна віднести наступні задачі комп’ютерного зору: класифікацію, локалізацію та виявлення об’єктів. Виявлення об’єктів полягає у визначенні наявності об’єкта у просторі (його позиції та приналежності до певного класу). Ця задача поєднує в собі локалізацію та класифікацію об’єктів.

Існує багато різних методів виявлення об’єктів на зображеннях. Відео є складнішим за зображення, оскільки воно має ще один вимір – час. Тим не менш, відео можна представити у вигляді серії (набору) зображень. І для його обробки достатньо прокрутити всі кадри у відеофайлі, застосувати відповідні методи розпізнавання об’єктів для кожного із кадрів. Але оскільки застосування одного і того самого алгоритму кожного разу не є ефективним з точки зору обчислень і не гарантує, що певний виділений об’єкт буде знайдено на кожному з кадрів (не всі детектори є стійкими до різних поз та ракурсів), додатково використовуються алгоритми відстеження об’єктів.

Відстеженням (трекінгом) об’єктів називають з’ясування місцеположення рухомого об’єкта (чи декількох об’єктів) у просторі з часом. Відстежування також передбачає прогнозування траєкторії подальшого руху об’єкта навіть у випадку, якщо він тимчасово вийде із зони спостереження (наприклад, зайде за транспортний засіб чи іншу конструкцію). Відстежувати можна один об’єкт (Visual Object Tracking) чи множину об’єктів (Multiple Object Tracking) (коли у відеокадрі

містяться два і більше об’єктів). Існують різні способи відстеження: кореляційний (полягає у навчанні кореляційного фільтра, за допомогою якого об’єкт відокремлюється від заднього фону на кадрі; будує траєкторії тільки заданих для пошуку об’єктів); на основі графів; міжкамерний (який передбачає встановлення синхронізованих відеокамер, що спостерігають за пов’язаними областями; об’єкт переходить з поля зору однієї камери в поле зору іншої). Та найпоширенішим методом є відстеження на основі виявлення (tracking-by-detection), в якому спочатку використовується алгоритм знаходження об’єкта, а далі за його координатами з попереднього кадру проводиться їх обчислення в наступному кадрі.

При розв’язанні задачі виявлення і відстеження об’єктів можуть виникнути проблеми з освітленням, перекриттям об’єктів (об’єкти можуть частково перекриватись або зникати з поля зору на невизначений час), зміною їх положення; наявністю дрібних об’єктів чи зашумленими зображеннями (відеокадрами). Ці фактори варто враховувати при розробці алгоритму виявлення об’єктів.

До методів саме розпізнавання об’єктів відносяться колірні фільтри, виділення контурів на зображенні, метод співставлення із шаблоном (що полягає у знаходженні на зображенні ділянок, які співпадають з зображенням шуканого об’єкта), метод виділення об’єкта на зображенні на основі ознак Хаара, метод опорних точок, нейронні мережі тощо. Так само і для трекінгу можна застосовувати алгоритми на основі міри Жаккара, фільтр Калмана чи витяг ознак. Вибір конкретного методу залежить від поставленої задачі тому надалі розглянуто класичні методи Віоли-Джонса, гістограми орієнтованих градієнтів та нейронні мережі для виявлення об’єктів. Відстеження проводитиметься на основі виявлення.

1.1 Метод Віоли-Джонса

Метод Віоли-Джонса – алгоритм виявлення об’єктів, запропонований у 2001 році Полом Віолою та Майклом Джонсом. Розроблявся насамперед для виявлення обличь, але так само може бути адаптований і до виявлення інших класів об’єктів. Він включає чотири етапи [6]:

1. створення інтегрального зображення;
2. вибір ознак Хаара;
3. навчання з використанням алгоритму AdaBoost (adaptive boosting);
4. каскадні класифікатори.

Для розрахунку яскравості прямокутної області зображення використовується інтегральне представлення зображення. Інтегральне представлення зображення – це матриця, яка збігається за розмірами з вхідним зображенням. Кожен елемент матриці є сумою інтенсивностей всіх пікселів, що знаходяться ліворуч і вище за цей елемент (рис.1.1).

Рисунок 1.1 – Приклад розрахунку інтегрального зображення

Ознаки, використовувані алгоритмом, використовують підсумовування пікселів з прямокутних регіонів. Ознака Хаара складається з суміжних прямокутних областей (рис.1.2). Вони позиціонуються на зображенні, далі підсумовується інтенсивність пікселів в областях, після чого обчислюється різниця між сумами. Ця різниця і буде значенням певної ознаки.

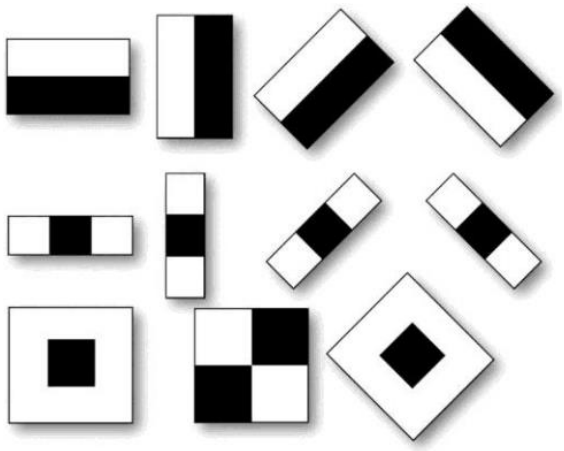


Рисунок 1.2 – Ознака Хаара

AdaBoost (Adaptive Boosting) – це жадібний алгоритм посилення класифікаторів шляхом об’єднання їх в композиції [6]. Він є адаптивним в тому сенсі, що кожна наступна композиція класифікаторів будується за об’єктами, невірно класифікованим попередніми композиціями.

Алгоритм полягає в наступному: кожна ознака застосовується до кожного зображення, відбираються ознаки з найменшою кількістю помилок. Спочатку тестовим зображенням присвоюються однакові ваги, а після кожної невірної класифікації вага збільшується. Ці операції повторюються доки не буде досягнута необхідна точність.

Переваги AdaBoost:

а) простота реалізації, швидкість роботи;

б) узагальнююча здатність, алгоритм підлаштовується під проблемні елементи навчальної вибірки;

в) можливість ідентифікації зашумлених об’єктів.

Недоліки AdaBoost:

а) чутливий до значної зашумленості й пропусків даних;

б) може призводити до побудови громіздких композицій, що складаються із сотень алгоритмів (такі композиції вимагають великих обсягів пам’яті та значних витрат на обчислення класифікацій).

в) повільне навчання, яке залежить від кількості класифікаторів і розміру навчальної вибірки.

Каскадна модель класифікатора – це дерево прийняття рішень (дерево, у ребрах якого записані атрибути, від яких залежить цільова функція, в «листях» – самі значення цільової функції, а в інших вузлах – атрибути, умови переходу), де кожен вузол якого побудований таким чином, щоб розпізнати усі необхідні образи, і відкинути решту. Крім цього, вузли дерева розміщені таким чином, що чим ближче вузол знаходиться до кореня, тим із меншої кількості примітивів він складається (вимагає менше часу на прийняття рішення). Такий вид каскадної моделі добре підходить для обробки зображень з невеликою кількістю образів.

1.2 Гістограма орієнтованих градієнтів

Гістограма орієнтованих градієнтів (Histogram of Oriented Gradients, HOG) – це метод витягу характеристик кольорів пікселів для класифікатора розпізнавання об’єктів, розроблений у 2005 році [7]. Підраховує випадки градієнтної орієнтації в певній частині зображення. Він дуже схожий на метод масштаб-незалежного перетворення ознак (Scale-Invariant Feature Transform, SIFT), але відрізняється тим, що обраховується в щільній сітці рівномірно розташованих комірок і для підвищення точності використовує локальну нормалізацію контрасту.

HOG використовує величину і кут градієнта для обчислення характеристик. Для областей зображення він генерує гістограми, використовуючи величину та орієнтацію градієнта. Основна ідея полягає в тому, що локальний вигляд і форму об’єкта на зображенні можна описати розподілом інтенсивності градієнтів або напрямком контурів.

Алгоритм HOG містить наступні кроки:

а) попередня обробка зображення (зміна розміру та нормалізація кольору);

б) обчислюється вектор градієнта кожного пікселя, а також його величина та напрямок; для цього спочатку обчислюються відповідні горизонтальний і

вертикальний градієнти фільтруванням зображення через ядра: $(-1 \quad 0 \quad 1)$, $(-1 \quad 0 \quad 1)^T$; далі знаходять величину та напрямок градієнтів за формулами: $g = \sqrt{g_x^2 + g_y^2}$, $\theta = \arctan\frac{g_y}{g_x}$; для кольорових зображень оцінюються градієнти трьох каналів (R, G, B), величина градієнта в пікселі є максимальною з цих трьох величин, а за напрямок обирають той, що відповідає максимальному градієнту;

в) зображення ділиться на невеликі області, які називаються комірками; для кожної комірки обчислюється гістограма градієнтів, яка є вектором із 9 бінів (чисел), що відповідають кутам від 0 до 160 градусів (для «беззнакових» градієнтів) чи від 0 до 360 градусів (для «зізнакових») (рис.1.3).

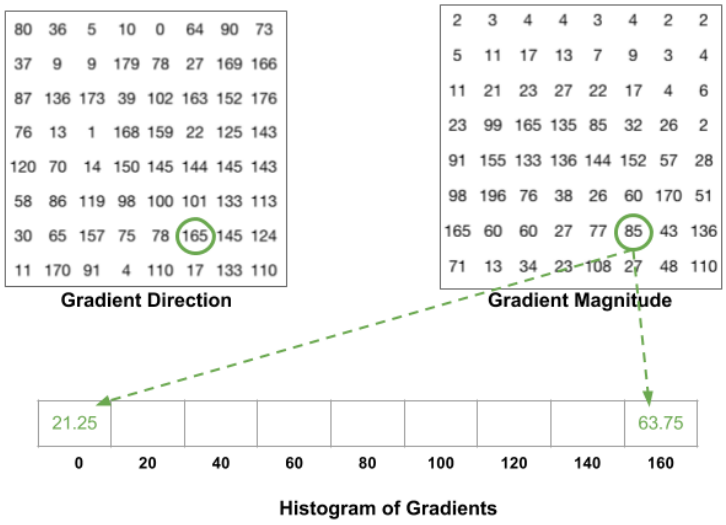


Рисунок 1.3 – Приклад розрахунку гістограми градієнтів

г) обчислені локальні гістограми нормалізуються за такими нормами:

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

де h – обчислені гістограми градієнтів, e – деяка мала константа;

це покращує інваріантність до освітлення, затінення та контрасту країв;

д) на останньому кроці локальні гістограми з усіх блоків, що перекривають вікно виявлення, зводяться у комбінований вектор ознак, які зрештою класифікуються.

1.3 Штучні нейронні мережі

Штучною нейронною мережею є математична модель, принцип роботи якої нагадує роботу мережі біологічних нейронів у мозку людини. Тож нейронна мережа – це сукупність великого числа порівняно простих елементів – нейронів, топологія з'єднань між якими залежить від типу мережі [8]. Штучний нейрон має послідовність вхідних сигналів. Сукупність таки нейронів організовано в шари. Вплив одного нейрону на інший визначається встановленими зв'язками. Вага зв'язку визначає силу впливу. Ваги відображають значимість кожного входу штучного нейрона. Щоб створити нейронну мережу для вирішення певної задачі, потрібно вибрати, яким чином з'єднувати нейрони один з одним, і підібрати значення вагових параметрів на цих зв'язках.

Залежно від кількості шарів, мережі можуть бути одношаровими і багатошаровими (що характеризуються наявністю одного або декількох скритих шарів). У багатошарових мережах нейрони кожного з шарів використовують в якості вхідних сигналів вихідні сигнали нейронів лише попереднього шару. Такі мережі можуть бути повнозв'язними (усі вузли кожного шару поєднані з усіма вузлами сусідніх шарів), і неповнозв'язними (окремі зв'язки в мережі відсутні).

1.3.1 Багатошаровий персептрон

Однією зі спрощених моделей штучного нейрона є персептрон. На вхідному шарі приймається вектор вхідних сигналів X, кожен з яких має відповідну вагу. Функція активації (яка буде детальніше описана в наступному підрозділі) обчислює результуюче значення нейрона і передає далі на вихідний шар. Схематично це зображено на рисунку 1.4.

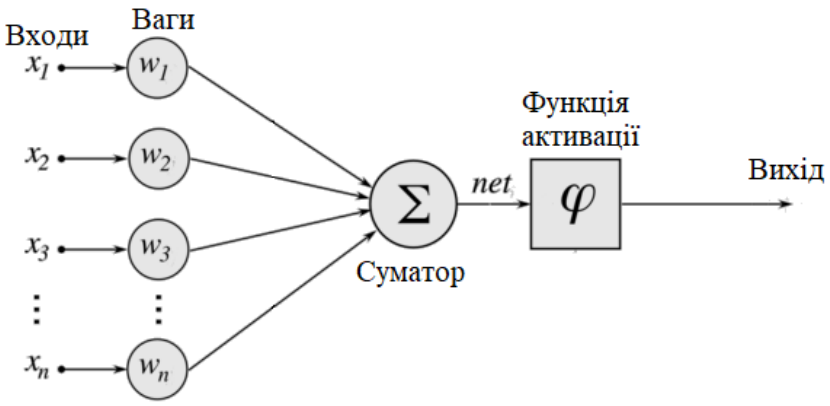


Рисунок 1.4 – Штучний нейрон

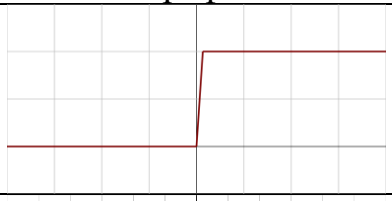
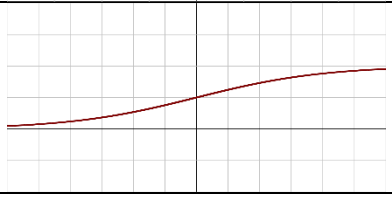
Багатошаровий персептрон – нейронна мережа прямого поширення сигналу (без зворотних зв'язків як у рекурентних мережах), у якій вхідний сигнал перетворюється у вихідний, проходячи послідовно через кілька скритих шарів [8].

Кожен вузол в шарі з'єднаний з кожен вузлом наступного шару, що робить мережу повнозв'язною. Така архітектура знаходить широке застосування в задачах розпізнавання мови і машинному перекладі.

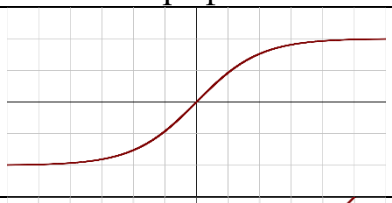
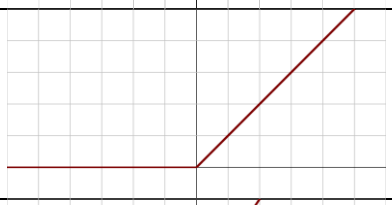
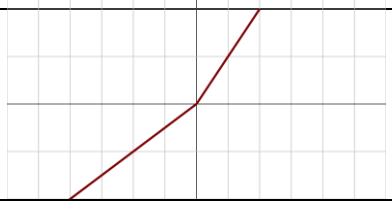
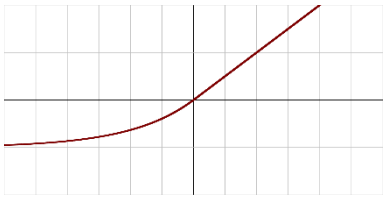
1.3.2 Функції активації

Функція активації нейрона виражає залежність вихідного сигналу штучного нейрона від вхідного [9]. В ній для визначення виходу нейрона сума (від суматора) порівнюється з певним порогом (значенням зазвичай від 0 до 1, чи від -1 до 1). Нижче наведена порівняльна таблиця існуючих функцій активації.

Таблиця 1.1 – Порівняльна таблиця різновидів функцій активації

Назва функції	Формула	Графік
Порогова	$f(x)=\begin{cases}0, x<0\\1, x\geq 0\end{cases}$	
Сигмоїдна	$f(x)=\frac{1}{1+e^{-x}}$	

Продовження таблиці 1.1

Назва функції	Формула	Графік
Гіперболічний тангенс	$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$f(x)=\begin{cases}0, x<0\\ x, x\geq 0\end{cases}$	
LeakyReLU	$f(x)=\begin{cases}a\cdot x, x<0\\ x, x\geq 0\end{cases}$	
ELU	$f(x)=\begin{cases}a\cdot(e^{x}-1), x<0\\ x, x\geq 0\end{cases}$	
SELU	$f(x)=\lambda\begin{cases}a\cdot(e^{x}-1), x<0\\ x, x\geq 0\end{cases}$, $\lambda=1.0507, a=1.6733$	

Розглянемо їх детальніше. Найпростішою є порогова функція активації. Вона повертає 0 для усіх від’ємних значень аргументу і 1 для додатних. Така функція активації корисна, наприклад, для задачі бінарної класифікації, але оскільки вона не диференційована на всій осі абсцис, її не можна застосовувати у мережах, які вимагають проміжних значень на виході нейрона.

Наступною функцією активації, яка наразі широко використовується в нейронних мережах, є сигмоїдна. Вона повертає значення в діапазоні від 0 до 1 і дозволяє перейти від бінарних (цифрових) значень виходів нейрона до дійсних (аналогових). Підвидом сигмоїдної є зміщена (або логістична) сигмоїдна функція.

Визачається як: $f(x) = \frac{1}{1+e^{-\beta \cdot x}}$, де β – параметр нахилу, що визначає крутизну. У граничному випадку вона переходить у порогову.

Функція гіперболічний тангенс повертає значення від -1 до 1. Швидше за сигмоїдну зростає і спадає. Відрізняється від неї і масштабом осей: для сигмоїдної функції точка 0 є точкою насичення, тоді як для гіперболічного тангенса 0 є проміжною, найменш стабільною точкою.

Так як сигмоїдна функція активації не може виразити «силу активації» (наприклад, їй важко відрізнити активацію «силою 5» (0,9933) і «силою 10» (0,9995)), було запропоновано функцію ReLU (rectified linear unit), що виводить вхідний сигнал безпосередньо, якщо він додатний, або нуль у протилежному випадку. Її похідна розраховується простіше і мережа може містити більше нейронів. Для сигмоїдної функції активації у кожен момент часу активується половина нейронів мережі, тоді як ReLU дозволяє досягти розрідженості активації, а тому більш точно відображає процеси, які проходять в мозку людини. Проте і ReLU має певні недоліки. Вона недиференційовна в нулі й має місце проблема «згасаючих елементів»: вихід деяких нейронів під час навчання рівний нулю, вони дезактивуються.

Модифікаціями ReLU є: LeakyReLU, ELU, SELU. LeakyReLU відрізняється від ReLU тим, що має невеликий нахил для від’ємних значень x (коефіцієнт a, що набуває невеликих (зазвичай 0,01) значень). Різновидами LeakyReLU є RReLU і PReLU. В рандомізованій LeakyReLU (RReLU) значення параметра a обирається випадковим чином в заданому діапазоні, а в параметризованій LeakyReLU (PReLU) оптимальне значення a обирається під час навчання мережі.

ELU (експоненційний лінійний елемент) застосовує для відємних знчень x логарифмічну криву. Вона дозволяє досягти швидкого насичення, але має більшу обчислювальну складність, ніж у ReLU чи LeakyReLU.

SELU – масштабований варіант ELU. Вона має властивість самонормалізації, що дозволяє уникнути проблеми зникнення та вибухового зростання градієнта під час навчання.

Ще однією функцією активації, але від декількох змінних, є softmax. Це узагальнення логістичної функції, її доцільно використовувати в задачах класифікації, оскільки вона представляє собою зважену і нормовану на одиницю суму експонент. Задається наступним чином: $f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$, i=1, …, J. Softmax часто застосовується в останньому шарі нейронних мереж.

Вибір конкретної функції активації залежить від поставленої задачі. У багатошаровому персептроні найчастіше використовуються нелінійні функції активації (гіперболічний тангенс або логістична функція).

1.3.3 Навчання мережі

Після задання архітектури мережі необхідно визначити її поведінку. Навчання нейронної мережі полягає у зміні значення ваг нейронів так, щоб досягти поставленої мети задачі. На сьогодні відомо три основні типи навчання нейронних мереж:

а) навчання з вчителем (supervised learning);

б) навчання без вчителя (unsupervised learning);

в) навчання з частковим залученням вчителя (напівконтрольоване навчання, semi-supervised learning).

При навчанні з учителем на вхід мережі подається набір навчальних даних і відповідні виходи. Задача полягає в тому, що мережа має знайти таке загальне правило, яке перетворює входи у виходи.

При навчанні без учителя відома навчальна вибірка, але мережі необхідно визначити внутрішні взаємозв’язки між даними і самостійно сформувати виходи.

За змішаного навчання мережі подають на вхід неповний тренувальний набір, у ньому відсутні деякі вихідні значення. І в цьому випадку мережа спочатку навчається на нерозмічених даних, а потім, використовуючи це наближення, донавчається на розмічених.

Одним з різновидів навчання з учителем є метод зворотного поширення помилки (backpropagation), який широко застосовується для навчання нейронних мереж. Зворотне поширення помилки являє собою ітераційний процес, який починається з останнього шару і рухається у зворотному напрямі через всі шари, поки не буде досягнуто першого. Тобто алгоритм зворотного поширення помилки дає можливість визначити помилку на виході попереднього шару, з огляду на помилку на виході в поточному шарі. Він включає наступні кроки:

а) нехай на входи мережі подано один з можливих образів $x = \{x_i\}_{i=\overline{1,I}}$;

б) позначаємо $y_i^{(0)} = x_i, i = \overline{1,I}$;

в) розраховуємо послідовно значення виходів для n-го шару (n=1,2,…,N):

$$s_j^{(n)} = \sum_{i=1}^{I} y_i^{(n-1)} w_{ij}^{(n)}, \; y_j^{(n)} = f(s_j^{(n)});$$

г) розраховуємо величини $\delta_i^{(N)}$ для нейронів вихідного шару:

$$\delta_i^{(N)} = (y_i^{(N)} - d_i)\frac{dy_i}{ds_i};$$

д) визначаємо $\Delta w_{jk}^{(N)}$;

е) використовуючи рекурентну формулу $\delta_j^{(n)} = \sum_{k=1}^{K} \delta_k^{(n+1)} w_{jk}^{(n+1)} \frac{dy_j}{ds_j}$ розраховуємо $\delta_j^{(n)}$ через $\delta_k^{(n+1)}$ і $\Delta w_{jk}^{(n+1)}$ для всіх попередніх шарів за формулою:

$$\Delta w_{ij}^{(n)} = -\eta \delta_j^{(n)} y_i^{(n-1)};$$

ж) Коректуємо ваги мережі відповідно до формули: $w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t)$;

з) На цьому перша ітерація (t) закінчується;

и) Розраховуємо $E = E(w(t))$; якщо $E(w(t)) < \overline{\varepsilon_{зад}^2}$, то навчання завершується, інакше переходимо до кроку 1 (t+1) ітерації.

Іншим підходом до навчання нейронних мереж є перенесене навчання (transfer learning) – метод машинного навчання, у якому використовується попередньо навчена модель для вирішення нової задачі. Так можна навчати усю мережу з нуля, а можна використати готову мережу і донавчити її виконувати конкретне завдання. При цьому попередньо навчена основа не змінюється, навчаються лише нові додані шари, а тому витрачається менше часу і ресурсів. До того ж, при перенесеному навчанні відбувається накопичення знань (використовуються знання, отримані з попередньо навченої моделі, у ході донавчання отримуються нові).

Також варто зазначити, що при навчанні мережі можуть виникнути проблеми з її недонавчанням чи перенавчанням. Недонавчання трапляється тоді, коли мережа не може вловити тенденцію, що лежить в основі даних. Її точність на перевірочній мережі значно вища за точність на тренувальній. Перенавчання мережі – зайва точна відповідність нейронної мережі до конкретного набору навчальних прикладів [8]. При цьому вона втрачає здатність до узагальнення. Така проблема може виникати у разі довгого навчання мережі або недостатньої кількості навчальних

прикладів. Проблеми недонавчання або перенавчання мережі також можуть виникати через невдало підібрану структуру мережі. Занадто прості нейромережі не здатні адекватно моделювати залежності в реальних задачах. Занадто складні мережі мають надмірну кількість вільних параметрів, які в процесі навчання налаштовуються не тільки на встановлення залежностей, але і на відтворення шуму.

1.3.4 Згорткові нейронні мережі

Застосування багатошарового персептрона при вирішенні задач виявлення об’єктів на зображеннях має певні недоліки. Вхідні зображення зазвичай мають велику розмірність, внаслідок чого зростає число нейронів і синаптичних зв’язків у мережі. Це спричиняє збільшення навчальної вибірки і часу навчання, обчислювальної складності процесу навчання. Тому для аналізу зображень широко застосовуються згорткові нейронні мережі – клас багатошарових штучних нейронних мереж, які застосовують операцію згортки для аналізу зображень.

Ідея цього типу нейронних мереж виникла при ретельному вивченні зорової кори мозку [10]. Окремі нейрони реагують на подразники лише в обмеженій області поля зору – локальному рецептивному полі. Набір таких полів перекривається, охоплюючи всю візуальну область. В згорткових мережах такі поля забезпечують локальну двовимірну зв’язність нейронів.

Особливостями згорткових нейронних мереж є:

1. розріджена зв’язність (оскільки ядро має меншу розмірність, ніж вхід, у згорткових мережах взаємодія зазвичай розріджена; з усього об’єму пікселів вхідного зображення обираються найбільш значущі);

2. розділення параметрів (один і той самий параметр використовується в декількох функціях моделі: у багатошаровому персептроні кожен елемент матриці ваг використовується лише один раз при обчисленні вихідного сигналу, кожен зв'язок має унікальну вагу; а у згорткових мережах кожен елемент ядра застосовується до кожного елемента входу);

3. використання входів змінного розміру;

4. інваріантність до зміщення.

Обмеження згорткових мереж – витяг ознак (оскільки кожен нейрон отримує вхідний сигнал від локального рецепторного поля попереднього шару, витягуються

його локальні ознаки; як тільки ознаку витягнуто, її точне місце розташування не має значення (воно встановлено приблизно відносно інших ознак)).

Згорткові нейронні мережі мають три основні типи шарів (рис.1.5), а саме:

- згортковий шар;
- об’єднуючий шар (шар пулінгу або субдискретизації);
- повнозв’язний шар.

Багатошарові згорткові мережі мають кілька скритих шарів (кожен з яких теж включає згортковий, об’єднуючий, повнозв’язний шари).

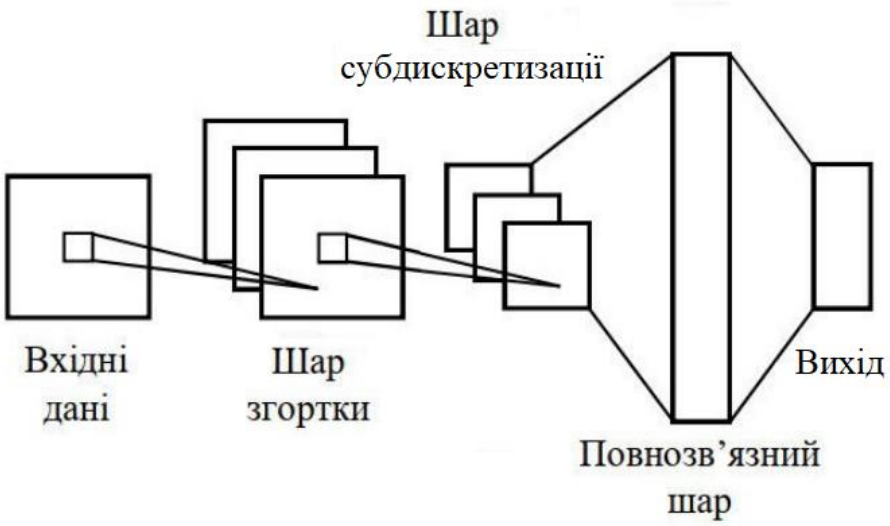


Рисунок 1.5 – Типова архітектура згорткової нейронної мережі

Розглянемо кожен з цих шарів окремо.

1.3.5 Архітектура згорткової мережі

Згорткові шари застосовують операцію згортки до входів мережі. Згортка – особливий вид лінійної операції над двома функціями одного аргументу. Її також можна подати як операцію, яка плавно переміщує одну функцію по іншій і визначається як інтеграл поточкового множення цих функцій:

$$s(t) = \int u(a) \cdot w(t-a)da = (u \cdot w)(t), \tag{1.1}$$

де u – вхід, w – ядро згортки.

Мета процесу згортки – зменшити розмірність карти ознак до такої міри, щоб з повним набором ознак могла працювати мережа прямого поширення (багатошаровий персептрон). Розглянемо приклад операції згортки. Нехай вхідне зображення можна подати у вигляді матриці розміром 3×4, ядро має розмірність 2×2. Початкове зображення покривається вікном такої розмірності, як і ядро (у даному випадку 2×2). Кожна з матриць: вікно і ядро, витягуються по рядкам у вектори. Далі ці вектори перемножуються. Такі ж дії повторюються і для наступних вікон (рис.1.6).

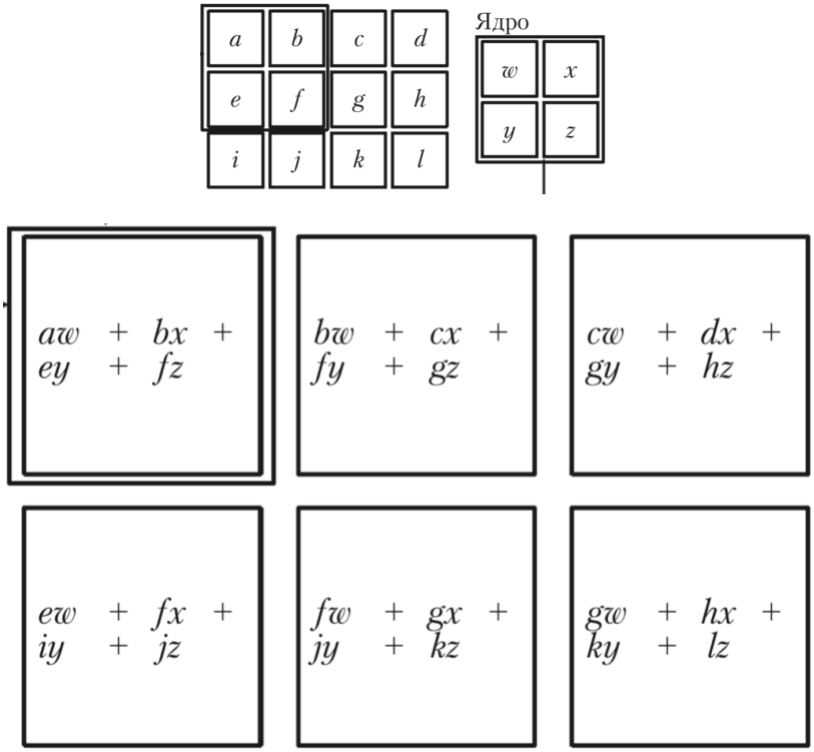


Рисунок 1.6 – Приклад операції згортки з кроком 1

Фільтри – багатовимірні матриці ваг зв’язків нейронів попереднього шару з нейронами згорткового шару [11]. Кожен фільтр визначає на зображенні певну властивість, їх ядра ініціалізуються випадковими значеннями. Тип фільтра визначає, для виділення яких ознак зображення він використовується, наприклад, виділення прямих ліній під певним кутом (у процесі навчання мережі ваги фільтрів змінюються, впливаючи на характер ознак, що ними виділяються). Зазвичай

використовуються квадратні фільтри. Кількість використовуваних фільтрів визначає глибину згорткового шару.

Результати згортки, які визначають місце розташування ознак вихідного зображення, називаються картами ознак. Такі карти мають форму площини, на якій усі нейрони спільно використовують одну і ту саму множину ваг.

Згортковий шар має й інші параметри. Зміщення (stride) – відстань між двома послідовними рецепторними полями [11]. Визначає на скільки пікселів зміщується фільтр при формуванні сигналу наступного нейрона згорткового шару.

Додавання нулів (padding): додає нейрони з нульовими значеннями по краях попереднього шару. Це дозволяє управляти шириною ядра і розміром виходу. Способи доповнення нулями входу згортки:

1. коректна (valid) згортка: доповнення нулями не використовується зовсім; ядро займає тільки ті позиції, де воно цілком вміщується всередині зображення;

2. конгруентна (same) згортка: доповнення такою кількістю нулів, щоб розмір виходу дорівнював розміру входу; не накладається обмеження на кількість шарів, пікселі біля межі шару (ближче до краю) менше впливають на вихідні сигнали, ніж центральні;

3. повна (full) згортка: доповненя такою кількістю нулів, щоб кожен піксель «відвідувався» k разів; вихідні пікселі поблизу межі є функціями меншого числа вхідних пікселів, ніж пікселів у центрі; тут важко навчити єдине ядро, щоб воно добре функціонувало в усіх позиціях карти ознак.

Обчислення виходу нейрона згорткового шару проводиться за формулою (1.2)

$$z_{i,j,k} = b_k + \sum_{u=0}^{a-1} \sum_{v=0}^{b-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \,, \tag{1.2}$$

де $i' = i \cdot s_h + u$; $j' = j \cdot s_w + v$; $z_{i,j,k}$ – вихід нейрона, розташованого в і-му рядку, j-му стовпці карти ознак з номером k; $f_{n'}$ – кількість карт ознак попереднього шару; b_k – зміщення карти ознак з номером k.

Для формування вихідного сигналу нейронів згорткового шару в згорткових мережах використовуються функції активації: ReLU, сігмоїдна, гіперболічний тангенс (див. п. 1.3.2).

Шар пулінгу (субдискретизації) застосовується для зменшення розмірності зображення. Він допомагає зменшити складність та обмежити ризик перенавчання мережі. Шар субдискретизації перетворює сигнали згорткового шару, виділяючи найбільш значущі за певними критеріями. Виконується узагальнення шуканих ознак (втрачається частина інформації про місцерозташування ознак, але зменшується розмірність). Задача шару пулінгу – зробити представлення зображення локально інваріантним відносно малих паралельних перенесень входу.

У цьому шарі використовується вікно певного розміру (U) для виділення областей нейронів попереднього шару, які будуть пов’язані з нейроном шару субдискретизації. Потім обирається і розраховується один сигнал нейрона шару субдискретизації.

Для формування сигналу цього шару використовують або максимальне значення серед сигналів попереднього шару (maxpooling), або шляхом розрахунку середнього значення (averagepooling). Процес формування сигналів шару субдискретизації показаний на рисунку 1.7.

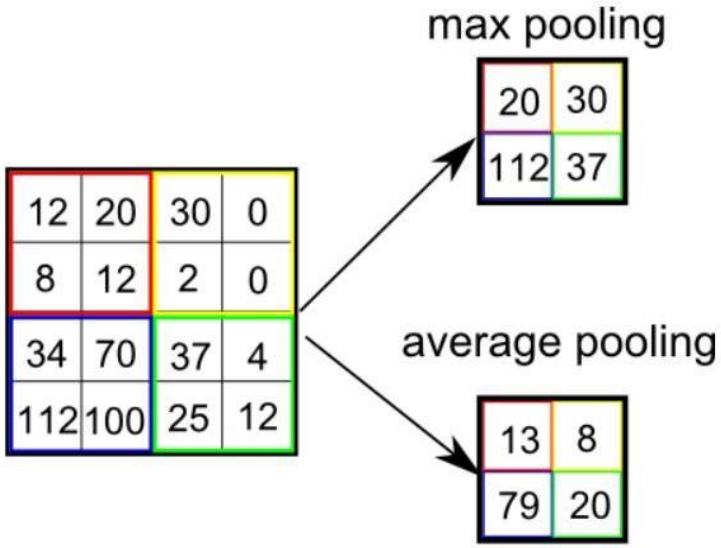


Рисунок 1.7 – Операція пулінгу з кроком 2

Використовують також і проміжний варіант між максимумом і середнім значенням: максимум береться не за усім вікном, а за деякою його випадковою підмножиною (L_2-норма в прямокутному околі або зважене середнє з вагами, що залежать від відстані до центрального пікселя).

Повнозв'язний шар формує вихід нейромережі. Він є одновимірним, у ньому кожен нейрон наступного шару пов'язаний з кожним нейроном попереднього шару на всіх рівнях (якщо у попереднього шару присутній параметр глибини). Основне його призначення – перетворення сигналів, отриманих на згорткових рівнях мережі, в одновимірне представлення і виділення ознак на одновимірному рівні. А якщо поставленою задачею є класифікація, повнозв'язний шар видає приналежність вхідного зображення до відповідного класу.

1.3.6 Алгоритм роботи мережі

Розглянемо послідовність роботи згорткової мережі. Наприклад, дано кольорове зображення розміром 32 на 32 пікселі, застосовується 12 фільтрів:

1. Вхідні дані містять інформацію про зображення (в даному випадку ширина 32, висота 32, 3 кольорові канали R, G, B).

2. Згортковий шар перемножує значення фільтра на вихідні значення пікселів зображення (поелементне множення), після чого всі добутки сумуються (рис.1.8):

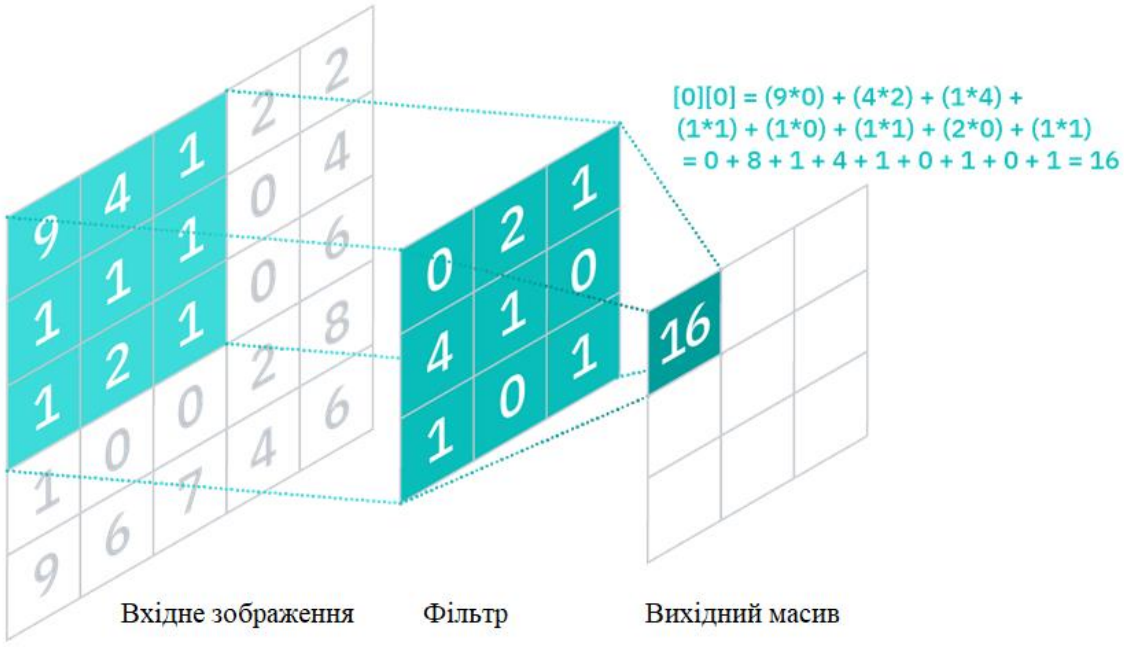


Рисунок 1.8 – Згортка зображення

3. Шар пулінгу зменшує обсяг до [16×16×12]. Подібно до згорткового шару, він об’єднує фільтр по всьому входу, основна різниця в тому, що цей фільтр не має ваг. Замість цього ядро застосовує функцію агрегації, заповнюючи вихідний масив. Існує два основні типи шарів пулінгу: max pooling (під час переміщення фільтра по входу він обирає піксель з максимальним значенням) і average pooling (коли фільтр переміщується по входу, він обчислює середнє значення).

4. Повнозв’язний шар виводить N-мірний вектор (N — кількість класів) шляхом звернення до виходу попереднього шару (карти ознак) та визначення властивостей, які найбільш характерні для певного класу.

1.3.7 Порівняння типових архітектур ЗНМ

Найпершою із запропонованих згорткових нейронних мереж була LeNet 5 (Ян Лекун, 1998). Її архітектуру наведено на рисунку 1.9.

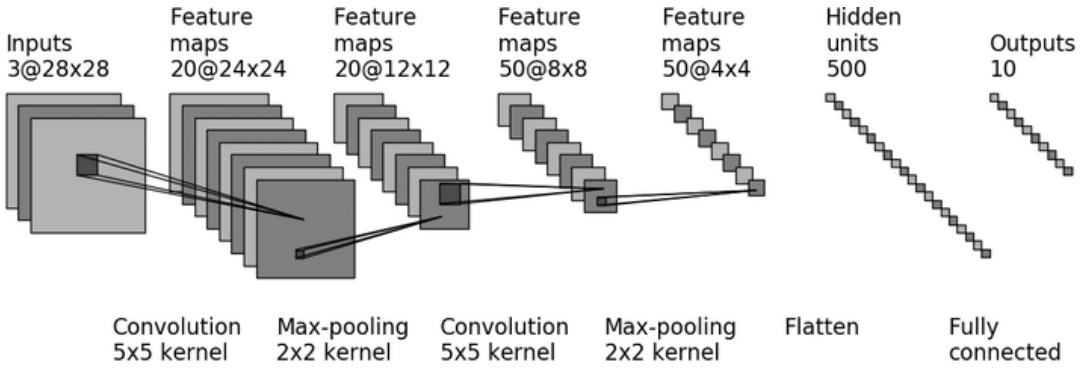


Рисунок 1.9 – Архітектура LeNet 5

LeNet 5 складається з двох наборів згорткових і шарів об’єднання. За ними слідує згортковий шар, потім два повноз’єднані шари і softmax класифікатор. Зображення на вході доповнюється нулями до розміру 32×32 і нормалізується. Шари пулінгу виконують об’єднання за середнім (кожен нейрон обчислює середнє значення входів, результат множиться на коефіцієнт, додається зміщення і застосовується функція активації). Вихідний шар мережі: замість обчислення добутку входів і вектора ваг, кожен нейрон видає квадрат евклідової відстані між вектором входів і вектором ваг. У LeNet 5 за функцію активації береться перехресна ентропія, оскільки вона породжує великі градієнти, а тому прискорює сходження до мінімуму.

Наступна нейронна мережа, AlexNet, перевершила LeNet 5 з великим відривом у змаганні на ImageNet (наборі даних мільйонів розмічених зображень з високою роздільною здатністю, що відносяться приблизно до 22 000 категорій). Запропонована у 2012 році Алексом Крижевським, Іллею Сатскевером і Джеффрі Хінтоном, є більш глибокою у порівнянні з LeNet 5 [12]. Її архітектуру наведено на рисунку 1.10.

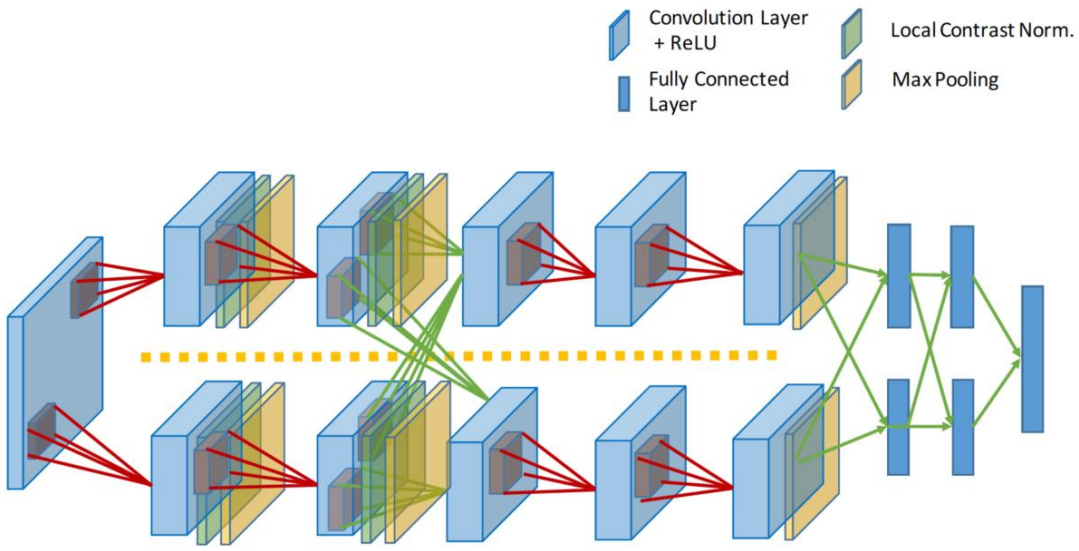


Рисунок 1.10 – Будова AlexNet

Її відмінністю від LeNet 5 є те, що згорткові шари накладаються один на одного замість розташування об'єднуючого шару поверх згорткового. Аби скоротити ризик перенавчання мережі також використано прийоми регуляризації:

1. застосування дропауту (dropout, метод регуляризації штучних нейронних мереж, що полягає у виділенні випадковим чином (з вказаною імовірністю) із загальної мережі підмережі, в рамках якої оновлюються ваги під час навчання);

2. доповнення даних (зсув, поворот навчальних зображень).

Ще однією особливістю AlexNet є використання змагальної нормалізації (локальної нормалізації відповіді після кроку ReLU). Ця нормалізація змушує нейрони, які найсильніше активуються, пригнічувати нейрони на тому самому місці, але в сусідніх картах ознак. Дана операція стимулює різні карти ознак спеціалізуватися і примушує мережу досліджувати більш широкий діапазон ознак.

Мережа AlexNet довела, що глибокі згорткові мережі можуть досягти достатньо точних результатів при роботі із зображеннями. Тому подальшою спробою її вдосконалення стали мережі GoogLeNet і VGGNet. GoogLeNet посіла перше місце у змаганні на Large Scale Visual Recognition Challenge (ILSVRC, що є підмножиною ImageNet із приблизно 1000 зображень, що відносяться до 1000 категорій) у 2014 році з частотою помилок 6,7%. Це стало можливим завдяки

додаванню модулів inception [13], які дозволяють мережі обирати розміри згорткових фільтрів у кожному блоці (рис.1.11).

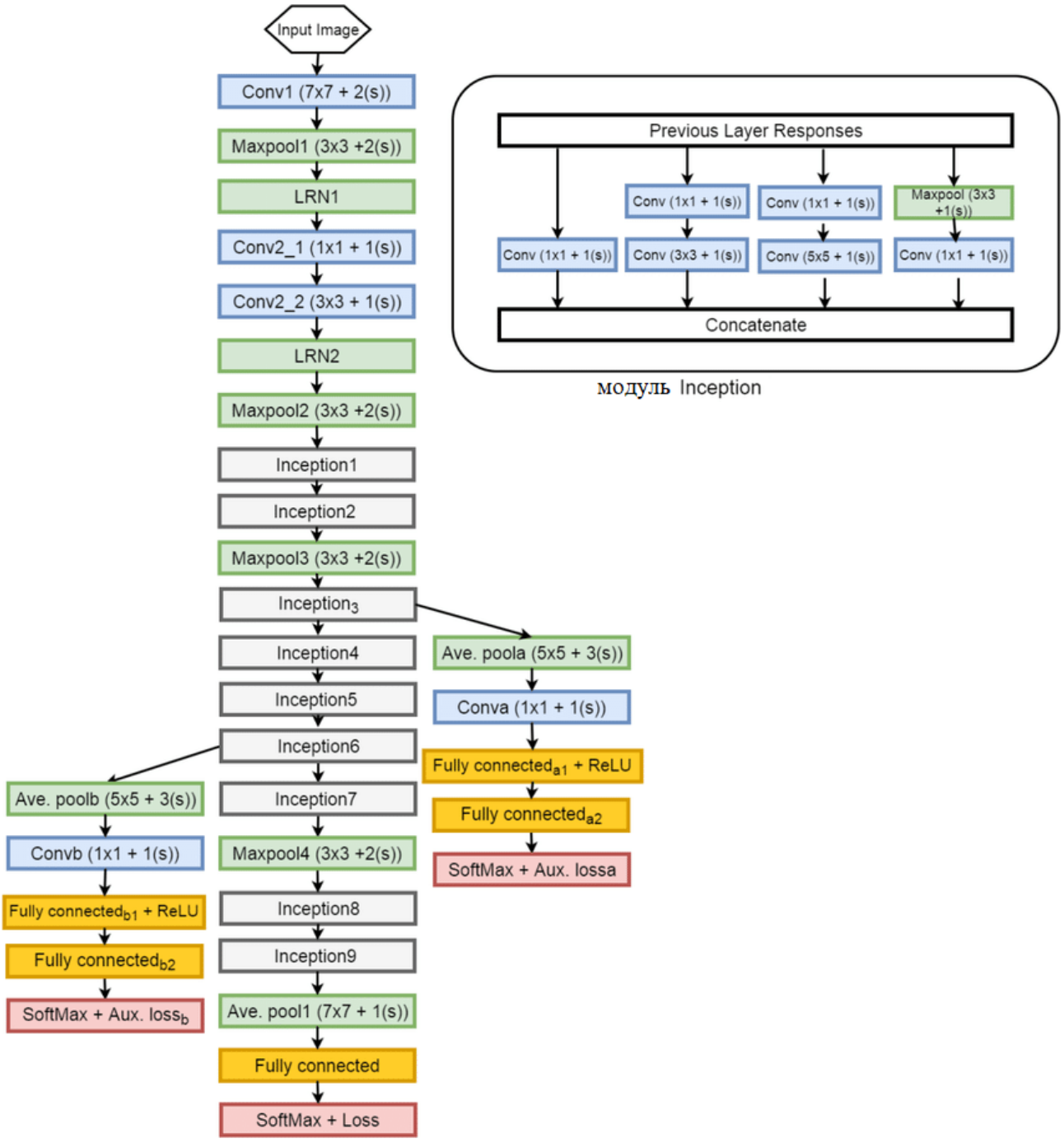


Рисунок 1.11 – Будова GoogLeNet і модуля inception

Такі модулі являють собою паралельно з’єднані згорткові шари (з різними фільтрами), які у подальшому об’єднуються в один. Для уникнення зростання числа

параметрів, перед кожним шаром згортки використовується згортка 1×1, яка зменшує число карт ознак.

Також в GoogLeNet, на відміну від решти архітектур згорткових мереж, замість повнозв'язного шару в кінці мережі використовується шар субдискретизації за середнім (averagepooling).

Недолік занадто великої кількості гіперпараметрів AlexNet був вирішений мережею VGGNet шляхом заміни великих фільтрів (11 і 5 у першому і другому шарах згортки відповідно) кількома фільтрами розміром 3×3. Архітектура, розроблена Симоньяном і Зіссерманом [14], складається з фільтрів розміром 3×3, шару субдискретизації з кроком 1, конгруентним доповнення нулями для збереження розмірності (рис.1.12).

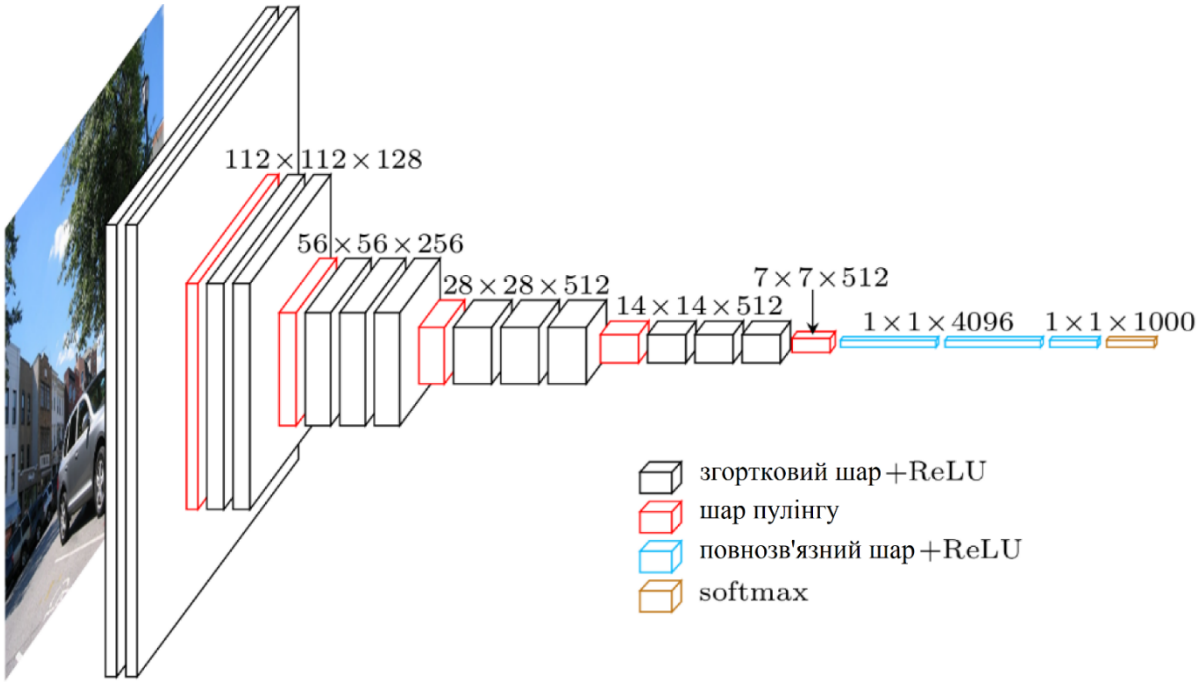


Рисунок 1.12 – Архітектура VGGNet

Всього мережа має 16 шарів: вхідне зображення розмірністю 224×224×3, 5 пар згорткових шарів (фільтри: 64, 128, 256, 512) і шари субдискретизації (max pooling). Вихідні дані цих шарів подаються на три повнозв'язні шари із функцією softmax у вихідному шарі. Але VGG має два недоліки:

1. дуже повільна швидкість навчання;

2. архітектура мережі занадто складна (з’являються проблеми з пропускною спроможністю).

Переможцем у змаганні на ImageNet у 2015 році була мережа ResNet (Residual Network), показавши частоту помилок нижче 3,6% (для порівняння AlexNet – 16%). ResNet є дуже глибокою мережею, вона місить 152 шари [15]. Ключем до ефективності настільки глибокої мережі є використання «обхідних зв’язків»: вхідний сигнал подається в шар мережі і до вихідного шару. Під час ініціалізації звичайної мережі її ваги близькі до нуля. При додаванні обхідного зв’язку, результуюча мережа видає копію входів (моделює функцію тотожності). У випадку, коли цільова функція близька до функції тотожності, навчання мережі прискорюється.

Тож, в цілому ResNet можна розглядати як глибоку залишкову мережу, тобто послідовність залишкових елементів (рис.1.13).

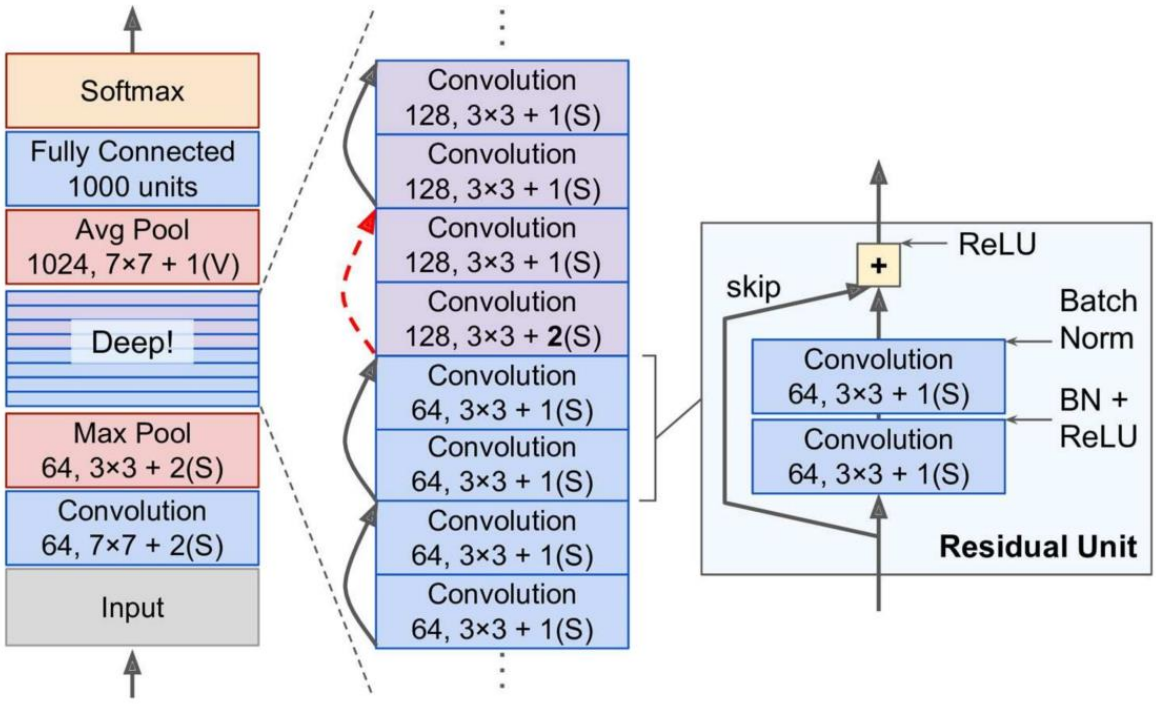


Рисунок 1.13 – Будова ResNet і залишкового елемента

Кожен такий залишковий елемент являє собою невелику нейронну мережу з обхідним зв’язком і складається з двох згорткових шарів з ядрами 3×3, функцією

активації ReLU, страйдом 1 і доповненням нулями SAME (для збереження розмірності зображення) (рис.1.13). На даний момент існує багато варіацій ResNet, але основна їх ідея полягає у використанні вихідних даних згорткового шару як вхідного параметра.

Цікавою є і архітектура SENet, запропонована у 2017 році, що досягла частоти помилок 2% [16]. Ключовою її особливістю є використання SE блоків «стиснення» (squeeze) і «збудження» (excitation) з метою посилення корисних властивостей (ознак) і стиснення непотрібних (рис.1.14). Кожен канал на вході згорткового шару стикається до одного числового значення за допомогою об’єднання за середнім. Повнозв’язний шар, за яким слідує функція ReLU, додає нелінійність. Другий повнозв’язний шар із сигмоїдною функцією активації надає кожному каналу згладжену функцію. Насамкінець відбувається зважування кожної карти ознак (рис.1.14).

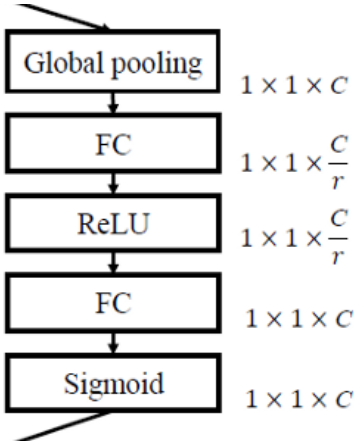


Рисунок 1.14 – Будова блока SE

Такі блоки можна навіть поєднувати із залишковими блоками ResNet і модулем inception від GoogLeNet у окрему мережу, наприклад SE-ResNet-50 [16].

1.4 Висновок до розділу

Таким чином, незважаючи на значний успіх методу Віоли-Джонса у задачі розпізнавання об’єктів, він навчається повільніше у порівнянні з іншими методами, оскільки використовує значну кількість ознак. HOG працює з локальними комірками, а тому інваріантний до геометричних і фотометричних перетворень об’єкта. Але все одно традиційні методи виявлення об’єктів краще підходять для задач, де набір тренувальних прикладів невеликий. Для великих наборів даних (при розмірі датасету більше ніж 1000 зображень) краще застосовувати нейронні мережі, які завдяки імітації роботи кори головного мозку людини краще справляються із задачами обробки зображень.

Найважливішою особливістю нейронних мереж, яка дозволяє успішно використовувати їх у різноманітних задачах (кластеризація, класифікація, розпізнавання образів, прогнозування і т. д.), полягає у паралельній обробці інформації усіма ланками, що дозволяє значно прискорити процес обробки інформації. Інша, не менш важлива властивість, – здатність до навчання. Згорткові нейронні мережі ефективно застосовуються у задачі розпізнавання об’єктів у режимі реального часу завдяки можливості до узагальнення накопичених знань і врахуванню інформації про співвідношення частин зображення між собою. Тому для вирішення підзадачі виявлення об’єктів у роботі було обрано згорткові нейронні мережі. А для можливості урахування зв’язку між відеокадрами буде створено окремий алгоритм на основі виявлення.

2 АНАЛІЗ ІСНУЮЧИХ ЗГОРТКОВИХ МЕРЕЖ ДЛЯ ВИЯВЛЕННЯ ОБ'ЄКТІВ

На сьогоднішній день існує багато різноманітних мереж для знаходження об'єктів на зображеннях чи у відеокадрах. Усі вони базуються на двох підходах – одно і двоступеневому виявленні об'єктів. Двоступеневі детектори спочатку визначають регіони (області), у яких можуть знаходитись об'єкти, а потім шукають їх саме у цих регіонах. Такі мережі мають високу точність розпізнавання. А одноступеневі детектори, що одночасно виконують локалізацію та класифікацію об'єктів у всіх частинах зображення за один прохід, демонструють високу швидкість передбачення (висновку).

Двоступеневими детекторами є мережа R-CNN (Region-based Convolutional Neural Networks) та її похідні (Fast R-CNN, Faster R-CNN, Mask R-CNN, Mesh R-CNN), а також RepPoints [17], що не використовує якоря (anchors). Одноступеневими детекторами, які використовують якоря є YOLO, SSD і RetinaNet. Серед тих, що не використовують якоря, вирізняються мережі CenterNet, CornerNet та FCOS.

Для виявлення об'єктів у відеопотоці двоступеневі детектори R-CNN і Fast R-CNN не підходять. Хоча Faster R-CNN стала першою нейронною мережею яка працює майже в режимі реального часу (17 кадрів на секунду). Тому в даній роботі для проведення порівняльного аналізу обрано мережі Faster R-CNN, YOLO, SSD, RetinaNet.

Створено програмний код мовою Python у середовищі Google Colaboratory для порівняння якісних характеристик нейронних мереж на датасеті HABBOF. Було використано операційну систему Windows 10 та серверний графічний процесор Tesla T4.

Нижче наведена коротка характеристика мереж Faster R CNN, YOLO, SSD, RetinaNet та результати їх порівняльного аналізу.

2.1 Faster R-CNN

Звичайні згорткові мережі не підходять для прямого виявлення об’єктів, так як кількість входжень шуканих об’єктів не є фіксованою і розмір вихідного шару є змінним. Одним з підходів до вирішення цієї проблеми є вибір певної області із зображення та використання згорткової мережі для класифікації присутнього в ній об’єкта. Але об’єкти по-різному розташовуються на зображенні та мають різні співвідношення сторін, а отже треба обирати величезну кількість областей, що може призвести до помилки обчислень.

Для уникнення великої кількості можливих областей на зображенні у 2014 році було розроблено мережу Region-based Convolutional Neural Network (R-CNN) [18]. Запропоновано використати алгоритм селективного пошуку (selective search). Селективний пошук є ресурсовитратним і виявляє низьку точність, а тому не може бути використаний для знаходження об’єктів самостійно. Тим часом він дозволяє мережі згенерувати обмежену кількість регіонів інтересу (RoI), які містять шукані об’єкти з більшою ймовірністю, ніж випадково обрані регіони. Цей метод ділить зображення на ~2000 регіонів (так звані, регіони пропозицій), у яких знаходяться потенційні об’єкти.

Тож, архітектура R-CNN складається з трьох модулів (рис. 2.1): спершу зображення на вході розбивається на регіони за методом селективного пошуку; далі відбувається вилучення ознак з кожного отриманого регіону за допомогою згорткової нейронної мережі (наприклад, AlexNet); ознаки класифікуються за методом опорних векторів (SVM, Support Vector Machine), а межі регіонів уточнюються за допомогою лінійної регресії.

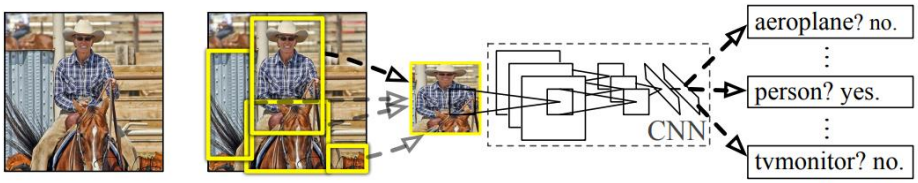


Рисунок 2.1 – Приклад обробки зображення в R-CNN

Але R-CNN має наступні недоліки:

1. вимагає багато часу на навчання, не пристосована для обробки відео;

2. селективний пошук не є алгоритмом машинного навчання, тому можуть виникнути проблеми із виявленням потенційних об’єктів на різних зображеннях.

Недоліки R-CNN призвели до створення у 2015 році покращеної Fast R-CNN мережі [19]. Вона використовує R-CNN як основу з наступними відмінностями:

1. R-CNN використовує spatial pyramid pooling (SPP, шар пулінгу, завдяки якому згорткова мережа не вимагає вхідного зображення фіксованого розміру) [20] поверх останнього згорткового шару; Fast R-CNN замість SPP використовує об’єднання регіонів інтересу (Region of Interest pooling, RoI pooling); тобто на кожен регіон накладається сітка і застосовується maxpooling для зменшення розмірності, в результаті чого усі регіони потенційних об’єктів мають фіксовану розмірність;

2. замість SVM класифікатора використовується softmax класифікатор; основною відмінністю цих класифікаторів є їхня функція втрат: SVM будує гіперплощину у просторі, яка розділяє точки різних класів так, щоб найближчі до гіперплощини знаходились якнайдалі від неї та у якості функції втрат використовує завісні втрати (hinge loss); softmax визначає ймовірність приналежності об’єкта до певного класу і за функцію втрат обирає перехресну ентропію;

3. тренування відбувається в одну стадію, під час тренування оновлюються усі рівні мережі; це сприяє досягненню більшої точності.

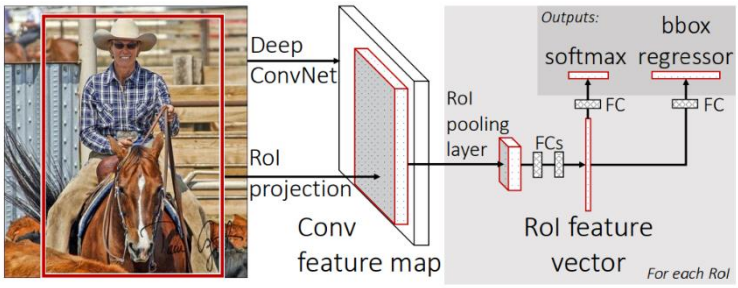


Рисунок 2.2 – Приклад обробки зображення у Fast R-CNN

Fast R-CNN показує більш високу точність і менший час обробки, ніж R-CNN, так як на шар згортки подаються не всі регіони пропозицій. Тим не менш, удосконаленням Fast R-CNN стала мережа Faster R-CNN, у якій замість ресурсовитратного селективного пошуку регіонів пропозицій було впроваджено новий спосіб локалізації об'єкта – RPN (Region Proporsal Network) [21]. В основі такого підходу лежить система якорів (anchor boxes, що є комбінацією центра ковзного вікна, масштабу та співвідношення сторін).

Архітектура Faster R-CNN представлена на рисунку 2.3:

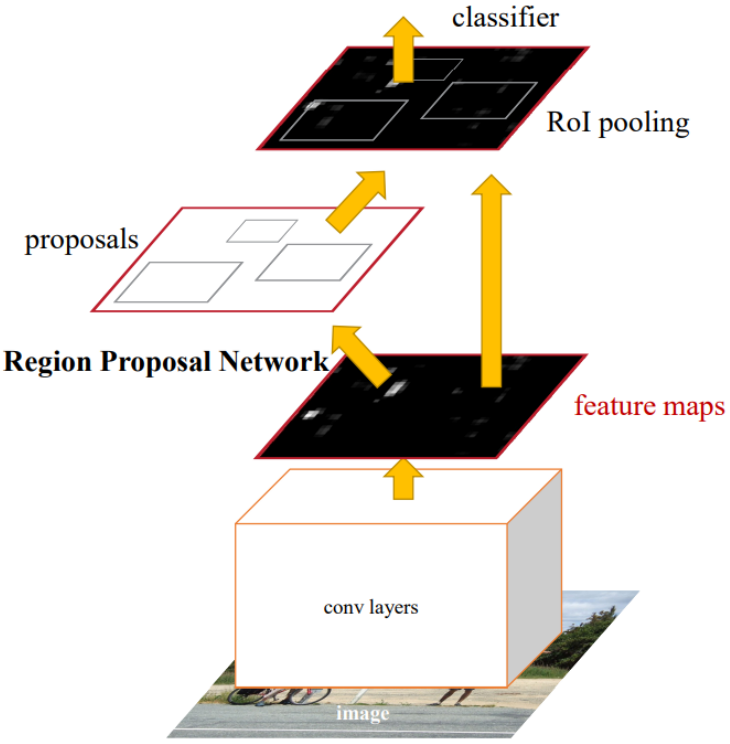


Рисунок 2.3 – Архітектура Faster R-CNN

Для зображення, що подається на вхід мережі, формується карта ознак, яка надалі обробляється шаром RPN (рис. 2.4). Ковзне вікно (sliding window) проходить картою ознак. Центр ковзного вікна пов'язаний із центром якоря. Використовується метрика IoU (Intersection over Union, також відома як індекс Жаккара або подібність Жаккара): міра, яка визначає ступінь перетину областей (у випадку Faster R-CNN – якорів) і використовується для оцінки подібності між двома об'єктами. У задачі виявлення об'єктів IoU відповідає перетину площі двох обмежувальних рамок,

поділеної на суму цих площ. На основі неї та розмічених прямокутників визначається, чи є об’єкт у даній області зображення.

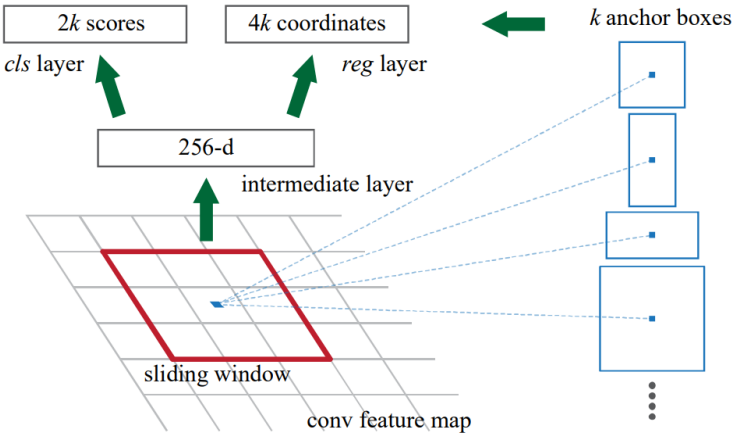


Рисунок 2.4 – Region Proporsal Network

Далі карта ознак разом з отриманими об’єктами передаються шару RoI з подальшою класифікацією, а також із визначенням зміщення місцеположення потенційних об’єктів.

Функція втрат, що використовується у Faster R-CNN, поєднує втрати класифікації та регресії обмежувальної рамки:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \sum_i p_i^* \cdot L_1^{smooth}(t_i - t_i^*), \quad (2.1)$$

де i – індекс прив’язки (якоря) в міні-батчі; p_i – прогнозована ймовірність того, що якір є об’єктом; мітка істинності p_i^* дорівнює 1, якщо якір позитивний, і 0, якщо негативний; t_i – вектор, що представляє 4 координати передбаченої обмежувальної рамки, а t_i^* – це справжній прямокутник навколо об’єкта, пов’язаний із позитивним якорем; L_{cls} – перехресна ентропія за двома класами (об’єкт і не об’єкт).

Модель Faster R-CNN, в цілому, гірше справляється з локалізацією об’єктів, але працює швидше за Fast R-CNN. ЇЇ модифікацією для вирішення задачі сегментації зображень є мережа Mask R-CNN [22].

2.2 YOLO

Модель R-CNN та їй подібні досягають значної точності (для прикладу Fast R-CNN з mAP 66% на PASCAL VOC2012 [19]), але працюють дуже повільно і тільки Faster R-CNN може бути застосована для роботи в режимі реального часу. Такі мережі використовують окремі регіони для локалізації об’єкта на зображенні, тобто не враховують повної інформації про зображення.

Одноступеневі детектори є набагато швидшими, хоча і менш точними за мережі R-CNN. Перша версія YOLO (You Look Only Once) була представлена у 2016 році [23]. Сучасні версії обробляють зображення зі швидкістю до 30 кадрів/с і мають mAP 57,9% на датасеті COCO (Microsoft Common Objects in Context). Ключовою особливістю YOLO є те, що вона розглядає виявлення об’єктів як проблему регресії просторово розділених обмежувальних рамок і пов’язаних імовірностей класу. Вхідне зображення за один раз повністю проходить через нейронну мережу й одразу виявляються об’єкти.

Мережева архітектура YOLO натхненна моделлю GoogLeNet для класифікації зображень. У ній наявні 24 згорткові, а потім 2 повноз’єднані шари (рис. 2.5). Замість модулів inception, як у GoogLeNet, YOLO використовує редукційні шари 1×1 з подальшими 3×3 згортковими.

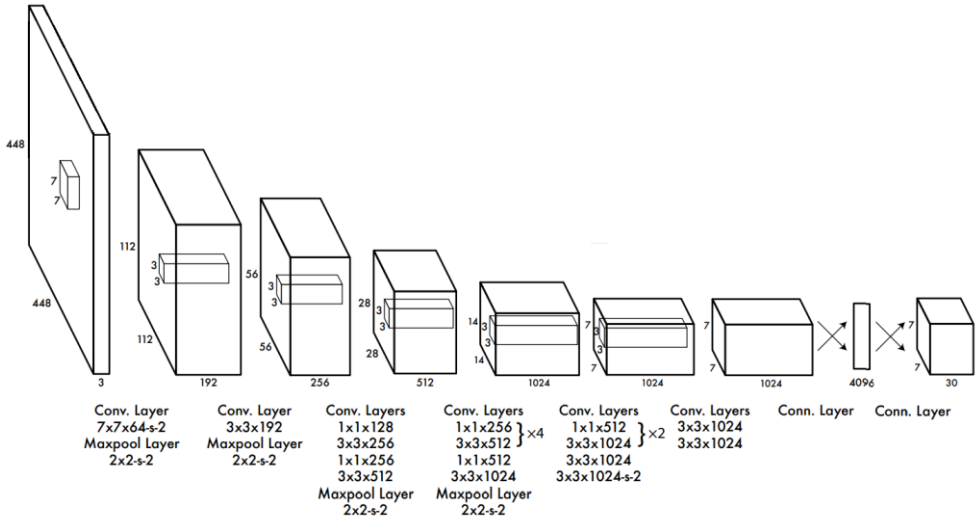


Рисунок 2.5 – Будова YOLO

Алгоритм роботи YOLO полягає в наступному: зображення розбивається на сітку з комірок, які є своєрідними якорями, до яких прикріплюються декілька обмежувальних рамок. Для кожної з них мережа виводить значення п’яти параметрів (рис. 2.6): координати (x, y) центра рамки відносно меж комірки (c_x, c_y); ширина та висота (w, h) рамки відносно усього зображення; показник впевненості (confidence) (ймовірність того, що в даній комірці міститься об’єкт).

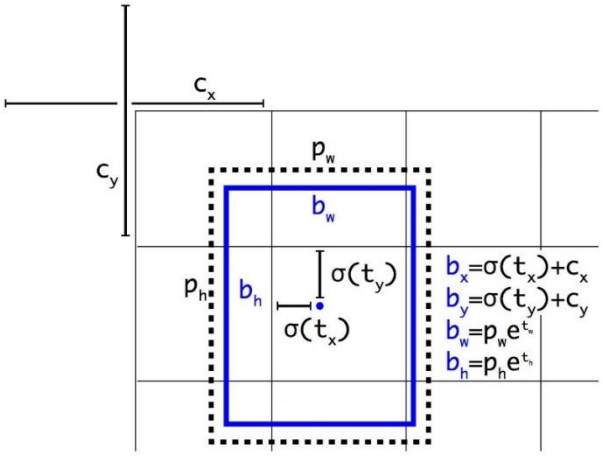


Рисунок 2.6 – Параметри обмежувальної рамки

Обмежувальні рамки, імовірність класу яких перевищує деяке порогове значення, обираються для визначення місцеположення об’єкта на зображенні. Для того, щоб позбутись дублікатів прогнозованих рамок застосовується метод немаксимального придушення (non-maximum suppression, NMS): відбираються рамки з максимальним значенням показника впевненості, решта ігноруються.

YOLO використовує квадратичну похибку між прогнозованими і справжніми значеннями для розрахунку функції втрат (2.2), яка складається з:

1. втрати класифікації (останній доданок; якщо об’єкт виявлено, вона у кожній комірці є квадратом помилки умовних ймовірностей кожного класу);

2. втрати локалізації (перші два доданки);

3. втрати впевненості (якщо об’єкт виявлено, третій доданок не нульовий, а якщо немає об’єкта – четвертий доданок не нуль; через те, що більшість рамок не

містять жодних предметів, виникає проблема дисбалансу класів, додається коефіцієнт λ_{noobj}, який за замовчуванням 0,5).

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

Обмеження YOLO полягає в тому, що вона погано працює з дрібними об’єктами на зображенні. Наступна версія YOLOv2 була випущена роком пізніше, у 2017 [24]. Нова модель використала мережу Darknet-19 (що містить 19 шарів згортки та 5 шарів максимального об’єднання) як базову. YOLOv2 підвищує середню точність за допомогою пакетної нормалізації. Ще одним доповненням до першої версії YOLO стало додавання блоків якорів. YOLO передбачає наявність одного об’єкта в комірці. Це стає незручним при наявності більшої кількості об’єктів. YOLOv2 уникає цього завдяки тому, що передбачає п’ять обмежувальних рамок для кожної комірки.

У 2018 році з’явилася третя версія – YOLOv3 (на основі Darknet-53), яка складається з 106 згорткових шарів і з високою точністю розпізнає невеликі об’єкти на зображенні [25]. Дозволяє прогнозувати обмежувальні рамки в трьох різних масштабах, причому карти ознак для цього витягуються на шарах 82, 94 і 106. Завдяки цьому YOLOv3 компенсує недоліки YOLOv2 і YOLO у виявленні дрібних об’єктів. Також, на відміну від попередників, YOLOv3 передбачає наявність трьох (замість п’яти у попередній версії) обмежувальних рамок на комірку, але робить це в різних масштабах. Тобто загалом до дев’яти опорних рамок.

Наступні версії YOLO не вважаються офіційними. Тим не менш, у 2020 році запропоновано YOLOv4 [26, 27]. Базується на архітектурі SPDarknet-53 і запроваджує такі концепції, як зважені залишкові з’єднання (Weighted Residual Connections, WRC), міжстадійні часткові з’єднання (Cross-Stage-Partial connections, CSP), крос міні пакетну нормалізацію (cross mini-batch normalization, CmBN),

самонавчання (self-adversarial training, SAT). У результаті нововведень YOLOv4 досягла на 10% вищої середньої точності та на 12% кращого показника кадрів за секунду порівняно з YOLOv3.

2.3 SSD

Іншою, у чомусь подібною до YOLO, мережею для виявлення об’єктів у режимі реального часу є SSD (Single Shot MultiBox Detector) [28]. У даному випадку Single Shot означає, що задачі локалізації та класифікації об’єктів виконуються за один прохід мережі. SSD була однією з перших одноступеневих детекторів, що досягли відносно близької до двоступеневих детекторів точності, зберігши при цьому здатність працювати в режимі реального часу. MultiBox – методика пошуку обмежувального прямокутника, Detector: нейронна мережа виявляє об’єкти.

Головною перевагою мережі стала більша точність у виявленні дрібних об’єктів порівняно з іншими одноступеневими детекторами: 77% на датасеті Pascal VOC2007 завдяки виявленню об’єктів у різних масштабах. Вона більше підходить для відеокриміналістики, юридичного виявлення та виявлення орієнтирів.

SSD в цілому складається з двох частин: попередньо натренованої мережі для витягу карт ознак і декількох шарів згортки для виявлення об’єктів. Оригінальна архітектура SSD, представлена на рисунку 2.7, базується на VGG-16, хоча за основу можна взяти іншу мережу.

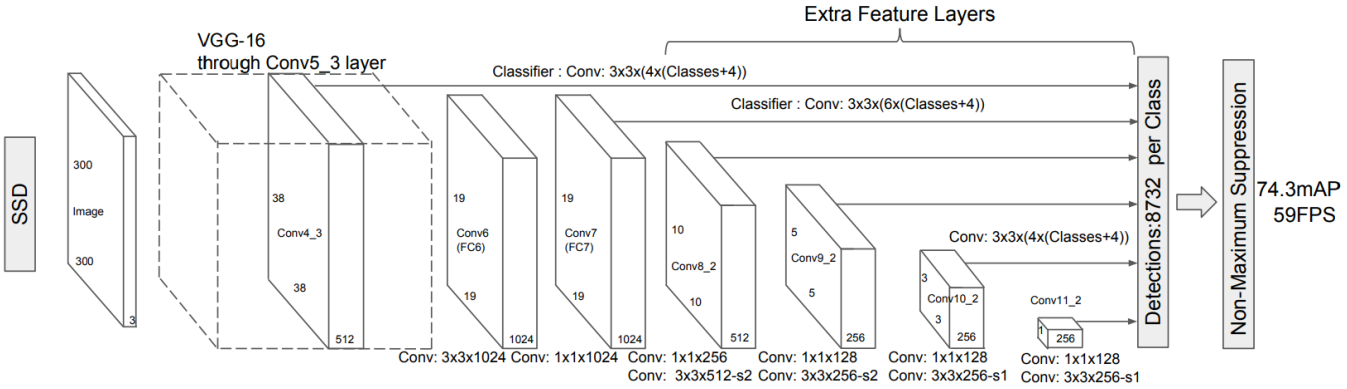


Рисунок 2.7 – Архітектура SSD

VGG-16 використана як базова, оскільки вона демонструє високу продуктивність у задачах класифікації зображень високої роздільної здатності. Але замість повнозв'язних шарів VGG був доданий набір допоміжних згорткових шарів, що дозволило витягувати об'єкти в декількох масштабах і поступово зменшувати розмір вхідних даних до кожного наступного шару. Аби зменшити кількість виявлених обмежувальних рамок і видалити дублікати в кінці застосовується техніка немаксимального придушення (non-maximum suppression).

Як і YOLO, SSD ділить зображення на сітку, кожна комірка сітки відповідає за виявлення об'єктів у даній області зображення. SSD має рецептивне поле (receptive field), завдяки якому виявляє об'єкти в різних масштабах, та використовує обмежувальні рамки за замовчуванням, еквівалентні якорям у Faster R-CNN. Тоді як в YOLO використовується кластеризація k-середніх у навчальному наборі даних аби визначити рамки за замовчуванням. Прогноз для обмежувальних рамок і достовірності для різних об'єктів на зображенні здійснюється за допомогою кількох карт ознак різних розмірів (в кількох масштабах). Загалом SSD використовує 8732 обмежувальні рамки за замовчуванням. Під час навчання вони зіставляються за співвідношенням сторін, місцем розташування та масштабом із справжніми рамками. З усіх прямокутників обираються ті, що мають найбільше перекриття з шуканими обмежувальними рамками (тобто з IoU більшим за 0,5).

Використовується функція втрат, що є зваженою сумою втрат локалізації (localization loss, L_{loc}) та класифікації (confidence loss, L_{conf}):

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

Є два різновиди даної моделі: SSD300 і SSD512. SSD300 приймає на вхід зображення розміром 300×300. Має меншу роздільну здатність, мережа працює швидше. SSD300 досягає 74,3% mAP при 59 FPS (frames per second) на наборі VOC2007 [28].

На вхід мережі SSD512 подається зображення розміром 512×512, вища роздільна здатність. Ця мережа працює точніше, тоді як SSD500 досягає 76,9% mAP при 22 FPS, що перевершує Faster R-CNN (73,2% mAP при 7 FPS) і YOLOv1 (63,4% mAP при 45 FPS) [28].

2.4 RetinaNet

Ще одним одноступеневим детектором, який застосовується для виявлення об’єктів в режимі реального часу є RetinaNet [29]. Ця мережа набула широкого використання для аналізу аерофотознімків й супутникових зображень.

На думку авторів RetinaNet, точність одноступеневих детекторів нижча через те, що при їх навчанні не приділяється увага контрасту між об’єктами та фоном. Тому була представлена нова функція втрат – фокусна втрата (focal loss), яка при тренуванні мережі концентрує увагу на більш складних прикладах.

RetinaNet базується на ResNet, використовує Feature Pyramid Network (FPN) для витягу ознак, мережу для класифікації блоків якорів (із сигмоїдною функцією активації) і мережу для регресії від блоків якорів до блоків з реальними об’єктами (рис.2.8).

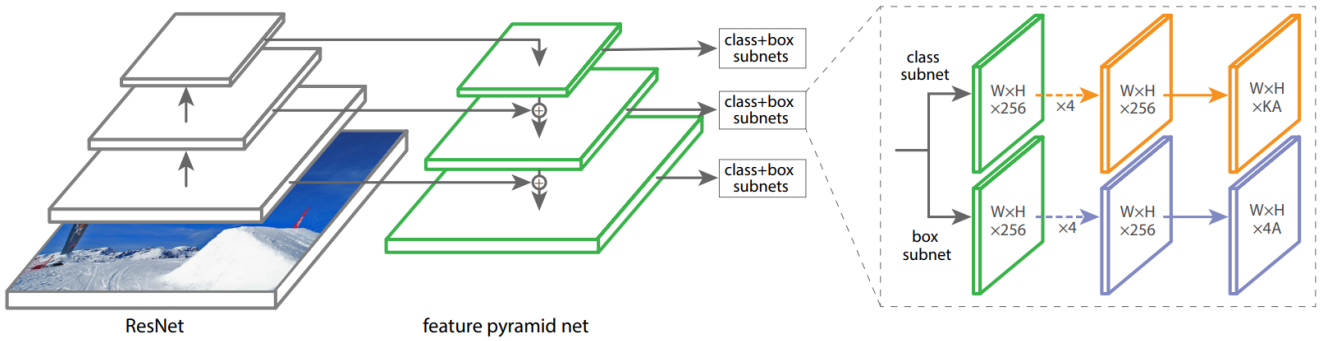


Рисунок 2.8 – Архітектура RetinaNet

Feature Pyramid Network (FPN) складається з трьох основних частин (рис.2.9): висхідний шлях (bottom-up pathway), низхідний шлях (top-down pathway) та бічні з’єднання (lateral connections).

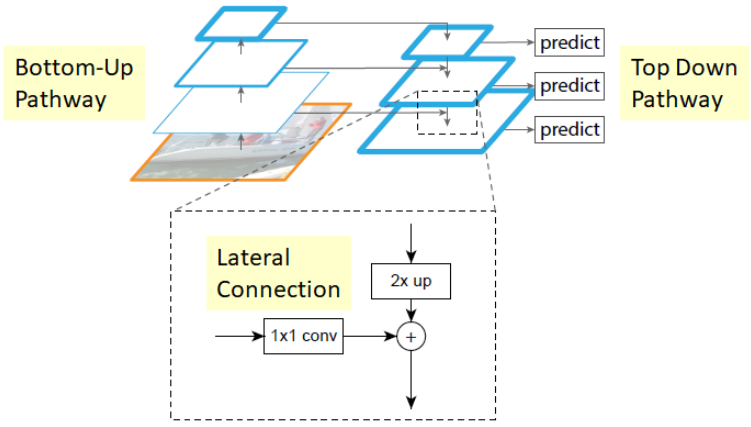


Рисунок 2.9 – Будова Feature Pyramid Network

Висхідний шлях є послідовністю згорткових шарів з поступово зменшувальною розмірністю. Верхні шари згорткової мережі мають більше семантичне значення, але меншу роздільну здатність, а нижні – навпаки. В оригінальній архітектурі RetinaNet цю функцію виконує базова підмережа ResNet. Низхідний шлях є послідовністю шарів, де карта ознак шару вище поступово збільшується до розмірів карти ознак нижчого шару. Завдяки бічним з'єднанням карти ознак відповідних шарів bottom-up і top-down пірамід поелементно складаються. Завдяки такому поєднанню згорткових шарів архітектура RetinaNet є мастабно-інваріантною.

Як уже зазначалося, в мережі запропоновано нову функцію втрат: фокусну втрату (focal loss) – це один із варіантів перехресної ентропії, що намагається вирішити проблему дисбалансу класів шляхом призначення більшої ваги важким і неправильно класифікованим прикладам (зашумленому фону або частково перекритим об'єктам) та меншої – простим прикладам (фоновим об'єктам) [29].

Реалізована додаванням коефіцієнта модуляції у перехресну ентропію:

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

де γ – параметр фокусування, який налаштовується за допомогою перехресної перевірки.

На зображенні нижче показано значення фокусної втрати залежно від різних значень γ. Зі збільшенням ймовірності правильно обраного класу коефіцієнт масштабування спадає до нуля.

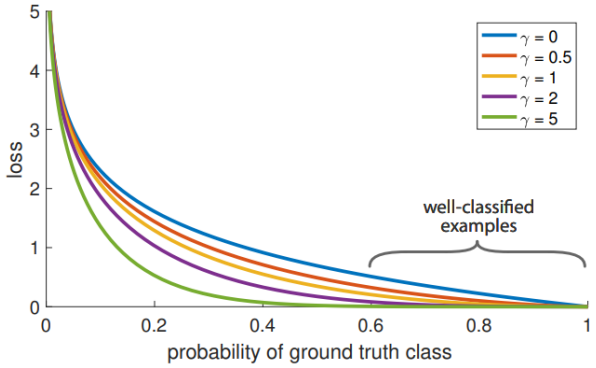


Рисунок 2.10 – Графік значень фокусної втрати

Характеристики фокусної втрати:

1. якщо приклад неправильно класифікований і значення p_t невелике, коефіцієнт модуляції близький до 1 і втрата не враховується;

2. параметр фокусування γ плавно регулює швидкість, з якою вага легких прикладів зменшується;

3. зі збільшенням значення γ збільшується і коефіцієнт модуляції (найоптимальніше значення γ, за результатами експериментів самих дослідників, – 2).

RetinaNet використовує α-збалансований варіант фокусної втрати (2.5), який використовує ваговий коефіцієнт α зі значенням параметрів α = 0,25, γ = 2.

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{2.5}$$

Впровадження фокусної втрати допомогло вирішити проблему дисбалансу класів.

2.5 Порівняння ефективності нейронних мереж

Для порівняння існуючих архітектур було обрано датасет Human-Aligned Bounding Boxes from Overhead Fisheye cameras (HABBOF) [30]. Розроблений в

лабораторії обробки візуальної інформації (Visual Information Processing, VIP) Бостонського університету та опублікований у вересні 2019 року. Набір даних містить 4 відеозаписи, записані камерами типу «риб'яче око» (fisheye cameras) у двох різних кімнатах (комп'ютерна лабораторія та невеликий конференц-зал) та відповідні анотації загальною кількістю 5837 кадрів. У кожному з цих відео 3 або 4 людини стоять, ходять, сидять, пишуть на дошці. У деяких відео вмикають і вимикають світло, пересувають меблі. Це підвищує реалістичність і рівень складності таких задач, як виявлення та стеження за людьми. Більш детальна інформація про кожне відео представлена в таблиці 2.1.

Таблиця 2.1 – Характеристики відео з датасету HABBOF [30]

Назва відео	Meeting 1	Meeting 2	Lab 1	Lab 2
Сценарій	Конференц-зал	Конференц-зал	Комп'ютерна лабораторія	Комп'ютерна лабораторія
Максимальна кількість людей	3	3	4	4
Кількість кадрів	1,119	1,121	1,792	1,805
Роздільна здатність	2048×2048×30 Гц	2048×2048×30 Гц	2048×2048×30 Гц	2048×2048×12 Гц
Тип камери	Axis M3057 PLVE	Axis M3057 PLVE	Geovision GV-FER12203	Geovision GV-FER12203
Виклики	Ходіння в центрі зображення та на периферії	Ходіння на периферії зображення	Сильні оклюзії, складні пози, ходіння по периферії зображення, нерівномірне освітлення	Близьке розташування людей, нерівномірне освітлення

Кожна обмежувальна рамка об'єкта описується у файлі з анотаціями наступними параметрами: [object_class, x, y, w, h, R], де object_class – клас об'єкта

(у даному випадку завжди «особа» (person)), x і y – координати центру обмежувальної рамки в пікселях від верхнього лівого кута зображення, w і h – ширина і висота рамки в пікселях, R – кут повороту рамки за годинниковою стрілкою від вертикальної осі у градусах (встановлюється в межах від -90 до +90 градусів).

У зв'язку з обмеженими обчислюваними ресурсами, для навчання мереж обрано підхід transfer learning (див. п. 1.3.3). Для тренування моделей з датасету обрано відео Meeting1, а для тестування – Meeting2. Завантажено попередньо натреновані моделі, налаштовано необхідні параметри. Усі моделі навчалися 30 епох.

Порівняння роботи мереж проводилося за наступними метриками:

1. середня точність (mAP): показник, який використовується для порівняння моделей і вимірювання точності детекторів; його складовими є метрики precision (точність) і recall (повнота) [31]; точність визначається відношенням істинно позитивних (TP) до загальної кількості прогнозованих позитивних результатів (суми істинно позитивних і хибно позитивних результатів (FP)): $precision = \frac{TP}{TP + FP}$; повнота є відношенням кількості істинно позитивних (TP) до суми істинно позитивних і хибно негативних (FN) результатів: $recall = \frac{TP}{TP + FN}$; показує, скільки обмежувальних рамок з усього набору було виявлено правильно.

2. кількість кадрів за секунду (FPS): кількість часу, що витрачається на обробку одного кадру [31]; ця метрика надається для оцінки швидкості моделей під час висновку (прогнозування) й може бути розрахована як відношення одиниці до часу висновку мережі (inference time); для роботи в режимі реального часу моделі бажано значення вище 24 FPS.

Спершу було обрано модель Faster R-CNN. Датасет було змінено під потреби моделі, а саме: анотації для зображень у файлах з форматом .txt перероблено у

xml-файли. З бібліотеки PyTorch (відкритої бібліотеки машинного навчання, яку використовують у задачах комп'ютерного зору та обробки природної мови) завантажено натреновану модель torchvision.models.detection.fasterrcnn_resnet50_fpn [32]. Навчання моделі тривало майже 3 години (160 хв). Значення функції втрат на останньому кроці навчання становило 19,7. Час висновку мережі – 0,116 секунд, що при роботі з відео означає 9 (1/0,116) кадрів за секунду. Приклад обробки відеокадру з тестувальної вибірки представлено на рисунку 2.11.

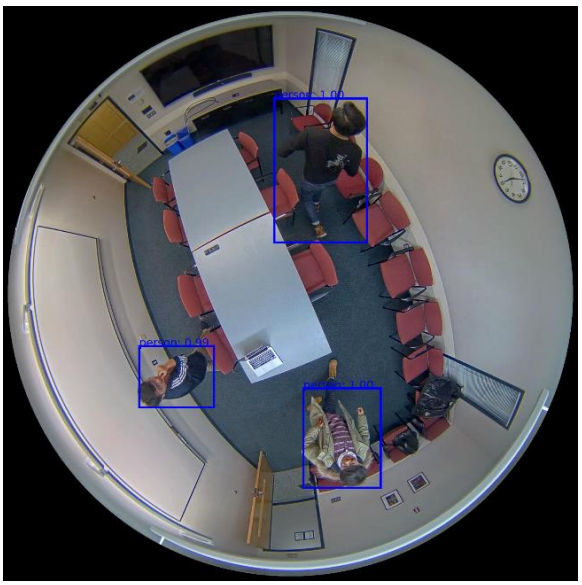


Рисунок 2.11 – Приклад роботи мережі Faster R-CNN

Отримано наступні показники: precision = 51,3%, recall = 31,7% при 9 кадрах за секунду. Це підтверджує високу точність двоступеневих детекторів і неможливість їхнього застосування для роботи в режимі реального часу через низьку швидкість обробки кадрів.

Далі проводилося тренування моделі YOLOv3, завантаженої з репозиторію [33]. Навчання тривало 47 хвилин. Час висновку становив 0,026 секунд.

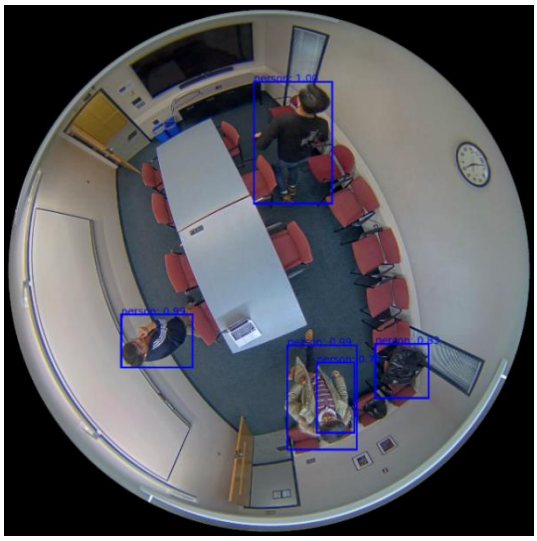


Рисунок 2.12 – Приклад роботи мережі YOLO

Отримано наступні показники: precision = 37,5%, recall = 26,4% при 39 кадрах за секунду. Це вказує на значну швидкість мережі з відносною точністю. Дана мережа підходить для виявлення об’єктів у відеопослідовностях завдяки високій швидкості обробки кадрів й достатній точності.

Попередньо натренована модель SSD була завантажена з репозиторію [34]. Для її тренування використовувалась бібліотеки TensorFlow і Keras (відкриті бібліотеки для реалізації машинного навчання Python). Файли анотацій з датасету були перероблені у формат .csv. Модель навчалася 2 години. Час висновку: 0,028 секунд.

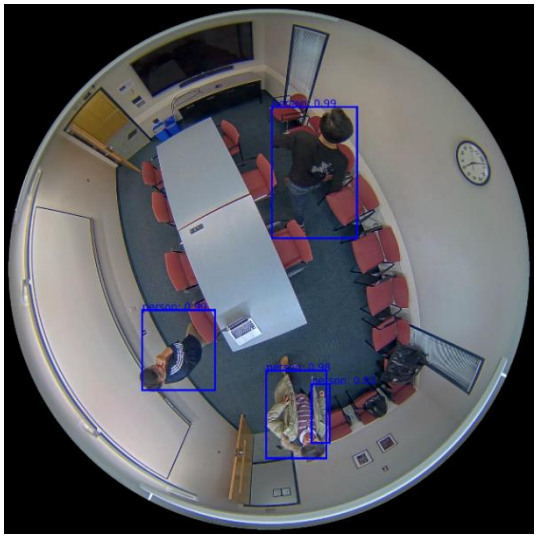


Рисунок 2.13 – Приклад роботи мережі SSD

Отримано наступні показники: precision = 39,7%, recall = 27,6% і 36 кадрів за секунду. Тобто вона є більш точною, але обробляє відеокадри повільніше у порівнянні з YOLO. Це досягається завдяки здатності мережі виявляти об’єкти у різних масштабах.

Насамкінець реалізовано RetinaNet за допомогою бібліотеки PyTorch: torchvision.models.detection.retinanet_resnet50_fpn [35]. Дана модель, як і Faster R-CNN вимагає анотації для зображень у форматі .xml. Навчання моделі тривало 2 год (120 хв). Час висновку становив 0,036 секунд. Приклад обробки зображення мережею представлено на рисунку 2.14.

Рисунок 2.14 – Приклад роботи мережі RetinaNet

Отримано наступні показники: precision = 41,2%, recall = 33,1% і 28 кадрів за секунду. Тож вона працює повільніше за решту одноступеневих детекторів (YOLO і SSD), проте вона є більш точною. А також має високе значення показника recall, що свідчить про її надійність.

Насамкінець за отриманим результатами складено узагальнену порівняльну таблицю (табл.2.2).

Таблиця 2.2 – Порівняння мереж на датасеті HABBOF

Мережа	Основа	mAP, %	recall, %	fps
Faster R-CNN	ResNet50	51,3	31,7	9
YOLOv3	DarkNet-53	37,5	26,4	39
SSD300	VGG16	39,7	27,6	36
RetinaNet	ResNet50	41,2	33,1	28

2.6 Висновок до розділу

З метою виявлення найкращої мережі для виявлення об’єктів у режимі реального часу було проведено порівняльний аналіз ефективності сучасних нейронних мереж, які використовують згорткові шари: двоступеневої Faster R-CNN та одноступеневих YOLO, SSD і RetinaNet. Різниця між дво- і одноступеневими детекторами полягає в тому, що перші шукають окремі регіони на зображенні для локалізації об’єкта, а другі – враховують повну інформацію про зображення.

Порівняння проводилося на датасеті HABBOF за метриками точності (accuracy), mean average precision (mAP), кількістю кадрів за секунду. Для навчання мереж було використано transfer learning.

Отримано наступні результати. Найшвидшою і найменш точною з усіх виявилася мережа YOLO з 39 кадрами за секунду і точністю 37,5%. На противагу, двоступенева мережа показала найвищу точність 51,3%. За показником recall на перше місце вийшла мережа RetinaNet: 33,1%.

В цілому, мережа Faster R-CNN досить точно виявляє об’єкти (51,3% на HABBOF), але її не варто застосовувати для роботи з відео (лише 9 кадрів за секунду).

YOLO працює значно швидше (39 кадрів за секунду) і підходить для роботи в режимі реального часу, але показує відносно невелику точність (37,5%). Виявляє більше помилок локалізації, ніж Faster R-CNN, та має труднощі з виявленням дрібних та об’єднаних у групи об’єктів.

SSD і RetinaNet можуть бути застосовані тоді, коли важливою є точність виявлення об’єктів. Хоча вони поступаються швидкістю YOLO. SSD при цьому виявляє об’єкти у різних масштабах, а RetinaNet володіє високим значенням повноти (33,1%), що вказує на правильність класифікації об’єктів.

Таким чином, оскільки для розв’язання поставленої в роботі задачі важливими є точність виявлення і можливість роботи в режимі реального часу, мережу YOLO обрано для подальшого удосконалення.

3 РОЗРОБЛЕННЯ Й ОЦІНКА ЕФЕКТИВНОСТІ ГІБРИДНОЇ НЕЙРОННОЇ МЕРЕЖІ

За результатами аналізу, наведеними у попередньому розділі, для удосконалення обрано мережу YOLO. З усіх можливих версій обрано третю, так як вона демонструє кращу збалансованість між точністю й швидкістю.

Було висунуто гіпотезу про те, що запровадження алгоритму HOG для витягу ознак має підвищити швидкість і точність детектора YOLO. Також вирішено додати окремий метод, який спиратиметься на координати центру обмежувальної рамки для забезпечення трекінгу об’єктів.

Створення гібридної мережі відбувається у кілька етапів:

а) вибір середовища програмування;

б) проектування програмного продукту;

в) підготовка датасету, розбиття на тестову і тренувальну вибірки;

г) розробка структури мережі;

д) навчання мережі, збереження у відповідні файли;

е) оцінка ефективності роботи мережі.

Надалі кожен з цих етапів розписано окремо.

3.1 Вибір середовища та інструментів розроблення

Вибір конкретного середовища і мови програмування залежить від поставленої задачі. Найпопулярнішою мовою програмування для роботи із нейронними мережами за даними [36] є Python (рис. 3.1). Вона містить набір спеціальних інструментів та алгоритмів машинного навчання, зокрема бібліотеки Scikit Learn і ChatterBot (призначену для обробки мовлення та навчання на наборах даних у форматі діалогів).

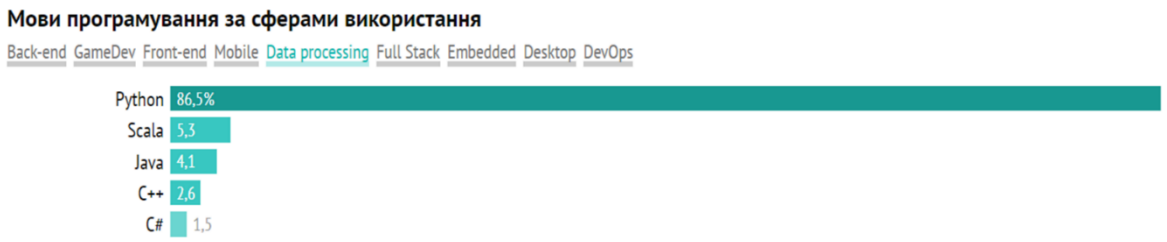


Рисунок 3.1 – Рейтинг мов програмування у сфері аналізу даних

Python підтримує структурне, об'єктно-орієнтоване, функціональне програмування. Основні риси – динамічна типізація, автоматичне управління пам'яттю, механізм обробки винятків, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі та об'єднання їх у пакети.

Scala – мова програмування, що поєднує властивості об'єктно-орієнтованого та функціонального програмування [36]. Програми мовою Scala виконуються на віртуальній машині Java, Scala сумісна з Java.

Іншою, не менш важливою мовою програмування є Java. Вона широко застосовується для бекенд-розроблення веб-додатків, додатків для смартфонів, але так само успішно може працювати і з нейронними мережами.

На Java і Scala навіть створено такий проект, як Smile (Statistical Machine Intelligence and Learning Engine) [37]. Це швидка комплексна система, призначена для реалізації машинного навчання, лінійної алгебри, інтерполяції та візуалізації, роботи з графами. Реалізує наступні алгоритми машинного навчання:

а) класифікація: дерева рішень, AdaBoost, логістична регресія, мережі RBF, наївний байєсівський класифікатор, аналіз Фішера;

б) регресія: опорна векторна регресія, дерева регресії, посилення градієнта, випадковий ліс, мережі OLS та LASSO, гребенева регресія;

в) кластеризація: BIRCH, DBSCAN, K-Means, G-Means, Neural Gas, карти самоорганізації, спектральна та ієрархічна кластеризації;

г) пошук найближчого сусіда: BK-Tree, KD-Tree, SimHash, LSH;

д) прихована модель Маркова, умовне випадкове поле.

Іншою мовою, яку широко використовують для машинного навчання є C++. Для роботи з нейромережами створено окремі бібліотеки TensorFlow. Популярність C++ обумовлена також розвитком розподіленої високопродуктивної платформи для градієнтного бустингу Microsoft LightGBM (що підвищує швидкість та ефективність навчання моделі) та бібліотеки Turi Create (яка спрощує створення моделей машинного навчання для розробників-початківців).

Найбільш зручною для створення нейронних мереж є мова програмування Python, яку і використано для розв'язання поставленої в роботі задачі. Порівняно з іншими ця мова програмування має більше бібліотек машинного навчання і, до того ж, стрімко розвивається.

В якості середовища розроблення мовою Python обрано Google Colaboratory, або просто Colab. Безкоштовний хмарний сервіс на основі Jupyter Notebook, що дозволяє писати та виконувати код безпосередньо у браузері. Google Colab надає безліч функцій, як і будь-яке інше середовище розроблення:

а) написання і запуск коду без локальної установки;

б) імпорт наборів даних із зовнішніх джерел та робіт з диску Google;

в) можливість зберігати роботи на диску Google;

Також Google Colab надає сервіси GPU і TPU, інтеграцію з PyTorch, Tensor Flow і Open CV, можливість імпорту або публікації проектів безпосередньо з GitHub. Проте сервіси GPU і TPU у безкоштовній версії Colab надаються на обмежений проміжок часу.

3.2 Проектування мережі

Уніфікована мова моделювання (UML) – графічна мова для специфікації, візуалізації, конструювання і документування програмних систем [38]. Вона є невід'ємною частиною процесу розробки програмного забезпечення, що відображає як концептуальні елементи системи (системні функції, бізнес процеси),

так і особливості їх реалізації (класи, схеми баз даних). Дозволяє поглянути на систему з різних точок зору. UML оперує такими основними поняттями, як: сутності, відношення та діаграма. Сутності – певні абстракції, які є базовими елементами моделей [38] (актори, класи, прецеденти, стани і т. д.). Відношення – відображення зв'язку між сутностями (відношення залежності, асоціації, узагальнення, реалізації). Діаграма – графічне відображення елементів системи у формі графа з вершинами (сутностями) і ребрами (відношеннями). Діаграми представляють систему в такому вигляді, у якому її можна легко перевести в програмний код. В UML використовуються 14 видів діаграм: діаграма класів, діаграма об'єктів, діаграма прецедентів, діаграма послідовностей, діаграма станів, діаграма діяльностей і інші.

Для проектування структури мережі застосовано діаграму класів та послідовності. Діаграма класів – статичне представлення структури моделі. Вона дає узагальнене представлення архітектури мережі через відповідні класи у програмі. Діаграму класів наведено у додатку А.

Діаграма діяльності описує послідовність переходу від однієї дії до іншої. Вона відображає динамічні аспекти програми і є різновидом діаграми станів (діаграми, що визначає зміну станів об'єкту). У даному випадку демонструє алгоритм роботи мережі. Діаграму діяльності подано у додатку Б.

3.3 Підготовка датасету

Камери кругового огляду (omni-directional cameras) широко використовуються для відеоспостереження. Їх різновидами є камери типу «риб'яче око», які мають кут огляду до 360˚. Вони застосовуються в аеропортах, роздрібних магазинах, готелях, лікарнях, офісних приміщеннях. Тому для тренування гібридної мережі використано датасет HABBOF, детально описаний у попередньому розділі.

Вхідні дані було розбито на тестову й тенувальну вибірки: відео Lab2, із загальною кількістю кадрів 1805 було застосовано для тренування мережі, а Meeting2 із 1121 кадрами – для валідації.

Перед подачею на навчання моделі дані були попередньо оброблені. Для цього було використано бібліотеку KerasCV – набір модулів (шарів, функцій втрат, доповнення даних), які можуть бути використані у побудові моделей для задач класифікації чи сегментації зображень, виявлення об’єктів, аугментації даних, попередньої обробки зображеннь тощо. Основна мета – надати узгоджений API для навчання сучасних моделей комп’ютерного зору.

Для попередньої обробки даних у KerasCV наявні такі методи, як RandomFlip(), RandAugment(), CutMix(), MixUp(). CutMix() і MixUp() дозволяють створювати міжкласові приклади. CutMix() випадковим чином вирізає частини одного зображення та розміщує їх поверх іншого, а MixUp() інтерполює значення пікселів між двома зображеннями. Обидва вони запобігають перенавчанню моделі та покращують здатність моделі до узагальнення. У даній роботі застосовано саме аугментацію і поворот зображень. Аугментація – процес розширення вхідних даних. Тобто створення додаткових даних для навчання моделі. RandomFlip() довільно перевертає зображення під час навчання горизонтально чи вертикально залежно від атрибута 'mode':

```
random_flip = keras_cv.layers.RandomFlip(mode="horizontal_and_vertical",
seed=None)
rand_augment = keras_cv.layers.RandAugment(
    value_range=(0, 255),
    augmentations_per_image=2,
    geometric=False,
)
def augment(inputs):
    inputs["images"] = rand_augment(inputs["images"])
    inputs = random_flip(inputs)
    return inputs

train_ds = train_dataset.map(augment, num_parallel_calls=tf.data.AUTOTUNE)
def dict_to_tuple(inputs):
    return inputs["images"], inputs["bounding_boxes"]
train_ds = train_ds.map(dict_to_tuple, num_parallel_calls=tf.data.AUTOTUNE)
```

```
val_ds = val_dataset.map(dict_to_tuple,
num_parallel_calls=tf.data.AUTOTUNE)
train_ds = train_ds.prefetch(tf.data.AUTOTUNE)
val_ds = val_ds.prefetch(tf.data.AUTOTUNE)
```

Конвеєр (pipeline) – це послідовність кроків під час перетворення даних. За допомогою .prefetch(tf.data.AUTOTUNE) створюється модель продуктивності вхідного конвеєра та запускається алгоритм оптимізації для рівномірного розподілу ресурсів ЦП під час навчання.

3.4 Опис структури мережі

Після підготовки датасету розробляється безпосередньо структура мережі. YOLOv3 як базова для вирішення підзадачі виявлення об’єктів. Оскільки побудова такої моделі «з нуля» вимагає багато обчислювальних ресурсів, було вирішено завантажити модель, попередньо натреновану на датасеті COCO [39]. Дескриптор HOG витягує ознаки з вхідного зображення і подає їх на вхід мережі YOLO для виявлення об’єктів. Далі за інформацією про місцеположення центральної точки об’єкта на кожному з кадрів реалізується відстеження цього об’єкта.

Структуру мережі подано на рисунку 3.2.

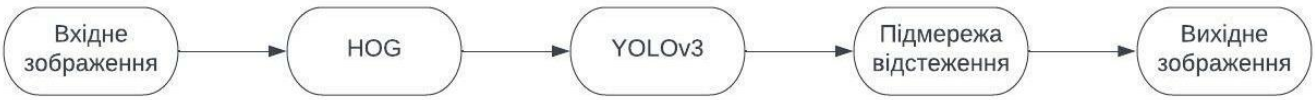


Рисунок 3.2. Структура гібридної мережі

Розглянемо детальніше алгоритм роботи мережі. На вхід мережі подається відео. За допомогою спеціального коду програма перетворює його у послідовність кадрів (масив зображень). Кожне зображення попередньо обробляється (попередній підпункт) і подається на вхід розробленої мережі. Відбувається виділення (витяг) ознак із кадру за допомогою алгоритму HOG. Зображення ділиться на невеликі області, у яких підраховуються напрямки і величина градієнтів. Отримані значення для кожної області збираються у гістограму градієнтів (яка є вектором із 9 чисел, що відповідають кутам від 0 до 160 градусів). Далі локальні гістограми з усіх

областей, на які ділиться зображення, зводяться у комбінований вектор ознак. Ці ознаки передаються на вхід попередньо натренованої моделі YOLO.

Уся модель формується класом:

```
class YOLOHOG(keras.Model):
    def __init__(self, num_classes, **kwargs):
        super(YOLOHOG, self).__init__(name="RetinaNet", **kwargs)
        self.fpn = HOGFeatExtract(backbone=None)
        self.num_classes = num_classes
        self.yolobase = YOLObackbone(num_classes)

    def call(self, image, training=False):
        features = self.fpn(image, training=training)
        N = tf.shape(image)[0]
        box_outputs = []
        for feature in features:
            box_outputs.append(tf.reshape(self.yolobase(feature),  [N,  -1,
4]))
        return tf.concat(box_outputs, axis=1)
```

У конструкторі цього класу (методі, що автоматично викликається під час створення об’єкта класу) створюються окремо обєкти усіх відповідних класів і мережа збирається в одне ціле. У основному коді створюється об’єкт класу YOLOHOG, який далі передається навчанню.

Насамкінець відбувається відстеження отриманих об’єктів. Для цього створено окремий алгоритм: у програмному коді створено окремий клас PntTracking, який за допомогою методу найближчих сусідів проводить двостороннє зіставлення дескрипторів точок (як у мережі SuperPoint для виявлення і відстеження точок відеокадрів [40]).

Так, при зчитуванні кадру відбувається виявлення об’єкта, розраховуються висота і ширина обмежувальних рамок, коєфіцієнт впевненості яких більший за 0,6. На основі отриманих даних про координати центра рамки викликається метод update(), який додає нові точки у трекер:

```
def update(self, pts, desc):
    if pts is None or desc is None:
      print('PointTracker: Warning, no points were added to tracker.')
      return
    assert pts.shape[1] == desc.shape[1]
    if self.last_desc is None:
```

```
      self.last_desc = np.zeros((desc.shape[0], 0))
    remove_size = self.all_pts[0].shape[1]
    self.all_pts.pop(0)
    self.all_pts.append(pts)
    self.tracks = np.delete(self.tracks, 2, axis=1)
    for i in range(2, self.tracks.shape[1]):
      self.tracks[:, i] -= remove_size
    self.tracks[:, 2:][self.tracks[:, 2:] < -1] = -1
    offsets = self.get_offsets()
    self.tracks = np.hstack((self.tracks, -1*np.ones((self.tracks.shape[0],
1))))
    matched = np.zeros((pts.shape[1])).astype(bool)
    matches = self.nn_match_two_way(self.last_desc, desc, self.nn_thresh)
    for match in matches.T:
      id1 = int(match[0]) + offsets[-2]
      id2 = int(match[1]) + offsets[-1]
      found = np.argwhere(self.tracks[:, -2] == id1)
      if found.shape[0] > 0:
        matched[int(match[1])] = True
        row = int(found)
        self.tracks[row, -1] = id2
        if self.tracks[row, 1] == self.max_score:
          self.tracks[row, 1] = match[2]
        else:
          track_len = (self.tracks[row, 2:] != -1).sum() - 1.
          frac = 1. / float(track_len)
          self.tracks[row, 1] = (1.-frac)*self.tracks[row, 1] + frac*match[2]
    new_ids = np.arange(pts.shape[1]) + offsets[-1]
    new_ids = new_ids[~matched]
    new_tracks = -1*np.ones((new_ids.shape[0], self.maxl + 2))
    new_tracks[:, -1] = new_ids
    new_num = new_ids.shape[0]
    new_trackids = self.track_count + np.arange(new_num)
    new_tracks[:, 0] = new_trackids
    new_tracks[:, 1] = self.max_score*np.ones(new_ids.shape[0])
    self.tracks = np.vstack((self.tracks, new_tracks))
    self.track_count += new_num
    keep_rows = np.any(self.tracks[:, 2:] >= 0, axis=1)
    self.tracks = self.tracks[keep_rows, :]
    self.last_desc = desc.copy()
    return
```

Тут спершу видаляються з трекера, проте зберігаються в пам’яті, старі значення, оновлюються зміщенння треків, а далі додаються нові значення на основі методу двостороннього зіставлення найближчого сусіда.

Виконується двостороннє зіставлення найближчих сусідів двох наборів дескрипторів точок. Двостороннє у даному випадку означає, що зіставлення дескриптора A→B дорівнює зіставленню з B→A.

Метод найближчого сусіда – один з найпростіших методів класифікації [41]. Процес навчання полягає у запам’ятовуванні всіх об’єктів навчальної вибірки. Об’єкт X_0 відносять до того класу, представник якого найближче розташований до нього тобто:

$$X_0 \in S_l : X_q \in S_l, d(X_0, X_q) = \min_{i=1,N} d(X_0, X_i) \tag{3.1}$$

Перевагами методу є простота реалізації та наочна інтерпретація результатів.

Недоліками:

а) метод потребує зберігання усієї навчальної вибірки, що призводить до значних витрат пам’яті.

б) нестійкість до шуму; якщо, наприклад, серед навчальних об’єктів є викид, тобто об’єкт, розташований серед представників чужого класу, то всі об’єкти, для яких він виявиться найближчим, будуть класифіковані неправильно.

в) пошук найближчого сусіда передбачає порівняння класифікованого об’єкта з усіма об’єктами навчальної вибірки, тобто потребує багато часу для задач із великими навчальними вибірками.

У даній роботі використано саме цей метод, хоча на практиці більш поширеним є його модифікація: метод k найближчих сусідів, який стійкіший до шуму. Його відмінність від методу найближчого сусіда полягає в тому, що на етапі класифікації знаходять не один, а k об’єктів, ближчих до X_0 в обраній метриці. Об’єкт X_0 відносять до того класу, представників якого виявилося більше серед k сусідів.

Після додавання нових точок таким чином відбувається побудова треків. Для цього присутній метод get_tracks(), який видає матрицю, що зберігає індекси треків:

```
def get_tracks(self, min_length):
    if min_length < 1:
      raise ValueError('\'min_length\' too small.')
    valid = np.ones((self.tracks.shape[0])).astype(bool)
    good_len = np.sum(self.tracks[:, 2:] != -1, axis=1) >= min_length
    not_headless = (self.tracks[:, -1] != -1)
    keepers = np.logical_and.reduce((valid, good_len, not_headless))
```

```
returned_tracks = self.tracks[keepers, :].copy()
return returned_tracks
```

При цьому треки, які не спостерігаються в останніх кадрах, видаляються. Насамкінець візуалізуються виявлені об’єкти і їх треки на вхідному кадрі.

3.5 Тренування мережі

В якості оптимізатора навчання підмережі виявлення об’єктів обрано стохастичний градієнтний спуск з моментом 0,9. Стохастичний градієнтний спуск – ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення (випадкового наближення). У самому методі градієнтного спуску для знаходження локального мінімуму функції здійснюються кроки, пропорційні протилежному значенню градієнту функції в поточній точці. У якості функції помилок обрано квадратичну похибку між прогнозованими і справжніми значеннями (див. п. 2.2). Оптимальну кількість навчальних епох встановлено ексериментальним шляхом. Спочатку обрано 30 епох, як і для детекторів з попереднього розділу. Мережа досягла точності 38,9 % при 41 кадрі за секунду. У порівнянні з показниками, отриманими для звичайної моделі YOLO (див. п. 2.5), поєднання YOLO з дескриптором HOG для витягу ознак на даному наборі даних підвищило точність і швидкість моделі. Але навчання зайняло на понад 11 хвилин більше часу, що ймовірно зумовлено комбінованою архітектурою мережі.

Судячи з отриманого графіку зміни точності (рис. 3.3), має місце проблема недонавчання мережі.

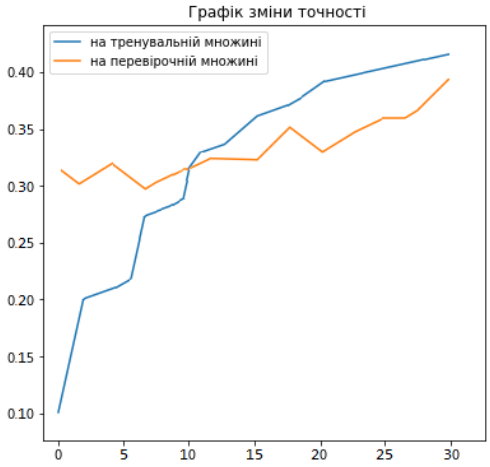


Рисунок 3.3. Графік зміни точності для 30 епох навчання

Тож, було вирішено донавчити мережу, застосувавши більшу кількість епох навчання. При навчанні 100 епох мережа досягла точності 39,2 % при 43 кадрах за секунду. Аналізуючи графіки навчання (рис. 3.4) можна стверджувати, що проблему недонавчання мережі усунуто.

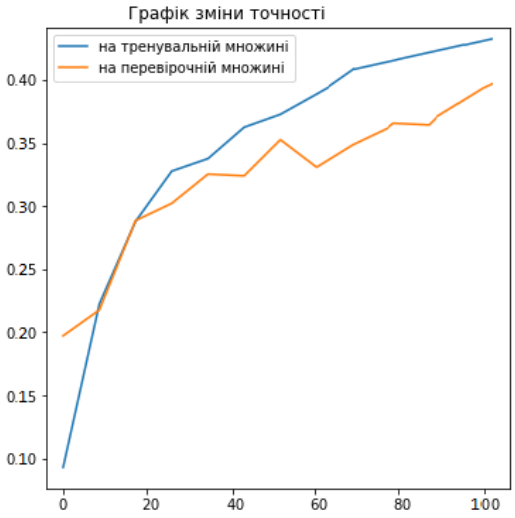


Рисунок 3.4. Графік зміни точності й функції втрат для 100 епох навчання

Таким чином було встановлено, що використання дескриптора HOG разом з YOLO для набору даних HABBOF дало виграш у точності на 1,7% і на чотири кадри за секунду швидшу роботу (табл.3.1). Проте навчання зайняло на 50 хвилин більше часу.

Таблиця 3.1 – Порівняння детектора YOLOv3 і YOLOv3 з HOG для витягу ознак на датасеті HABBOF

Мережа	mAP, %	recall, %	fps
YOLOv3	37,5	26,4	39
YOLOv3+HOG	39,2	27,1	43

3.6 Тестування продуктивнсті

Готова нейронна мережа може обробляти вхідні дані з:

а) відеофайлу форматі .mp4;

б) веб-камери USB.

Емпіричним шляхом проаналізовано результати роботи мережі в різних умовах: перекриття осіб у відео, погано освітлені кадри.

Спершу на вхід мережі подано відео (послідовність кадрів) з камери спостереження конференц-залу.

Програма вивела відеопослідовність з розпізнаними об’єктами. Визначені об’єкти позначаються прямокутниками з назвою відповідного класу та точністю розпізнавання. Протягом усієї послідовності програма надає можливість бачити зміну положення виявлених об’єктів.

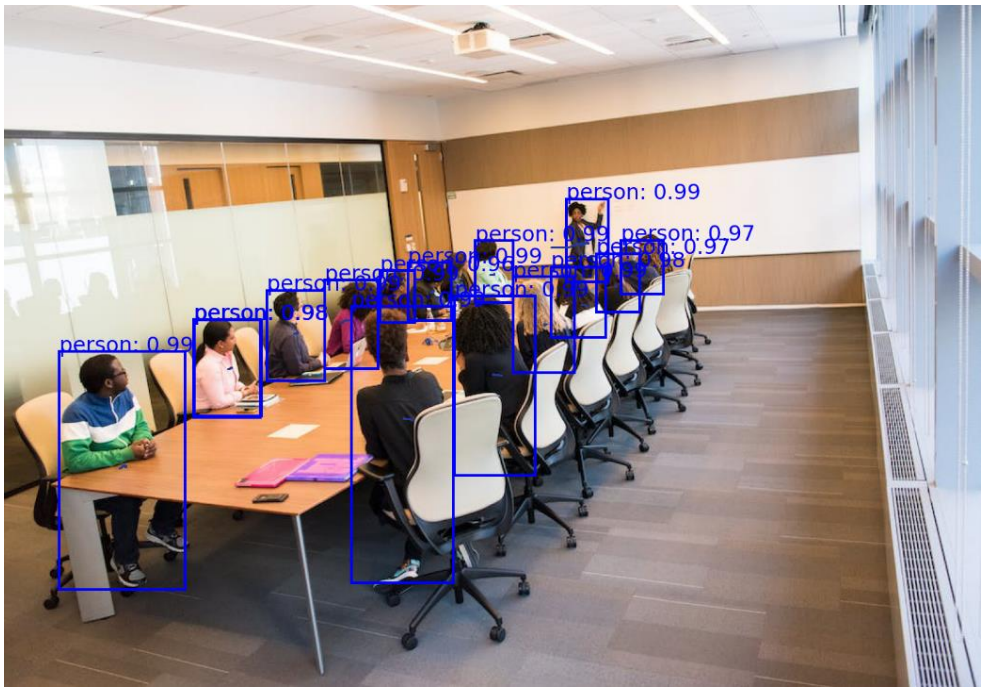


Рисунок 3.5. Результат роботи мережі

Результати роботи програми повністю співпадають з вимогами: знайдені об’єкти коректно виявлені й відстежені.

Другий експеримент полягав у перевірці стійкості мережі до спотворень вхідних даних.

На вхід мережі подано відео з пішоходами на перехресті, зняте у вечірній час. Подалі від камери обєкти відстеження приймали дуже нечіткі контури, майже зливалися із заднім фоном.

Програма знову ж таки вивела відеопослідовність з розпізнаними об’єктами, відстежила зміну їх положення. Кадр результуючого відео представлено на рисунку 3.6.

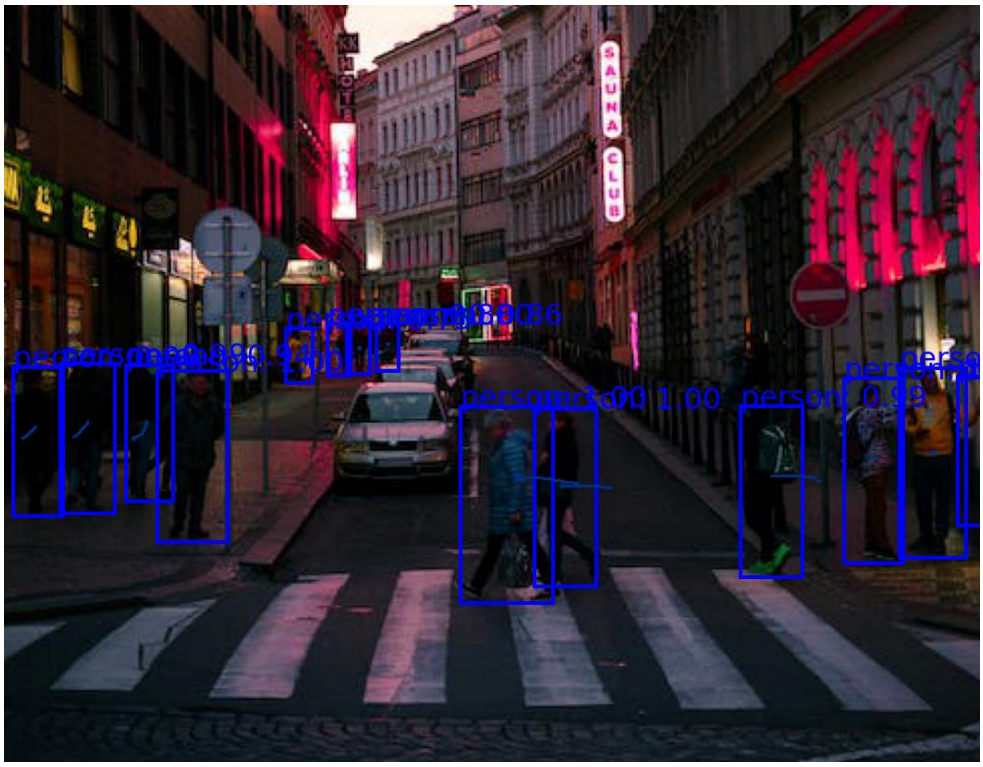


Рисунок 3.6. Результати роботи мережі в умовах поганого освітлення

Як і було очікувано, мережа впоралася із поставленою задачею. Достатньо точно розпізнала і змогла відстежити об’єкти, що знаходилися подалі від камери. Також можна спостерігати, що із наближенням об’єктів точність їх розпізнавання підвищується.

Надалі на вхід мережі було подано відео з камер спостереження магазину. У ньому наявна велика кількість дрібних об’єктів, а частина людей перекривається.

Мережа достатньо точно виявила і відстежила осіб, що знаходилися найближче до об’єктиву. Приклад кадру результату мережі представлено на рисунку 3.7.

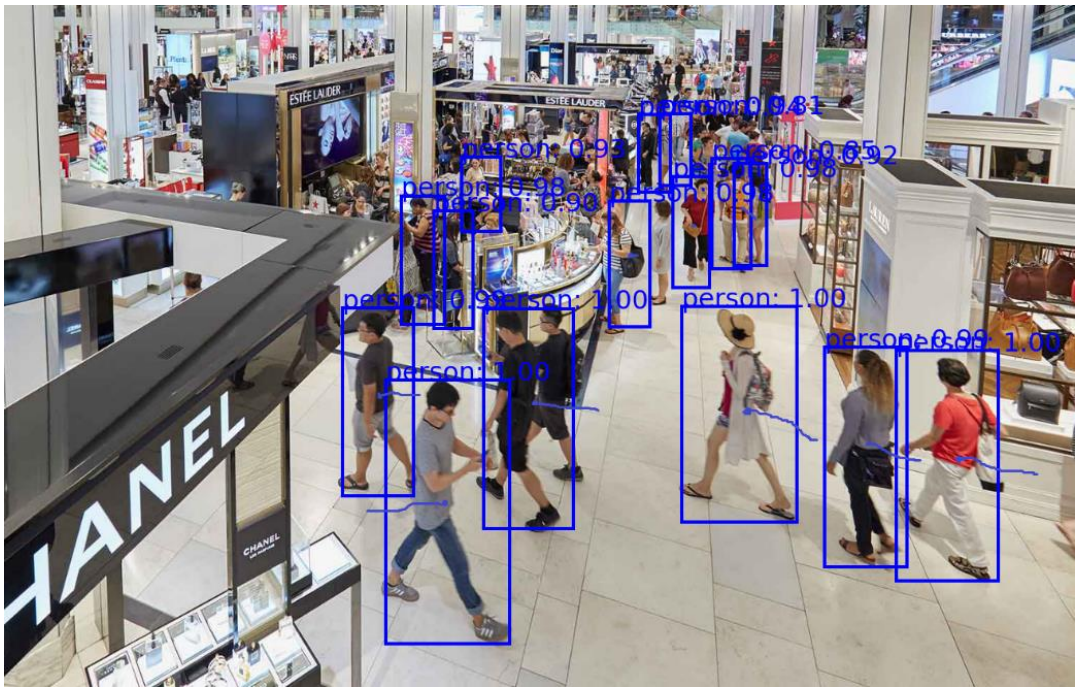


Рисунок 3.7. Результати роботи мережі з дрібними об’єктами

Мережа впоралася відносно добре навіть при перекритті об’єктів. Проте очікування не справдилися і проблема виявлення дрібних об’єктів все ж залишається не вирішеною до кінця.

Насамкінець на вхід мережі подано відео з веб-камери. Зйомка відбувалася в торгівельному центрі. Присутні люди досить точно були виявлені й відстежені.

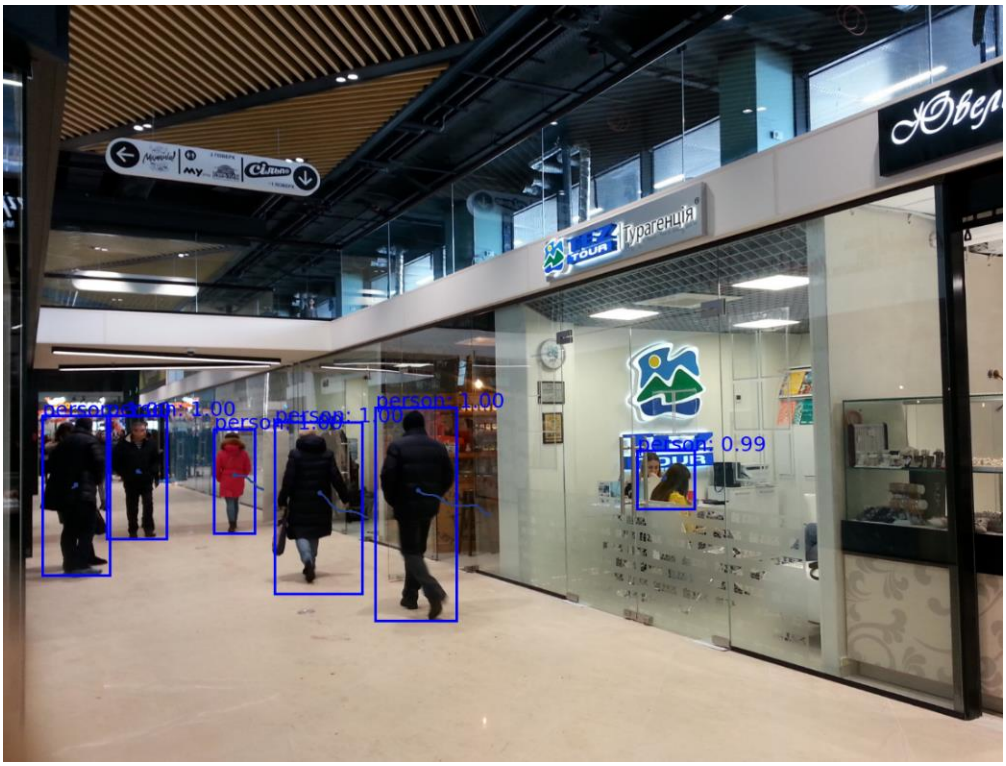


Рисунок 3.8. Результати роботи з відео з веб-камери

В цілому результат даного прикладу повністю задовольняє очікування. Мережа виявляє непогані результати як при роботі із завантаженим, так і з потоковим відео.

Варто також зазначити, що мережа швидше працює (обробляється більша кількість кадрів за секунду) при роботі із завантаженим відео.

3.7 Висновок до розділу

Таким чином, було розроблено нейронну мережу для покращення виявлення і відстеження людини у відеопотоці.

Для цього застосовано алгоритм HOG для витягу ознак з відеокадра. Отримана інформація подається на вхід попередньо натренованої YOLO. Додатково створено алгоритм, який на основі виявлених обмежувальних рамок відстежує об’єкти.

Гіпотеза справдилась – застосування алгоритму HOG для витягу ознак перед подачею на вхід мережі YOLO призвело до покращення ефективності виявлення об’єктів.

Створена мережа досягла точності 39,2 % при 43 кадрах за секунду. При цьому вона проявила гарні результати і при роботі в умовах поганого освітлення. Проте відкритим залишається питання виявлення дрібних об’єктів.

4 СТВОРЕННЯ ТА ЕКОНОМІЧНА ОЦІНКА СТАРТАП-ПРОЕКТУ

Наразі існує велика потреба в якісних системах обробки інформації, зібраної з камер відеоспостереження. Останнім часом для виявлення і відстеження людини як на зображеннях, так і у відео широкого поширення набули згорткові нейронні мережі. Але при роботі з ними можуть виникнути проблеми з виявленням дрібних чи неповних об’єктів, стійкістю до зашумлених чи погано освітлених зображень. Деякі мережі обробляють відео зі швидкістю до 10 кадрів за секунду що не прийнятно для застосування у режимі реального часу. Тому проблема пошуку ефективної мережі залишається актуальною. Запропонована гібридна мережа демонструє швидкість і точність, вищу, ніж існуючі мережі, тож стартап може стати успішним на ринку.

Розроблення стартап-проекту включає наступні етапи:

а) маркетинговий аналіз стартап-проекту:

розроблення опису ідеї проекту, визначення загальних напрямків використання потенційного товару чи послуги, а також їх відмінностей від конкурентів; аналіз ринкових можливостей щодо реалізації проекту та створення стратегії ринкового впровадження потенційного товару в його межах;

б) організація стартап-проекту:

складання календарного плану, графіку реалізації проекту; розрахунок потреби в основних засобах та нематеріальних активах; визначення планового обсягу виробництва потенційного товару; розрахунок загальних початкових витрат на запуск проекту та планових загальногосподарських витрат, необхідних для реалізації проекту;

в) фінансово-економічний аналіз та оцінка ризиків проекту:

визначення обсягу інвестиційних витрат; розрахунок основних фінансово-економічних показників проекту (обсягу виробництва продукції, собівартості виробництва, ціни реалізації, податкового навантаження та чистого прибутку) та

визначення показників інвестиційної привабливості проекту (запасу фінансової міцності, рентабельності продажів та інвестицій, періоду окупності проекту); визначення рівня ризикованості проекту, основних ризиків проекту та шляхів їх запобігання;

г) заходи з комерціалізації проекту:

визначення цільової групи інвесторів та опису їх ділових інтересів; складання інвест-пропозиції (оферти, стислої характеристики проекту для попереднього ознайомлення інвестора із проектом); планування заходів з просування оферти (визначення комунікаційних каналів та планування системи заходів з просування в межах обраних каналів); планування ресурсів для реалізації заходів з просування оферти.

У даній роботі детально розписано саме маркетинговий аналіз стартап-проекту.

4.1 Опис ідеї проекту

Зміст ідеї проекту, а також можливі напрямки його застосування і вигоди для користувача подано в таблиці 4.1.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Автоматизоване швидке і ефективне виявлення та відстеження людей у відеокадрах	1. Системи автоматичного керування для виявлення пішоходів на дорогах	Швидке виявлення пішоходів на дорогах (навіть в умовах слабкого освітлення)
	2. Зупинки громадського транспорту і аеропорти	Якісний аналіз пасажиропотоку
	3. Роздрібна торгівля	Точний підрахунок кількості та аналіз поведінки клієнтів магазинів
	4. Офісні приміщення і системи розумного будинку	Відстеження кількості та поведінки присутніх осіб

В результаті аналізу потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів визначено (табл. 4.2):

а) перелік техніко-економічних властивостей та характеристик ідеї;

б) попереднє коло товарів-замінників чи товарів-аналогів, що вже існують на ринку;

в) показники, що мають гірші (W, слабкі), аналогічні (N, нейтральні), чи кращі (S, сильні) значення для власної ідеї.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Потенційні концепції конкурентів				W	N	S
		Мій проект	YOLO	SSD	RetinaNet			
1	Середня точність виявлення (mAP), %	38,9	37,5	39,7	41,2		+	
2	Швидкість виявлення (fps)	41	39	36	28			+
3	Робота в умовах поганого освітлення	Добре	Погано	Відносно добре	Відносно добре			+
4	Робота з дрібними об’єктами	Погано	Погано	Добре	Відносно добре	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

Проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Для визначення технологічної здійсненності ідеї проекту було проаналізовано наступні складові (табл. 4.3): технологія, за якою буде виготовлено товар згідно ідеї проекту; наявність чи відсутність такої технології; доступність технології.

Таблиця 4.3 – Технологічна здійсненість ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення нейронної мережі	Мова програмування Python	Наявні	Доступні
2		Мова програмування C#	Наявні, необхідні допрацювання	Доступні
Обрана технологія реалізації ідеї проекту: мова програмування Python				
3	Мережа гібридна	Створення власної мережі	Потрібно поєднувати власні створені мережі	Навчання глибокої мережі може зайняти багато часу
4		Модифікація існуючої мережі	Необхідні допрацювання існуючих мереж	Доступна
Обрана технологія реалізації ідеї проекту: модифікація існуючої мережі				
5	Мережа приймає на вхід послідовність відеокадрів і виявляє в них людей	Бібіотека Keras	Наявна	Доступна
6		Бібліотека OpenCV	Наявна	Доступна
7		Бібліотека TensorFlow	Наявні деякі з функіцй	Доступна
8		Бібліотека PyTorch	Наявні деякі з функіцй	Доступна
Обрана технологія реалізації ідеї проекту: бібліотеки Keras, OpenCV				

За результатами аналізу таблиці технологічним шляхом реалізації проекту обрано такі технології, як мова програмування Python, бібліотеки OpenCV і Keras

через їх доступність. Було прийнято рішення модифікувати готову нейронну мережу.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначено ринкові можливості, які можна використати під час впровадження проекту, та ринкові загрози, що можуть перешкодити реалізації проекту. Це дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій конкурентів.

За результатами проведеного аналізу попиту складено таблицю 4.4.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/рік	2700000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Необхідні потужні обчислювальні ресурси для запуску і, за потреби, перенавчання мережі
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	35

Порівняно з банківським відсотком на вкладення середня норма рентабельності в галузі є меншою, ринок зростає, а вимоги до стандартизації відсутні. Тому, ринок є привабливим для входження.

Надалі з’ясовано потенційні групи клієнтів, їх характеристики, та сформовано орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Точне виявлення і відстеження пішоходів на дорогах	Компанії, що виробляють системи автоматичного керування	Необхідно швидко виявляти точне місцеположення пішоходів, збільшення попиту на продукцію	Швидка робота системи
2	Підрахунок кількості й відстеження поведінки осіб у громадських місцях	Власники магазинів, адміністрація систем відеоспостереження	Потреба точної роботи в умовах великої щільності осіб	Точність виявлення
3	Покращення систем відеоспостереження	Розробники підсистем відеоспостереження для розумних будинків	Зацікавленні у точності роботи в умовах низької освітленості, зашумленості відео	Можливість роботи при різних спотвореннях вхідних даних

За результатами аналізу ринкового середовища складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 4.6-4.7).

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок продуктів з кращими показниками швикості, точності	Вдосконалення продукту, додавання нових функцій
2	Зміна потреб користувачів	У користувачів може з’явитися потреба виявляє і відстежувати конкретних осіб у відео	Перенавчання мережі для досягнення бажаних результатів

Таблиця 4.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Точність та швидкість виявлення і відстеження	Мережа забезпечує високу точність навіть в умовах низької освітленості	Залучення більшої кількості користувачів
2	Розширення	Мережу можна адаптувати і для підрахунку кількості виявлених осіб	Додавання відповідного функціоналу

Визначено загальні риси конкуренції на ринку. Результати аналізу пропозиції представлено у таблиці 4.8, а умов конкуренції в галузі (за моделлю 5 сил М. Портера) – таблиці 4.9.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія	Представлено багато подібних проектів	Розробка унікального продукту, проведення рекламних кампаній
2. За рівнем конкурентної боротьби: міжнародний	Подібні проекти можуть бути доступними по всьому світу	Розширення цільової аудиторії, орієнтація на потреби корисувача
3. За галузевою ознакою: міжгалузева	Використання продукту в різних галузях	Захоплення компанією більше галузей
4. Конкуренція за видами товарів: товарно-родова	Конкуренція різних детекторів і трекерів об’єктів	Удосконалення існуючих функцій системи
5 За характером конку-рентних переваг: нецінова	Конкуренція між функціями товару	Залучення користувачів точністю і швидкістю роботи
6 За інтенсивністю: не марочна	Поняття марки на ринку відсутнє	Немає потреби розкручувати бренд

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу:	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Мережа YOLO+Deep Sort	Наявні обчислювальні ресурси	Відсутні	Контроль якості, рівень попиту	Відсутні
Висновки:	Інтенсивність конкурентної боротьби	Для створення подібних продуктів потрібні потужні обчислювальні ресурси	Постачальники відсутні	Клієнти визначають рівень попиту, спираються на якість продукту	Товари-замінники відсутні

Тож, з огляду на конкурентну ситуацію на ринку (табл. 4.9), а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6–4.7), можна визначити перелік основних факторів конкурентоспроможності проекту.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Точність та швидкість	Швидке і точне виявлення і відстеження об’єктів
2	Якість	Працює в умовах поганого освітлення
3	Собівартість	Для доступу потрібне Інтернет підключення та веб-камера
4	Доступність	Програмний продукт з відкритим вихідним кодом

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Точність	16					+		
2	Швидкість	17			+				
3	Якість	20				+			
4	Собівартість	18		+					
5	Доступність	20		+					

На основі виділених сильних і слабких сторін проекту (табл. 4.11) складено матрицю SWOT-аналізу (сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities)) (табл. 4.12). Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, але, на відміну від них, не є реалізованими на ринку (мають певну ймовірність здійснення).

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Точність Швидкість Низька собівартість	Слакбі сторони: Робота з дрібними об’єктами Не сформована база клієнтів
Можливості: Робота при поганому освітленні Розширення функціоналу	Загрози: Недовіра користувачів Поява нових конкурентів

На основі SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (табл. 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок як потенційна модель	70%	12 місяців
2	Фізична релізація за допомогою відеокамери	45%	15 місяців

Отримання ресурсів для першої альтернативи є більш простим та ймовірним. До того ж ця альтернатива має найбільш стислі строки реалізації. Її обрано для виведення стартап-проекту на ринок.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії починається з визначення стратегії захоплення ринку. Опис цільових груп потенційних споживачів наведено в таблиці 4.14.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Виробники систем автоматичного керування	Висока	72%	Висока	Середня
2	Адміністратори систем відеоспостереження	Висока	63%	Середня	Середня
3	Персональні користувачі	Низька	25%	Низька	Висока

За результатами аналізу потенційних груп споживачів (сегментів) обрано цільові групи один і два (виробники систем автоматичного керування та

адміністратори систем відеоспостереження), для яких і буде запропоновано товар. Стратегія охоплення ринку в цьому випадку – диференційований маркетинг.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (табл. 4.15) та стратегію конкурентної поведінки (табл. 4.16).

Таблиця 4.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Вихід на ринок як потенційна модель	Диференційований маркетинг	Швидкість системи і можливість роботи при поганому освітленні	Стратегія диференціації

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Шукати нових і забирати існуючих	Схожа архітектура мережі використо-вуватиметься як базова	Стратегія наслідування лідера

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та продукту (табл. 4.5), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (табл. 4.16) розроблено стратегію позиціонування (табл. 4.17). Вона полягає у формуванні ринкової позиції (комплексу асоціацій), за якою споживачі мають ідентифікувати торгівельну марку.

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції проекту	Вибір асоціацій, які мають сформувати комплексну позицію проекту
Швидкість, точність, зручність використання	Стратегія диференціації	Точність, швидкість, якість, собівартість, доступність	Швидкодія, легкість встановлення й стійкість до зашумлень

Таким чином, цільовою групою споживачів продукту є виробники систем автоматичного керування та адміністратори систем відеоспостереження. Стратегію диференціації обрано як базову стратегію розвитку, а стратегію наслідування лідера – як базову стратегію конкурентної поведінки.

4.5 Створення маркетингової програми стартап-проекту

Першим кроком створення маркетингової програми є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 4.18 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкість	Робота в режимі реального часу	Спрощений витяг ознак для підмережі виявлення
2	Точність	Висока точність виявлення і відстеження людей	
3	Якість	Можливість роботи в умовах поганого освітлення	Робота зі спотвореними відними даними
4	Простота у використанні	Зручність завантаження і використання	Приєднання програмного модулю до камери чи подання записаного відео на вхід мережі

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту, його фізичні складові, особливості процесу його надання (табл. 4.19).

Таблиця 4.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>Товар за задумом</td><td colspan="3">Швидке та точне розпізнавання і відстеження людей у відеопотоці в реальному часі з використанням згорткових нейронних мереж</td></tr>
<tr><td rowspan="6">Товар у реальному виконанні</td><td>Властивості</td><td>М/Нм*</td><td>Вр/Тх/Тл*</td></tr>
<tr><td>зручність користування під час виконання основних та допоміжних операцій</td><td>М</td><td>Тл</td></tr>
<tr><td>точність</td><td>М</td><td>Тх</td></tr>
<tr><td>швидкість</td><td>М</td><td>Тх</td></tr>
<tr><td>доступність</td><td>М</td><td>Вр</td></tr>
<tr><td colspan="3">Якість: відповідає нормам ISO/IEC 25010:2014 «Інженерія систем та програмного забезпечення – Вимоги до якості систем та програмного забезпечення»</td></tr>
<tr><td></td><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td rowspan="2">Товар із підкріпленням</td><td colspan="3">До продажу: нейронна мережа для завантаження</td></tr>
<tr><td colspan="3">Після продажу: можливість розширення функціоналу під власні потреби</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: потенційний продукт буде захищено від копіювання за допомогою використання патентів</td></tr>
</table>

*М/Нм – монотонні чи немонотонні; Вр/Тх/Тл – вартісні, технічні, технологічні.

Наступним кроком є визначення оптимальної системи збуту (табл. 4.20), в межах якої приймається рішення щодо:

а) проведення збуту власними силами чи залучення сторонніх посередників (власна або залучена система збуту);

б) вибору та обґрунтування оптимальної глибини каналу збуту;

в) вибір та обґрунтування виду посередників.

Таблиця 4.20 – Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Безкоштовний доступ до програми	Встановлення контактів із споживачами маркетингової інформації	Один рівень (від виробника до споживача)	Прямий (через Інтернет)
2		Формування попиту і стимулювання збуту		
3		Дослідницька робота зі збору даних про зацікавленість споживачів у продукції		

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.21).

Таблиця 4.21 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Використання у системах відеоспостере-ження	Інтернет, соціальні мережі	Швидкість, точність роботи, можливість виявлення в умовах поганого освітлення	Доведення переваг продукту над конкурентами	–

Таким чином розроблено ринкову (маркетингову) програму, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів,

конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

4.6 Висновок до розділу

Отже, в межах даного розділу був проведений маркетинговий аналіз розробленої гібридної нейронної мережі як стартап-проекту.

Виявлено сильні та слабкі сторони проекту, проведено SWOT аналіз, аналіз конкурентів та цільової аудиторії. На основі отриманих результатів сформовано стратегію виходу на ринок і маркетингову стратегію для обраної цільової аудиторії.

Бар’єрами для входження є наявність потужних обчислювальних ресурсів. Цільовими групами клієнтів є виробники систем автоматичного керування та адміністратори систем відеоспостереження. Базова стратегія розвитку – стратегія диференціації.

Попит на продукцію даного типу наявний і зростає. Рентабельність роботи та рівень конкуренції на ринку високі. Основним каналом збуту є Інтернет.

З огляду на визначені потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проекту, ринок є привабливим для входження. Подальша імплементація проекту є доцільною. Для ринкової реалізації проекту доцільно обрати альтернативу входження на ринок як потенційна модель.

ВИСНОВОК

У даній роботі було досліджено використання згорткових нейронних мереж у задачі виявлення та відстеження людини в режимі реального часу.

Отримано наступні результати. Проведено огляд існуючих методів виявлення об’єктів на зображеннях, зокрема: алгоритму Віоли-Джонса, гістограми орієнтованих градієнтів. Обґрунтовано вибір згорткових нейронних мереж для вирішення підзадачі виявлення об’єктів.

Розглянуто принципи роботи, переваги та недоліки мереж Faster R-CNN, YOLO, SSD та RetinaNet. Проведено їх порівняльний аналіз за показниками швидкості й точності розпізнавання на датасеті HABBOF. Визначено, що Faster R-CNN досить точно виявляє об’єкти (51,3%), але її не варто застосовувати для роботи з відео (лише 9 кадрів за секунду). YOLO працює значно швидше (39 кадрів за секунду) і підходить для роботи в режимі реального часу, але показує відносно невелику точність (37,5%). SSD і RetinaNet можуть бути застосовані тоді, коли важливою є точність виявлення об’єктів. SSD при цьому виявляє об’єкти у різних масштабах, а RetinaNet володіє високим значенням повноти (33,1%), що вказує на правильність класифікації об’єктів.

Розроблено структуру власної мережі для виявлення і відстеження людини у відеопотоці. Обрано і підготовано набір даних для навчання. Створена мережа досягла точності 39,2 % при 43 кадрах за секунду. Мету досягнуто: застосування алгоритму HOG для витягу ознак перед подачею на вхід мережі YOLO призвело до підвищення швидкості й точності виявлення об’єктів.

Зі створеною мережею проведено експерименти з метою оцінки роботи мережі в різних умовах. Встановлено, що вона добре працює в умовах поганого освітлення, проте відкритим залишається питання виявлення дрібних об’єктів.

Також було проведено маркетинговий аналіз розробленої створеної нейронної мережі як стартап-проекту. Виявлено сильні та слабкі сторони проекту, проведено

SWOT аналіз, аналіз конкурентів та цільової аудиторії. На основі отриманих результатів сформовано стратегію виходу на ринок і маркетингову стратегію для обраної цільової аудиторії.

Визначено бар'єри для входження на ринок, рівень попиту і пропозиції, цільові групи клієнтів, обрано базову стратегію розвитку.

Для досліджень було використано мову програмування Python, бібліотеки OpenCV та Keras та середовище розробки Google Colaboratory.

Створена нейронна мережа може бути використана у системах відеоспостереження шляхом прямого підключення до камери чи у засобах відеоаналітики.

У подальшому для покращення роботи мережі можна спробувати застосувати ще й попередню зміну розміру кадру, що подається на вхід мережі. Також мережу можна вдосконалити, додавши лічильник для підрахунку осіб у відеопотоці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Video Analytics. [Електронний ресурс]. URL: https://www.eetimes.com/Introduction-to-video-analytics/ (дата звернення: 06.06.22).

2. Русакова Л. О., Шаповал Н. В. Методи відстеження людини в відеопотоці // І Міжнародна науково-практична конференція «Scientific Research In The Modern World». 2022.

3. Русакова Л. О., Шаповал Н. В. Гібридна нейронна мережа для виявлення і відстеження людини у відеопотоці // І Всеукраїнська науково-практична конференція «Системні науки та інформатика». 2022.

4. Русакова Л. О., Шаповал Н. В. Метод виявлення і відстеження людини у реальному часі // XXII Міжнародна науково-технічна конференція «Штучний інтелект та інтелектуальні системи. Artificial Intelligence And Intellegent Systems (AIIS'2022)». 2022.

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

6. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition Conference (CVPR). 2001.

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

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

9. Функції активації в нейронних мережах. [Електронний ресурс]. URL: https://neurohive.io/ru/osnovy-data-science/activation-functions/ (дата звернення: 06.06.22).

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

11. Aston Z., Zachary C. L., Li M., Alexander J. S. Dive into Deep Learning. Release 1.0.0 alpha0. 2022.

12. Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks. In: Conference on Neural Information Processing Systems (NIPS). 2012.

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

14. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Conference on Neural Information Processing Systems (NIPS). 2015.

15. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

16. Hu J., Shen L., Sun G. Squeeze-and-Excitation Networks. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

17. 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.

18. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2014.

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

20. Ren S., He K., Girshick R., Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European Conference on Computer Vision (ECCV). 2014.

21. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Neural Information Processing Systems (NIPS). 2015.

22. He K., Gkioxari G., Dollar P., Girshick R. Mask R-CNN. In: International Conference on Computer Vision (ICCV). 2017. pp. 2961-2969.

23. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real time object detection. In: Conference on Computer Vision and Pattern Recognition (CVPR). 2016. pp. 779-788.

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

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

26. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. 2020.

27. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In: Computer Vision and Pattern Recognition (CVPR). 2021.

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

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

30. 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.

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

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

33. PyTorch YOLOv3. URL: https://github.com/roboflow-ai/yolov3.

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

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

36. Рейтинг мов програмування 2022. [Електронний ресурс]. URL: https://dou.ua/lenta/articles/language-rating-2022/ (дата звернення: 20.10.22).

37. Smile – Statistical Machine Intelligence and Learning Engine. URL: https://haifengl.github.io/index.html.

38. Моделювання бізнес процесів та архітектура програмного забезпечення. [Електронний ресурс]. URL: http://dspace.wunu.edu.ua/retrieve/17733/FCIT_kKN_mIPZPZS_dMBP_LEC.pdf (дата звернення: 20.10.22).

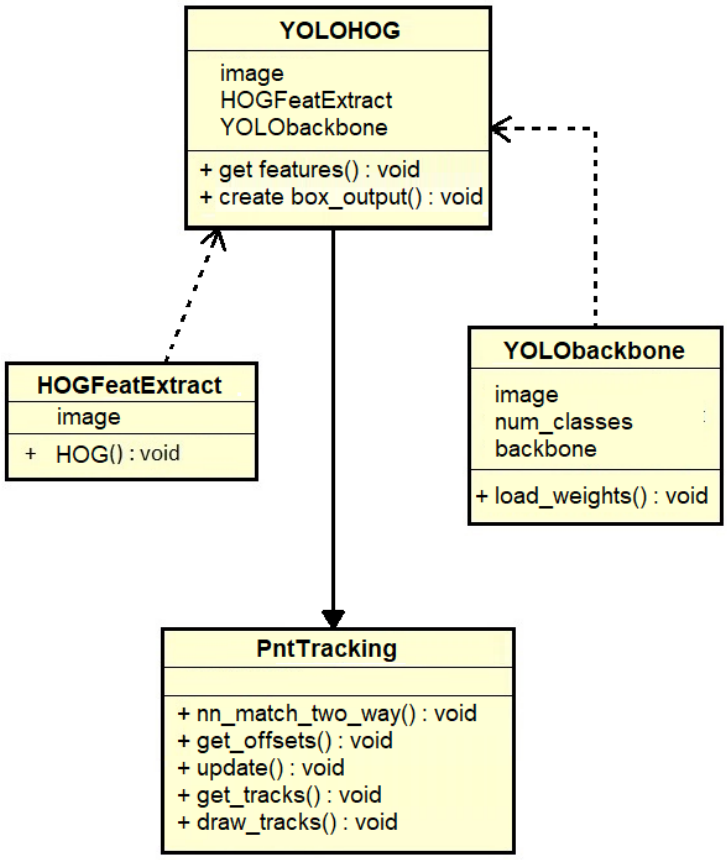
39. YOLO: Real-Time Object Detection. URL: https://pjreddie.com/darknet/yolo/.

40. SuperPoint Weights File and Demo Script. URL: https://github.com/magicleap/SuperPointPretrainedNetwork.

41. Мацуга О. М., Архангельська Ю. М., Єрещенко Н. М. Навчальний посібник до вивчення курсу «Інформаційні технології розпізнавання образів». 2016. 60 с.

ДОДАТОК А

Діаграма класів



ДОДАТОК Б

Діаграма послідовності