Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПІДРАХУНКУ ЛЮДЕЙ У НАТОВПІ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2

Гречишкін Д.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гречишкіну Данилу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку для підрахунку людей у натовпі

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, методи підрахунку кількості людей на основі теплових мап, методи підрахунку людей на основі детектування облич, навчена згорткова нейрона мережа MobileNetV2 з бібліотеки Keras, датасет ShanghaiTech, зображення з датасету WIDEFACE бібліотека комп'ютерного зору з відкритим кодом OpenCV, мова програмування Python, мова програмування Java, середовище розробки PyCharm та Android Studio IDE.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих методів підрахунку людей у натовпі.
2. Аналіз та порівняльна характеристика методів.
3. Реалізація програмного забезпечення на основі обраних методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми підрахунку натовпу, постановка задачі, сучасні методи підрахунку, вибір методу підрахунку, навчання моделі, проектування застосунку, ілюстрація роботи системи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних і програмних засобів	21.04.23-30.04.23	
5	Розробка методу	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	31.05.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____ доц. Яковлева О.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 83 с., 4 табл., 51 рис., 5 дод., 37 джерел.

ПІДРАХУНОК ЛЮДЕЙ У НАТОВПІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ДЕТЕКЦІЯ ОБЛИЧ, MCNN, CSRNET, PYTORCH, OPENCV, ANDROIDSTUDIO, REST.

Об’єктом роботи є вирішення питання підрахунку кількості людей у натовпі в реальному часі. У роботі проаналізовано різноманітні способи підрахунку людей у натовпі з використанням згорткових нейронних мереж і детекцією облич.

Метою роботи є розробка застосунку для підрахунку людей у натовпі. Особливу увагу необхідно приділити швидкості обробки фотозображень й відео, та точності підрахунку.

Дослідження базується на: згорткових нейронних мережах MCNN та CSRNet і методах детекції облич бібліотекою OpenCV. Середовищем розробки для реалізації моделі згорткової мережі виступив PyCharm. Для створення клієнтської частини було використано середовище AndroidStudio, що дозволяє тестувати застосунок на багатому наборі пристроїв, емулюючи їх.

У результаті роботи здійснена програмна реалізація клієнт-серверної системи для швидкого підрахунку людей у натовпі.

CROWD COUNTING, CONVOLUTIONAL NEURAL NETWORKS, FACE DETECTION, MCNN, CSRNET, PYTORCH, OPENCV, ANDROIDSTUDIO, REST.

The objective of this work is to solve the problem of crowd counting in real-time. Several approaches were analyzed in the scope of this work including the usage of convolutional neural networks and face detection.

The aim of this work is development of the application for crowd counting. Particular attention should be paid to the speed of photo and video processing and the accuracy of counting.

Research relies on the convolutional neural networks MCNN and CSRNet and face detection methods from OpenCV libraries. For the model part as the development environment, the PyCharm was chosen. AndroidStudio environment was used for the creation of client-side, because of its variety of emulated devices.

As a result, a comprehensive client-server solution for counting crowds in real time was provided.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КБ – комп’ютерне бачення

КЗ – комп’ютерний зір

CNN – Convolutional Neural Networks (згорткова нейронна мережа)

SGD – Stochastic Gradient Descent (стохастичний градієнтний спуск)

MCNN – Multi-Scale Convolutional Neural Networks (багатомасштабні згорткові нейронні мережі)

CSRNet – Congested Scene Recognition Convolutional Neural Networks (згорткові нейронні мережі для підрахунку перевантажених сцен)

3D – Three Dimensional (три вимірний)

GPU – Graphics Process Unit (графічний процесор)

CPU – Central Process Unit (центральний процесор)

URI (Uniform Resource Identifier) – уніфікований ідентифікатор ресурсів

IDE – Integrated Development Environment (інтегроване середовище розробки)

MVC – Model View Controller (модель, відображення, контролер)

ВСТУП

Комп’ютерне бачення – це відносно молодий напрям, який має на меті відстежування та розпізнавання об’єктів та образів [1]. Кінцевим продуктом розробок цієї царини є готова систем або машина, яка може виконувати задачі КБ. Ця галузь є доволі молодою і початком її розвиту прийнято вважати кінець 1970-х років, коли обчислювальні потужності дозволили обробляти відносно великі обсяги даних [2]. Типовими вважають наступні задачі КЗ:

– розпізнавання: воно є класичним в КЗ, обробці зображень та машинному зорі. Це завдання полягає у визначенні наявності характерного об’єкта, особливості або активності у відеоданих. Хоча це завдання вирішується легко та вірогідно людиною, досі не існує задовільного рішення для комп’ютерів у загальному випадку, зокрема, для випадкових об’єктів у випадкових ситуаціях;

– ідентифікація: для екземпляра об’єкта, такого як визначене людське обличчя або відбиток пальців, або визначення автомобіля. Ідентифікація умов у відеоданих, наприклад, виявлення можливих неправильних клітин чи тканин в медичних зображеннях;

– виявлення: воно ґрунтується на відносно простих та швидких обчисленнях, іноді використовується для знаходження невеликих ділянок в зображенні, яке аналізується, а потім досліджується додатково. Мета розпізнавання може бути вкрай різна: як виділення окремих елементів на зображенні, так і класифікація зображення в цілому. Виділяють наступні області застосування: розпізнавання текстів, створення тривимірних моделей людини по фотографіях, підрахунок об’єктів на зображеннях і так далі – розробки таких систем набирають обертів і популярності щодня [2, 3].

Задача підрахунку людей у натовпі – це одна з актуальних проблем, які можна успішно вирішувати системами комп’ютерного бачення. Підрахунок натовпу завжди був цікавим в соціально-політичному контексті. Додаткової важливості цьому питанню додала пандемія COVID-19. Організаторам

масових заходів конче необхідно слідкувати за кількістю людей, та обмежувати її відповідно до поставлених вимог. Підрахунок людей – це можливість апроксимувати можливий прибуток від туристичних об’єктів і рекламних біл-бордів.

Дана робота присвячена вирішенню питання підрахунку людей у натовпі. Розглянуто та порівняно методи на основі згорткових нейронних мереж. Для невеликих скупчень людей використано декілька методів детекції облич на зображенні.

1 СУЧАСНИЙ СТАН ПИТАННЯ ПІДРАХУНКУ ЛЮДЕЙ У НАТОВПІ

1.1 Актуальність задачі підрахунку людей у натовпі

Проблема підрахунку людей у натовпі є актуальною, оскільки вона має велике значення для безпеки, організації подій та збору статистики.

Наприклад, для планування евакуації в разі надзвичайної ситуації, потрібно знати кількість людей, які перебувають в приміщенні або на вулиці. Також підрахунок людей може бути корисним для організації масових заходів, контролю відвідуваності туристичних об’єктів та громадських місць, а також для статистичних досліджень у різних галузях, таких як маркетинг і реклама.

Оскільки люди можуть рухатися, змінювати своє положення і перекривати один одного на зображенні, задача підрахунку людей в натовпі є викликом для комп’ютерного зору та обробки зображень, і вирішення цієї проблеми може бути корисним для багатьох окремих сфер діяльності.

Вона має практичне застосування в багатьох галузях, таких як охорона громадського порядку, маркетинг та дослідження ринку, планування міської інфраструктури та транспортних систем, організація подій та багато іншого.

У багатьох випадках важливо точно знати, скільки людей знаходиться в певному місці в певний час, щоб приймати ефективні рішення [1] та забезпечувати безпеку та комфорт людей.

Частою небезпекою у натовпі є тиснява, що може призводити до вкрай трагічних випадків. До прикладу можна згадати Шанхайську тисняву 2015 року. Тому розв’язання цієї проблеми має великий практичний і соціальний потенціал.

1.2 Підходи до підрахунку людей у натовпі

Існує ціла низка способів підраховувати кількість людей у натовпі. Серед таких можна виділити: ручний, апаратний та підхід заснований на системах КЗ [2]. Кожен з зазначених способів має власні нюанси, які ми розглянемо нижче.

Ручний спосіб полягає у підрахунку за допомогою спеціальної особи, або команди. Цей спосіб потребує великий запас часу для підрахунку і має схильність до помилок при підрахунку.

Апаратний спосіб, як випливає з назви, використовує певну апаратну частину для підрахунку. Це можуть бути різноманітні сенсори, чи лазери, які встановлюються в стратегічних місцях (метрополітен, концерт холи, аеропорти тощо). Такі сенсори дозволяють мати уяву скільки людей і за який проміжок часу пройшли ділянку детекції. Такі способи показують велику точність, але мають недолік у тому, що потребують спеціальне апаратне приладдя.

Системи основані на КБ дозволяють отримати приблизну кількість людей у натовпі лише по зображенню [3 – 5]. З сильних сторін такого підходу, що в ділянці детекції треба мати лише передавач зображення (камера, дрон). Натомість негативні сторони – це створення системи, куди входить: розробка моделі, підбір апаратної частини для підрахунку.

Системи підрахунку людей у натовпі, що ґрунтуються на обробці зображень, зазвичай працюють за таким алгоритмом:

– вихідне зображення відеострічки проходить перпроцесинг, щоб знизити шум і видалити непотрібні елементи;

– далі застосовуються методи виявлення людей. Це може включати в себе використання нейронних мереж для виявлення людських силуетів на зображенні або використання методів сегментації зображень;

– після виявлення об’єктів використовується метод відстеження об’єктів, щоб відслідковувати кожного окремого людину на зображенні відео;

– зібрана інформація про кількість людей в кадрі обробляється і записується.

Деякі системи можуть використовувати додаткові технології, такі як аналіз щільності людей та їх руху на зображенні, щоб врахувати людей, які не розміщені у відкритих зонах та змінюють своє положення. Однак, враховуючи те, що натовпи можуть бути дуже різними, існує певний виклик у створення надійної та точної системи, яка зможе працювати в різних умовах [6, 7].

1.2.1 Класичні методи підрахунку людей у натовпі

Підрахунок людей у натовпі може бути здійснений за допомогою класичних методів, таких як метод текстурних ознак та метод регресії.

Метод текстурних ознак полягає в тому, що зображення натовпу аналізується з точки зору текстури. Зображення розбивається на дрібні блоки, і для кожного блоку визначаються характеристики текстури, такі як щільність, розмір та форма об’єктів. За допомогою цих характеристик можна визначити кількість людей у блоку і, відповідно, у всьому зображенні.

Деякі дослідники пропонують використовувати методи на основі регресії для підрахунку кількості людей у натовпі [7], метод регресії використовує статистичну модель, яка аналізує залежність між кількістю людей і різними параметрами зображення, такими як яскравість, контрастність та кольоровий спектр. Ця модель потім використовується для передбачення кількості людей у натовпі на основі параметрів зображення.

Обидва методи мають свої переваги та недоліки. Метод текстурних ознак може бути досить точним у визначенні кількості людей, але він може бути вразливим до змін у зовнішніх умовах, таких як освітлення, фон та перспектива. Метод регресії може бути більш стійким до таких змін, але може бути менш точним, якщо модель недостатньо точно описує залежність між параметрами зображення та кількістю людей.

1.2.2 Методи на основі згорткових нейронних мереж

Нейронна мережа (Neural network) – це комп’ютерна система, яка працює на основі послідовних шарів з’єднаних між собою «нейронів», що моделюють роботу людського мозку.

Кожен нейрон приймає вхідні сигнали, оброблює їх і передає вихідний сигнал до наступного шару нейронів. За допомогою навчання, нейронна мережа може самостійно визначати складні залежності між вхідними даними та вихідними результатами.

Нейронні мережі використовуються в багатьох сферах, таких як КЗ сприйняття, розпізнавання мови, рекомендаційні системи, обробка природних мов та багато інших.

Згорткова нейронна мережа (Convolutional Neural Network або CNN) – це спеціальний підвид нейронних мереж [8], який широко використовується в обробці зображень та відео.

Вони отримали свою назву через використання операції згортки в процесі обробки вхідних даних. Основна ідея згорткових нейронних мереж полягає в тому, щоб автоматично вивчати властивості зображень, розпізнавати ознаки та робити передбачення на основі цих властивостей.

Для цього використовуються шари згортки та пулінгу, які допомагають автоматично виявляти різні ознаки в зображеннях (наприклад, ребра, кути, текстури, форми тощо).

Згорткова нейронна мережа складається з кількох шарів згортки та пулінгу [9], які чергуються один за іншим. Після шарів згортки, отримані результати передаються до повнозв’язних шарів, де здійснюється класифікація або регресія на основі зібраних ознак.

В залежності від задачі, до згорткової нейронної мережі можуть бути додані додаткові шари або використані різні архітектури для досягнення більш точних результатів.

1.2.2.1 Особливість згорткових нейронних мереж

Одна з основних особливостей згорткових нейронних мереж – це їх здатність до ефективного виявлення та відображення локальних шаблонів в зображеннях. Це досягається за допомогою операції згортки, яка використовує фільтри або ядра для знаходження специфічних ознак у різних частинах зображення.

Згорткові нейронні мережі мають здатність до автоматичного визначення ваг для кожного фільтра під час навчання. Це означає, що навчання згорткової нейронної мережі полягає у встановленні і виявленні оптимальних значень для множини ваг, які дають найкращу ефективність при класифікації нових зображень [10 – 12].

Крім того, згорткові нейронні мережі мають декілька вихідних шарів, кожен з яких може розпізнавати різні ознаки зображення. Це дозволяє моделі відображати багатошарову структуру ознак, яка є корисною для знаходження складніших залежностей між ознаками та зображеннями.

Додатково до цього, згорткові нейронні мережі можуть бути використані не тільки для обробки зображень, але і для обробки інших типів даних, таких як звук або текст. Наприклад, для обробки звукових даних згорткові нейронні мережі можуть використовувати одновимірні згортки, а для обробки тексту – двовимірні згортки [13].

1.2.2.2 Навчання та перенавчання нейронних мереж

Навчання нейронної мережі – це процес встановлення оптимальних ваг і зсувів, щоб максимально точно передбачати вихідні значення для вхідних даних. Цей процес використовує набір тренувальних даних, який містить вхідні значення і відповідні цілі вихідні значення, які нейронна мережа намагається передбачити. Під час навчання нейронна мережа виконує

зворотнє поширення помилки (backpropagation), щоб знайти градієнти відносно ваг і зсувів, і потім використовує ці градієнти для оновлення ваг і зсувів за допомогою алгоритму оптимізації, такого як стохастичний градієнтний спуск (SGD) (рис. 1.1).

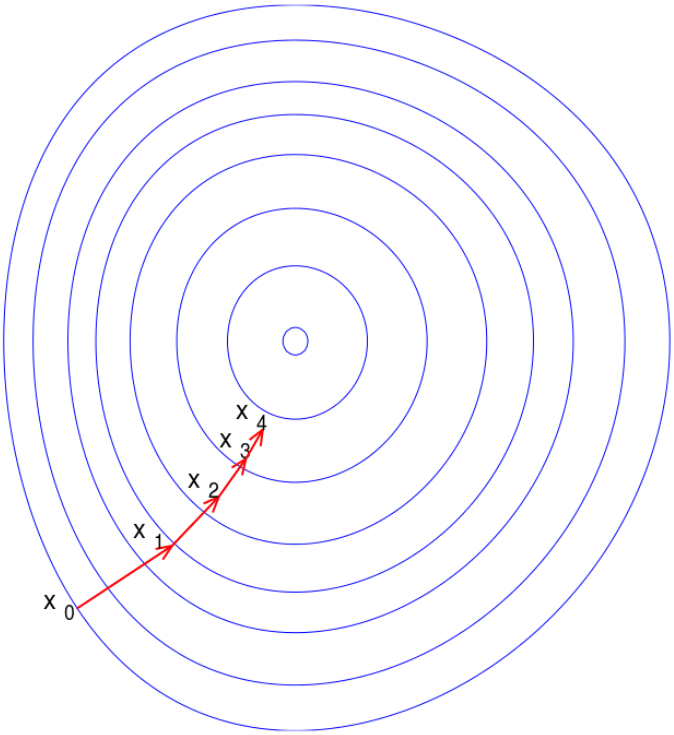


Рисунок 1.1 – Візуалізація градієнтного спуску до точки екстремуми функції

Після того, як нейронна мережа навчена на тренувальних даних, її можна перевірити на тестових даних, щоб оцінити точність поточного набору вагів. Якщо точність на тестових даних недостатня, можна застосувати техніку покращення навчання, таку як збільшення кількості прихованих шарів або збільшення розміру батчів – кількість картинок, які подаються на вхід одночасно.

Перенавчання (overfitting) – явище, коли нейронна мережа вивчає тренувальні дані занадто детально і не може коректно передбачати нові дані.

Це може статися, коли мережа має надто багато параметрів в порівнянні з кількістю тренувальних даних, або коли вона навчалася на даних, які мають багато шуму або непотрібних деталей. Для уникнення перенавчання можна

використовувати техніки регуляризації, або збільшення кількості тренувальних даних.

1.2.2.3 Приклади нейронних мереж для підрахунку людей

Існує багато моделей заснованих на згорткових нейронних мережах, але нижче будуть розглянуті лише ті, що показують найвищу точність.

Multi-Column Convolutional Neural Network (MCNN) – ця мережа створена, щоб підраховувати натовп з різною щільністю та довільним ракурсом, перспективою. Для цього впроваджено багато колонкову архітектуру для зіставлення зображення з картою щільності.

Запропонована мережа дозволяє використовувати зображення довільного розміру та роздільної здатності. Завдяки використанню фільтрів з рецептивними полями різного розміру, ознаки, адаптуються до варіацій розміру людей/голів через ефект перспективи або роздільну здатність зображення. Крім того, істинна карта щільності обчислюється точно на основі адаптивних до геометрії ядер, які не потребують знання карти перспективи вхідного зображення.

Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes (CSRNet) – це мережа для підрахунку перевантажених сцен. Вона реалізує метод глибинного навчання на основі даних і може виконувати точну оцінку кількості об’єктів, а також представляти високоякісні карти щільності. CSRNet складається з двох основних компонентів: згорткової нейронної мережі (CNN) як інтерфейсу для вилучення 2D-об’єктів і розширеної CNN для інтерфейсу, яка використовує розширені ядра для забезпечення більших пулів прийому і заміни операцій об’єднання. Вона дуже легко навчається завдяки своїй чистій згортковій структурі.

Learning Spatial Awareness to Improve Crowd Counting (SPANet) – ця мережа покликана враховувати просторовий контекст при підрахунку

натовпу. Для цього пропонується досягти максимальної втрати над пікселями шляхом знаходження субрегіону на рівні пікселів з високою розбіжністю з базовою істиною.

1.3 Метрики для оцінки якості алгоритмів підрахунку людей

Нижче розглянуто основні метрики, які використовуються при навчанні і побудові згорткових нейронних мереж.

Mean Absolute Error (MAE) – середнє абсолютне значення похибки, що вимірює різницю між реальною кількістю людей та прогнозованою кількістю

$$MAE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}. \quad (1.1)$$

Mean Squared Error (MSE) – це середнє квадратичне похибки, вимірює усереднення квадратів похибки значення та прогнозованого значення

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i)^2. \quad (1.2)$$

Root Mean Squared Error (RMSE) – квадратний корінь з MSE, який може бути використаний для зменшення впливу великих значень на оцінку похибки.

Precision та Recall – метрики, які вимірюють влучність та повноту відповідно. Для підрахунку людей, Precision відноситься до того, наскільки точні алгоритми виявляють людей, тоді як Recall відноситься до того, наскільки повні алгоритми виявляють людей на зображенні. Графічно ці метрики можна показати як множини та їх перетин (рис. 1.2).

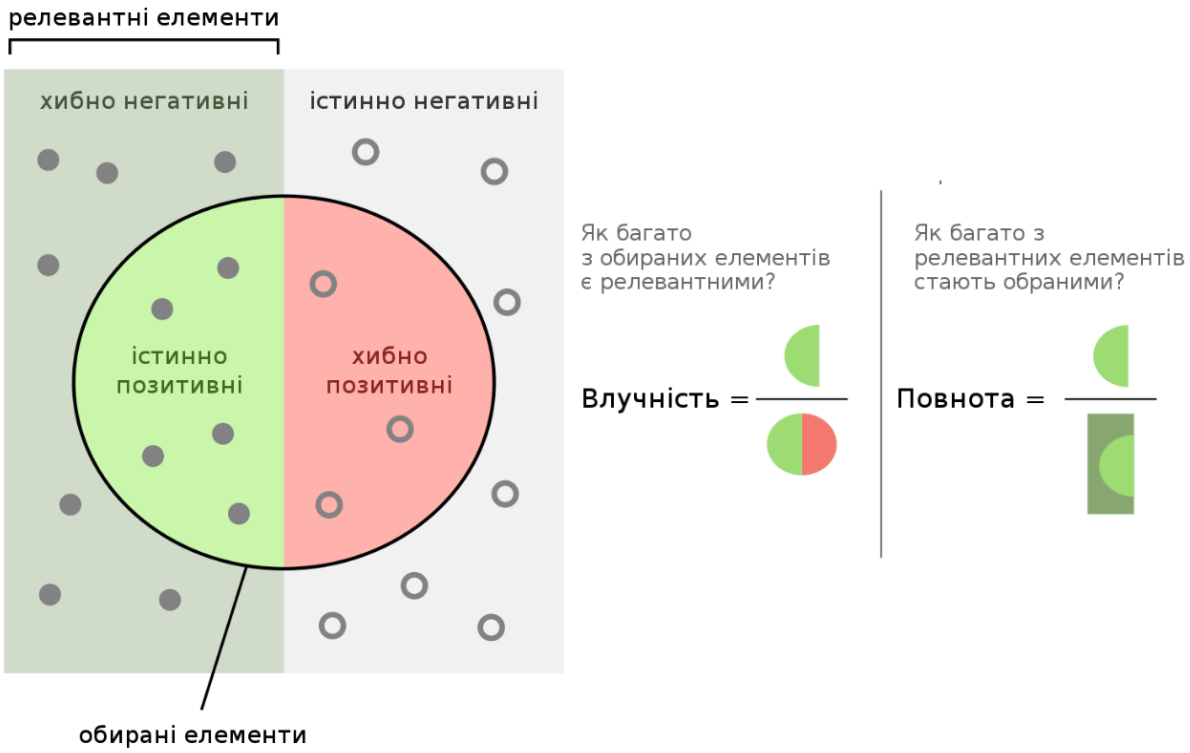


Рисунок 1.2 – Метрики Precision та Recall

F1-показник – гармонічне середнє точності та повноти, яке може бути використано для оцінки якості алгоритмів, що підраховують людей

$$F_1 = \frac{2}{\text{повнота}^{-1} + \text{влучність}^{-1}}. \quad (1.3)$$

Mean Absolute Percentage Error (MAPE) – середнє абсолютне значення відсоткової похибки, що вимірює відсоток різниці між прогнозованою кількістю та реальною кількістю людей

$$MAPE = \frac{1}{n}\sum_{j=1}^{n}\frac{|A_i - F_i|}{A_i}, \quad (1.4)$$

де A_i – кількість людей на зображені;

F_i – підрахована кількість людей;

n – кількість зображень.

1.4 Датасети зображень із зображенням натовпу

Датасети – це завчасно підготовані збірки з анотованими зображеннями. У випадку підрахунку людей, анотації це масив координат точок голів на зображеннях. Під час навчання мережі зображення датасету повинні бути максимально наближеними до зображень, з якими буде працювати навчена модель в практичному застосуванні [14]. Також кількість зображень датасету має бути достатньою для навчання до заданої точності. Тобто вибір датасету для навчання мережі для вирішення задачі підрахунку людей у натовпі залежить від умов, в яких буде цей підрахунок проводитися, та інших вимог до системи. Ось деякі мережі з тих, що можуть бути використані для вирішення задачі підрахунку людей у натовпі:

– UCF-CC-50 (University of Central Florida, 2013) складається із 50 зображень надзвичайно щільних натовпів (практично без інших об’єктів), середня кількість людей на одному зображені складає 1279 людини, всього має 63974 анотованих осіб (рис. 1.3);

Рисунок 1.3 – Приклади зображень датасету UCF-CC-50

– UCF-QNRF (University of Central Florida, 2018) містить 1535 зображень, на яких анотовано 1251642 осіб, ці зображення зібрано із інтернету, з різних куточків світу (зображення відрізняються кутами огляду натовпу, щільністю, освітленням сцени, розмірами зображення). Суттєва відмінність від UCF-CC-50, що зображення містять інші об’єкти, крім натовпу, такі як рослини, небо, дороги (рис. 1.4);

Рисунок 1.4 – Приклади зображень датасету UCF-QNRF

– JHU-CROWD++ (Johns Hopkins University, 2020) має 4372 зображень натовпу за різних погодних умов (дощ, туман, сніг), з різним освітленням (день, ніч) та певним розмиттям зображення, всього 1510000 анотацій (рис. 1.5);

Рисунок 1.5 – Приклади зображень датасету JHU-CROWD++

– ShanghaiTech датасет: включає 2 піднабори – частина A і частина B, що складаються з зображень натовпів. Частина A складається з 482 зображень з загальною кількістю 244 960 людей, а частина B містить 716 зображень з загальною кількістю 88 488 людей (рис. 1.6);

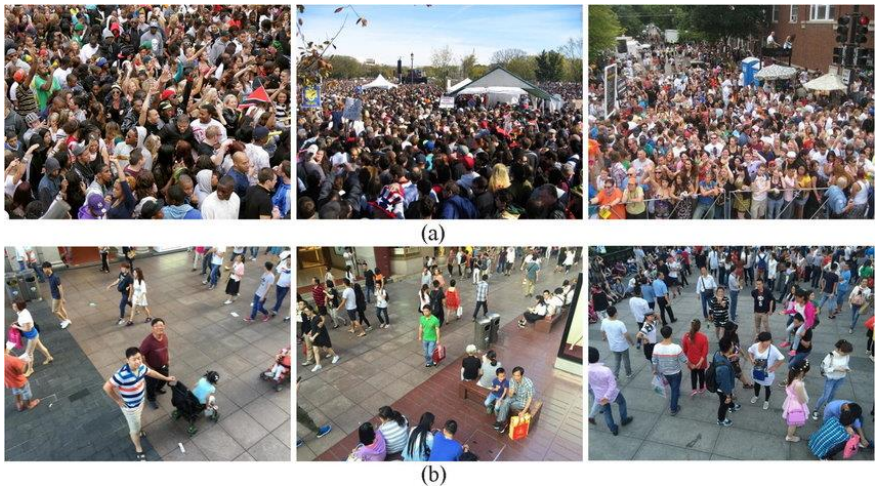


Рисунок 1.6 – Приклади зображень датасету ShanghaiTech

– WorldEXPO’10 датасет: містить зображення з 6 відео, знятих на Всесвітній виставці в Шанхаї в 2010 році. Кожне зображення містить до 43000 людей (рис. 1.7).

Рисунок 1.7 – Приклади зображень датасету WorldEXPO’10

1.5 Постановка задачі

Проблема підрахунку кількості людей у натовпі є важливою для різних сфер життя. Незважаючи на досягнуті успіхи у розв’язанні цієї задачі, існує потреба подальших досліджень у цій галузі, особливо актуальною є адаптація відомих методів і розробка нових для розв’язання задачі підрахунку в реальному часі та розробка реальних сервісів і застосунків.

Об’єктом роботи є вирішення питання підрахунку кількості людей у натовпі в реальному часі. У роботі проаналізовано різноманітні способи підрахунку людей у натовпі з використанням згорткових нейронних мереж і детекцією облич.

Метою роботи є розробка застосунку для підрахунку людей у натовпі. Особливу увагу необхідно приділити швидкості обробки фотозображень й відео, та точності підрахунку.

Задля досягнення цієї мети необхідно:

− розглянути основні принципи згорткових нейронних мереж, особливості їх навчання та перенавчання;

– проаналізувати метрики для оцінки якості підрахунку людей на основі нейронних мереж;

– провести огляд існуючих методів підрахунку людей у натовпі, та обрати найбільш перспективні для подальшого практичного дослідження на датасетах, що містять натовп;

– вивчити бібліотеки PyTorch та OpenCV для обробки та аналізу фотозображень та відео;

– спроєктувати структуру застосунку;

– вивчити середовище AndroidStudio для розробки клієнтської частини, та PyCharm для серверної;

– розробити клієнтську та серверну частини застосунку;

– зробити висновки щодо точності та швидкості розробленого застосунку.

2 МАТЕМАТИЧНА МОДЕЛЬ ПІДРАХУНКУ ЛЮДЕЙ У НАТОВПІ ТА РОЗРОБКА АЛГОРИТМУ

2.1 Опис використаних датасетів для вирішення задачі підрахунку людей

Процес навчання і обрані датасети для навчання грають ключову роль у тому, як модель буде виконувати поставлені задачі [15]. Для підрахунку людей у натовпі треба підготувати декілька класів зображень:

– щільний натовп (фото концертів, масових заходів);

– натовпу середньої щільності (зображення вулиць, скупчення людей);

– групові фото (кожну окрему людину добре видно, і можна детектувати обличчя).

Розбивши вхідні зображення на такі можливі класи, з’являється можливість покрити різні випадки зображень найбільш оптимально. Наступним кроком необхідно зібрати датасети, які зможуть покрити наведені вище вимоги.

2.1.1 Датасет ShanghaiTech

Розглянутий вище ShanghaiTech датасет вважається класичним для навчання та оцінки алгоритмів, які покликані визначати щільність людей на зображеннях. Він складається з двох частин A та B:

– частина A має ненормалізовані зображення (зображення мають різний розмір), велику щільність – це переважно концерти, фестивалі, масові заходи, виступи. 300 зображень відведено під навчання та 182 під тести та валідаційну вибірку;

– частина B має певні відмінності: всі картинки нормалізовані до одного розміру, щільність помірна – це міські вулиці, які середньо заповнені. Для навчання відведено 400 зображень, та 316 для валідації.

Таким чином модель навчена на частині А може показувати кращі успіхи у більш щільних сценах, а частина Б – напроти.

Більше прикладів зображень можна знайти в додатках А та Б відповідно до частин датасету.

2.1.2 Датасет WIDEFACE (з великими обличчями)

WIDER FACE – це датасет з обличчями людей, що містить 32203 зображень з 393703 людьми. Датасет створений спеціально для завдання визначення облич людей на зображеннях та є одним з найбільших датасетів облич людей, що доступні для загального використання.

Кожне зображення у датасеті містить від 1 до 100 облич, а кожне обличчя міститься в окремому файлі з анотаціями.

Анотації включають координати верхнього лівого та правого нижнього кутів прямокутника, який обмежує кожне обличчя. Для датасету також надані розміри зображень та частоти відображень облич.

Цей датасет може бути використаний для тренування та оцінки моделей машинного навчання для завдань детекції облич людей на зображеннях.

Його використання може бути корисним у багатьох сферах, таких як системи відео нагляду, системи безпеки, автоматичне розпізнавання облич людей у соціальних мережах та інших застосуваннях, які вимагають автоматизованого аналізу зображень з людьми.

Отже зображення з цього датасету будуть у нагоді, коли мова йде про групові фото і стоїть задача підрахунку облич, а не натовпу.

2.2 Моделі підрахунку людей у натовпі на основі теплових мап

Моделі підрахунку людей у натовпі на основі теплових мап є одними з популярних методів підрахунку людей на зображеннях. Для підрахунку людей у натовпі на основі теплових мап використовуються методи машинного навчання, які використовують нейронні мережі. У процесі навчання моделі, нейронна мережа приймає зображення людей у вигляді теплових мап та інформацію про реальну кількість людей на зображенні.

Процес навчання полягає у тому, що модель навчається розрізняти різну густину на теплових мапах, яка відповідає різній кількості людей на зображенні. Наприклад, якщо густина на тепловій карті висока, то це може вказувати на наявність великої кількості людей на зображенні. За допомогою цих моделей можна підраховувати кількість людей на зображеннях з високою точністю.

Мережа отримує зображення для навчання та теплову мапу, на якій вже є анотація людей, і таким чином мережа вчиться знаходити такі ознаки, які наближують її до генерації подібної теплової мапи до анотованої.

2.2.1 Нейронна мережа MCNN

Multi-column CNN for Crowd Counting, або скорочено MCNN – це архітектура глибокої нейронної мережі, розроблена для задачі детектування людей на зображеннях. Вона була запропонована у в 2016 році [15].

MCNN складається з трьох основних компонентів: входу, обробки та виходу. На вході подається зображення, яке проходить крізь декілька конволюційних шарів для вилучення ознак зображення різної масштабності. Кожен шар складається з кількох фільтрів, які здійснюють згортку з

попереднім шаром. Результат згортки потім піддається функції активації (наприклад, ReLU), щоб забезпечити нелінійність.

Оскільки людина може мати різний розмір на зображенні, то MCNN використовує піраміду зображень різної роздільної здатності, щоб враховувати людей з різними розмірами (рис. 2.1).

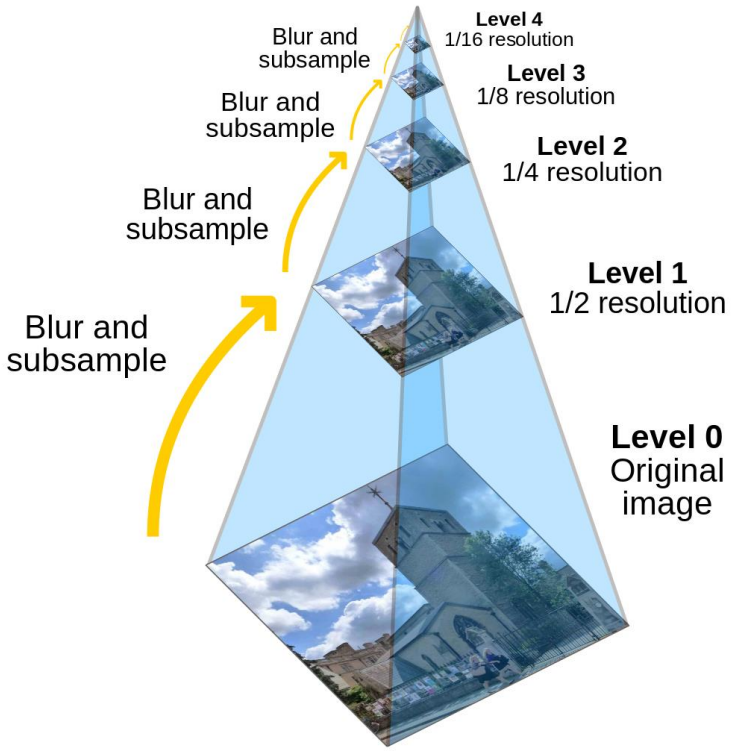


Рисунок 2.1 – Піраміда зображень різної роздільної здатності

Інформація, отримана з кожного шару, об’єднується в єдиний вектор ознак і передається на повнозв’язний шар, де відбувається остаточне прийняття рішення про наявність або відсутність людей на зображенні.

MCNN має кілька переваг у порівнянні з іншими методами детектування людей. Зокрема, вона може ефективно підраховувати людей різних розмірів на зображенні, а також дозволяє ефективно використовувати обмежені обчислювальні ресурси. Додатково, вона може бути застосована не тільки для детектування людей, але і для інших завдань підрахунку об’єктів.

2.2.2 Структура нейронної мережі MCNN

Головну роль відіграє архітектура мережі, яка складається з трьох паралельних згорткових нейронних мереж. Це пов’язано з тим, що кожна з трьох мереж має фільтри згортки різних розмірів. Щоб не ускладнювати структуру для всіх стовпців взята спільна структура. Відмінними є розміри та кількість фільтрів (рис. 2.2).

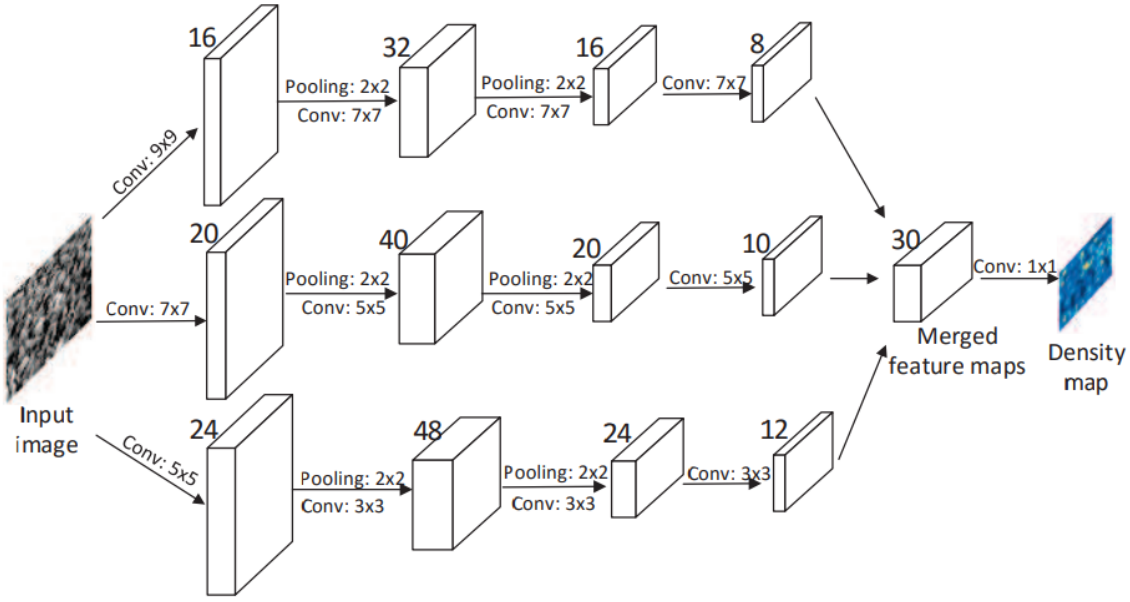


Рисунок 2.2 – Архітектура мережі MCNN

Структура першої мережі та її шарів:

– ядро згортки, розмір 9×9 з кількістю мап ознак – 16;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 7×7 з кількістю ознак – 32;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 7×7 з кількістю ознак 16;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 7×7 з кількістю ознак 8.

Структура другої мережі та її шарів:

– ядро згортки, розмір 7×7 з кількістю мап ознак – 20;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 5×5 з кількістю ознак – 40;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 5×5 з кількістю ознак 20;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 5×5 з кількістю ознак 10.

Структура третьої мережі та її шарів:

– ядро згортки, розмір 5×5 з кількістю мап ознак – 24;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 3×3 з кількістю ознак – 48;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 3×3 з кількістю ознак 24;
– вибір максимального значення з квадрату 2×2;
– ядро згортки, розмір 5×5 з кількістю ознак 12.

Після проходження крізь шари мережі, результати сходяться на останній комбінуючий шар, розмір згортки якого 1×1. На виході мережі генерується теплова мапа, або мапа щільності натовпу за допомоги якої можна підрахувати кількість людей на вхідному зображенні.

2.2.3 Нейронна мережа CSRNET

CSRNet була запропонована в 2018 році та показала дуже високу точність в задачі підрахунку людей на зображеннях [16]. Вона стала популярною моделлю для обробки зображень, пов’язаних з трафіком, людськими масовими заходами, та іншими подібними задачами. Автори виступили з критикою доцільності використання MCNN. Ідея декількох колонок полягає у тому, що кожна націлена на підрахунок густини різних сцен, але окремі шари мали вкрай спільні графіки похибки для 50 тестів, що

свідчить про те, що окремі колонки навчилися розпізнавати скоріше схожі ознаки, аніж різні (рис. 2.3).

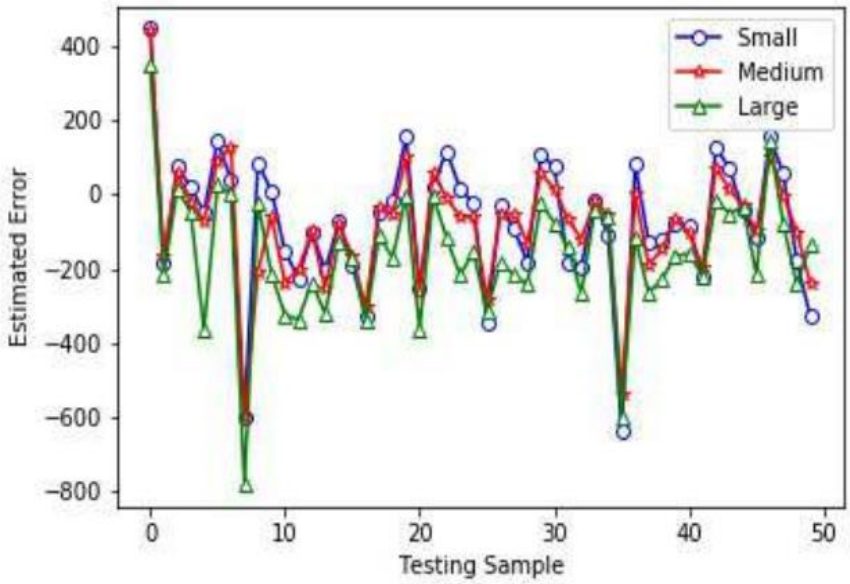


Рисунок 2.3 – Графіки похибки для окремих підмереж MCNN

2.2.4 Структура нейронної мережі CSRNET

За перший шар, або фронтенд шар, обрано VGG16 через його сильну здатність до трансферного навчання та гнучку архітектуру, що дозволяє легко поєднувати бекенд для генерації карт щільності.

Спираючись на досвід попередніх моделей, використання таких шарів без модифікацій призводить до малої продуктивності. У цій моделі спочатку виокремлюється класифікаційна частина VGG16, де всі шари повністю з’єднані, а сама CSRNet вже побудована зі згорнутими шарами VGG16.

Вхідний розмір інтерфейсної мережі складає 1×8 від вхідного розміру. Додавання згорткових шарів не призведе до покращення мап густини, а лише зменшить вхідний розмір, що є недоцільним. Фінальна архітектура представлена на рисунку 2.4.

Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1 conv3-64-1			
max-pooling			
conv3-128-1 conv3-128-1			
max-pooling			
conv3-256-1 conv3-256-1 conv3-256-1			
max-pooling			
conv3-512-1 conv3-512-1 conv3-512-1			
back-end (four different configurations)			
conv3-512-1 conv3-512-1 conv3-512-1 conv3-256-1 conv3-128-1 conv3-64-1	conv3-512-2 conv3-512-2 conv3-512-2 conv3-256-2 conv3-128-2 conv3-64-2	conv3-512-2 conv3-512-2 conv3-512-2 conv3-256-4 conv3-128-4 conv3-64-4	conv3-512-4 conv3-512-4 conv3-512-4 conv3-256-4 conv3-128-4 conv3-64-4
conv1-1-1			

Рисунок 2.4 – Архітектура мережі CSRNet

2.2.5 Створення ground truth

Перед початком навчання анотовані данні потрібно підготувати та перетворити на правдиву основу, або ground truth. Якість цієї основи напряму впливає якість навчання кінцевої моделі.

Для цього кожна картинка перетворюється на матрицю пікселів і оброблюється відносно своєї анотації. Кожна анотація до зображення представляє собою список координат, де знаходиться голова анотованої людини. Таким чином треба перетворити усі зображення та усі їх анотаційні точки на формат, який можна буде використовувати при навчанні моделі.

Об’єднання анотації та зображення до неї виглядає, як представлено на рисунку 2.5.

Рисунок 2.5 – Візуалізація анотацій

Після комбінації анотації та зображення, виходить матриця, розміру вхідного зображення, в якій одиницями заповнені точки, де на зображенні знаходяться голови, і нулями все інше. Тоді, якщо в пікселі x_i є голова, то вона представляється дельта функцією, або функцією Дірака $\delta(x - x_i)$. Маючи N голів, це можна показати так:

$$H(x) = \sum_{i=1}^{N} \delta(x - x_i). \quad (2.1)$$

Щоб перевести це в неперервну функцію щільності треба згорнути функцію за допомоги гаусіанівського ядра. Тоді густина буде дорівнювати $F(x) = H(x) * G_\sigma(x)$. Але в данному випадку, щоб підрахувати густину, треба брати до уваги те, що анотовані голови знаходяться в 3D просторі і отримане зображення, де даних про топографію відповідно немає, на виході має спотворення перспективи. Щоб вирішити ці особливості, треба прийняти наче

натовп розподілений рівномірно, з цього випливає що середня відстань між найближчіми *k*-сусідами (на зображенні) і буде обґрунтованою оцінкою спотворень, пов'язаних з перспективою [17].

Питання підбору параметру σ є доволі складним, але є компромісне рішення, яке полягає у тому, що на практиці розмір голови залежить від відстані між центрами двох сусідніх людей тому пропонується адаптивно підраховувати параметр для кожної людини виходячи з середньої відстані до сусідів, яку дорівнює:

$$d^{-i} = \frac{1}{m}\sum_{j=1}^{m} d_j^i, \tag{2.2}$$

де d_j^i – відстань до *j*-ого сусіду.

Виходячи з розрахунків вище, піксель пов'язаний з x_i, відповідає певній ділянці в просторі з приблизним в радіусом, який є пропорційним до d^{-i}. Тож, для оцінки густини натовпу навколо пікселя x_i, необхідно зробити згортку $\delta(x - x_i)$ за гаусіанівським ядром з дисперсією σ_i, яка пропорційна до d^{-i}:

$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i}(x), \tag{2.3}$$

де $\sigma_i = \beta d^{-i}$ для деякого параметра β.

Емпірично виявлено, що оптимальним значенням, яке дає найкращі вихідні результати, є β = 0,3. Фінальні мапи густини в порівнянні з початковими зображеннями продемонстровано на рисунку 2.6.

Приклади деяких прорахованих мап густини для зображень можна знайти в додатку В.

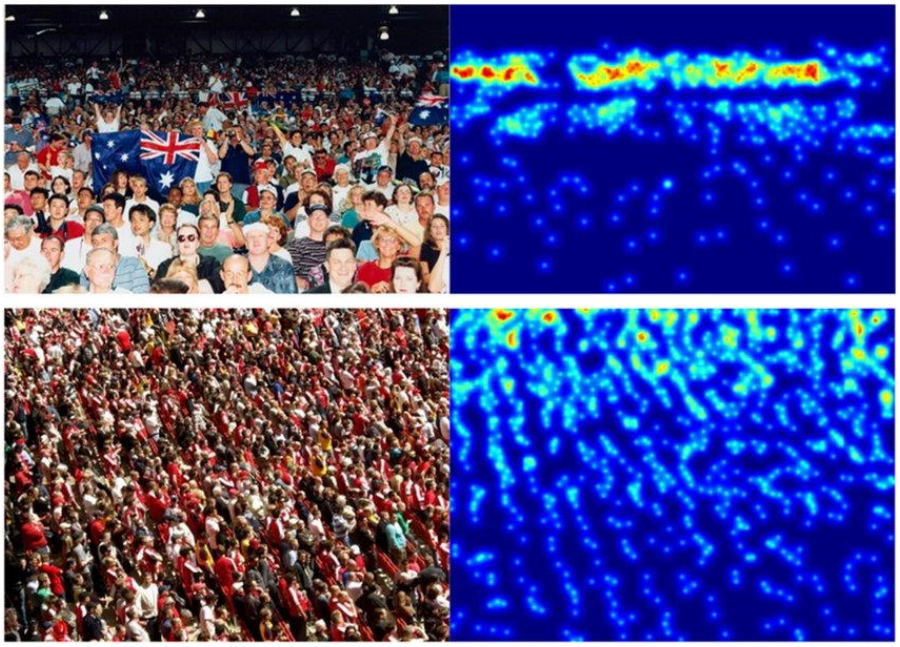


Рисунок 2.6 – Мапи щільності натовпу

2.2.6 Навчання нейронної мережі MCNN

Після підготовки ground truth починається етап навчанні. В ньому навчальна вибірка зображень пропускається крізь мережу і різниця вихідної теплової мапи та ground truth мапи підраховується функцією витрат для того, щоб оцінити точність заданої моделі. В моделі в якості функції витрат використовується функція Евклідової відстані, яка і знаходить різницю між вихідним значенням і ground truth:

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}\left|\left|F(X_i;\theta) - F_i\right|\right|_2^2, \quad (2.4)$$

де N – розмір навчальної партії;

$F(X_i;\theta)$ – представляє мапу густини від зображення з параметрами, позначеними як θ;

F_i – правдива основа для цього зображення.

2.3 Підрахунок фронтальних облич на основі каскадів Haar

Підрахунок фронтальних облич на основі каскадів Haar – це техніка комп’ютерного зору для детектування облич на зображенні. Вона базується на використанні каскадів класифікаторів, які використовують характеристики Хаар для визначення облич на зображенні. Характеристики Хаар – це деякі прості функції, які обчислюються на підмножинах зображення та дозволяють описувати його властивості, такі як контрастність та текстура. Наприклад, одна з характеристик Хаар може визначати різницю у сумі пікселів між двома прямокутними областями на зображенні.

Каскад класифікаторів – це послідовність класифікаторів, які використовуються для визначення облич на зображенні. Кожен класифікатор в каскаді складається з декількох класифікаторів Хаар, які обчислюють характеристики зображення та визначають, чи є на зображенні обличчя. Якщо один з класифікаторів в каскаді визначає, що обличчя немає на зображенні, то обробка зображення завершується і переходить до наступного зображення.

Каскади Haar є досить ефективною технікою для підрахунку фронтальних облич на зображенні. Вони знаходять широке застосування у багатьох областях, де потрібно визначати присутність облич на зображеннях, таких як системи безпеки, автомобільна промисловість, відеоспостереження тощо. Приклади анотованих облич можна знайти в додатку Г.

2.4 Підрахунок людей на основі DSFD

DSFD (Deep Scale-Space Face Detector) – це алгоритм машинного навчання для виявлення облич у зображеннях. Основними нововведеннями є такі зміни у ключових аспектах детекції облич:

– покращене навчання ознак;

– прогресивний дизайн втрат;

– доповнення даних на основі прив’язки до якорної точки.

Для покращення вихідних карт ознак використовується модуль Feature Enhance Module (FEM). Також використовується прогресивна втрата для якірних точок – Progressive Anchor Loss (PAL), що обчислюється за допомогою двох різних наборів опорних точок. Останнім нововведенням щодо якірних точок є використання покращеного узгодження – Improved Anchor Matching (IAM).

DSFD є алгоритмом машинного навчання, який використовує двопотоковий дизайн, що дозволяє йому бути надійнішим і ефективнішим у виявленні облич. Результати широких експериментів на популярних бенчмарках, таких як WIDER FACE та FDDB, підтверджують перевагу DSFD порівняно з іншими найновішими детекторами облич.

2.5 Виявлення типу зображень

2.5.1 Нейронна мережа MobileNetV2

MobileNetV2 – це нейронна мережа глибокого навчання, що покликана обробляти зображення на мобільних пристроях з обмеженими ресурсами. Ця мережа є покращенням оригінальної MobileNet, яка була випущена в 2017 році. MobileNetV2 використовує нові архітектурні ідеї, такі як «блок легких з’єднань» та «листопад», що дозволяють досягати кращих результатів при незначному збільшенні кількості параметрів мережі.

MobileNetV2 має більш широку архітектурну гнучкість, що дозволяє вибирати розмір мережі в залежності від конкретних вимог та поставленної задачі.

Це забезпечує зручність використання мережі на різних пристроях з різними обмеженнями відносно ресурси. З переваг мережі можна виділити наступні сильні сторони:

– має невелику вагу, тим самим забезпечує високу швидкість виконання;

– має менший розмір у порівнянні з MobileNetV1, що робить її більш придатною для вбудованих систем і мобільних пристроїв;

– значно зменшує кількість параметрів, що роблять її менш складною.

2.5.2 Формування датасету

Зображення для моделі повинні максимально підходити і бути схожими на ті, які мережа буде отримувати вже при використанні. Тому треба приділити особливу увагу обраним зображенням.

Питання успішної класифікації залежить від підготованого набору зображень. У ньому всі зображення вже повинні належати до певних класів, себто, це певний датасет з класифікованими зображеннями.

Для нейронної мережі стануть у нагоді зображення з датасетів, які були розглянуті раніше.

Для формування декількох класів зображень необхідно підготувати зображення трьох класів:

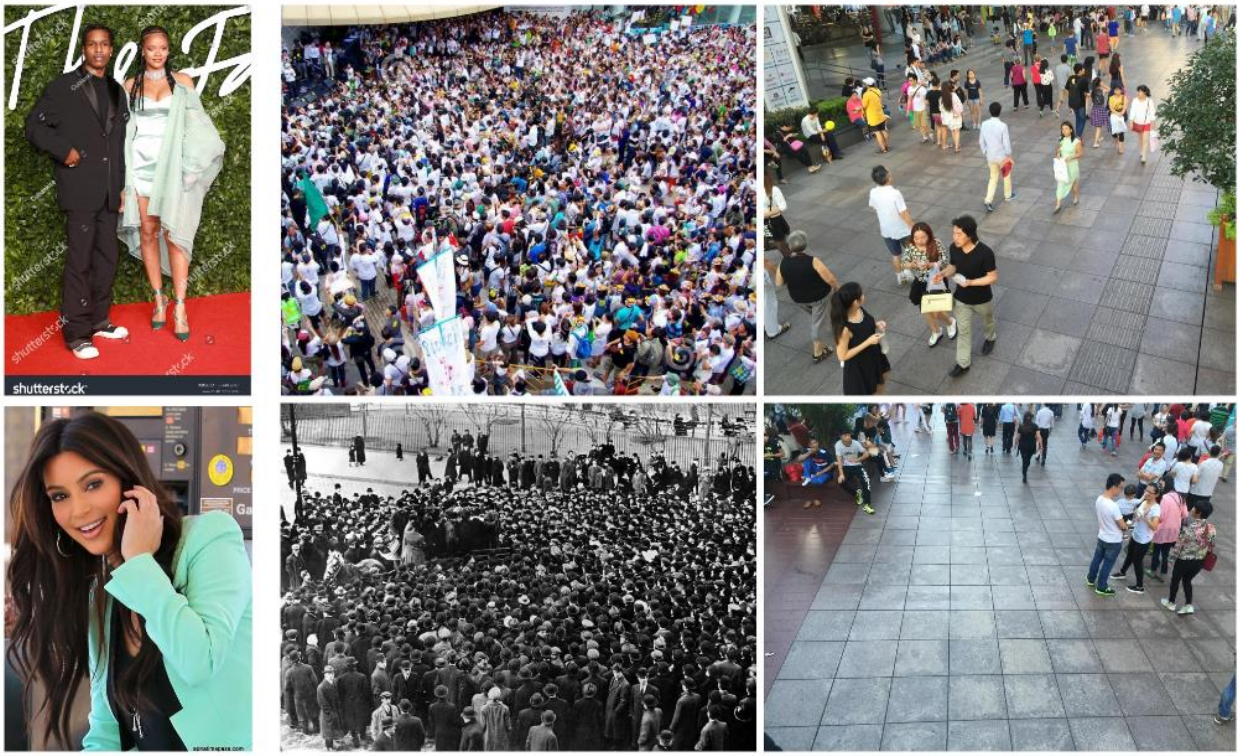
– групові зображення, що мають фронтальні обличчя крупним планом (сімейні фото, фото знаменитих людей тощо);

– щільний натовп представлений в А частині Шанхайського датасету (фото з масових заходів, концертів, виступів);

– середній натовп представлений в Шанхайському датасеті, його В частині (фото з вулиць та провулків).

Таким чином для навчання мережі-класифікатора необхідно відібрати щонайменше 400 зображень кожного класу, виділити з них ті, що будуть подаватися на тренування, наприклад, 300 зображень, і ті, що будуть валідаційними, остача.

Останні зображення мережа отримає на обробку вже після навчання (рис. 2.7). Розбиття зображень на валідаційні і тестові дозволить навчити мережу і перевірити якість навченої моделі за допомогою метрик.



(а) (б) (в)

Рисунок 2.7 – Зображення з різних класів:
(а) групові фото з фронтальними обличчями; (б) щільний натовп;
(в) помірний натовп

2.5.3 Перенавчання для класифікації зображень (натовп/великі обличчя)

MobileNetV2 є досить ефективною мережею для класифікації зображень. Для класифікації зображень за допомогою MobileNetV2, їх спочатку нормалізують: зображення змінюють розмір до 224×224 (або іншого фіксованого розміру).

Після цього, зображення подається в мережу, яка виконує кілька згорткових та пулінгових шарів, що послідовно зменшують розмір зображення та підвищують рівень його абстракції. Далі, отримані вектори ознак подаються в повнозв'язний шар, який відповідає за визначення кінцевого класу зображення.

В ході роботи було навчено модель класифікатора. Графіки навчання представлені на рисунку 2.8, а відповідні метрики представлені у таблиці 2.1. Результати класифікації зображень представлені на рисунку 2.9.

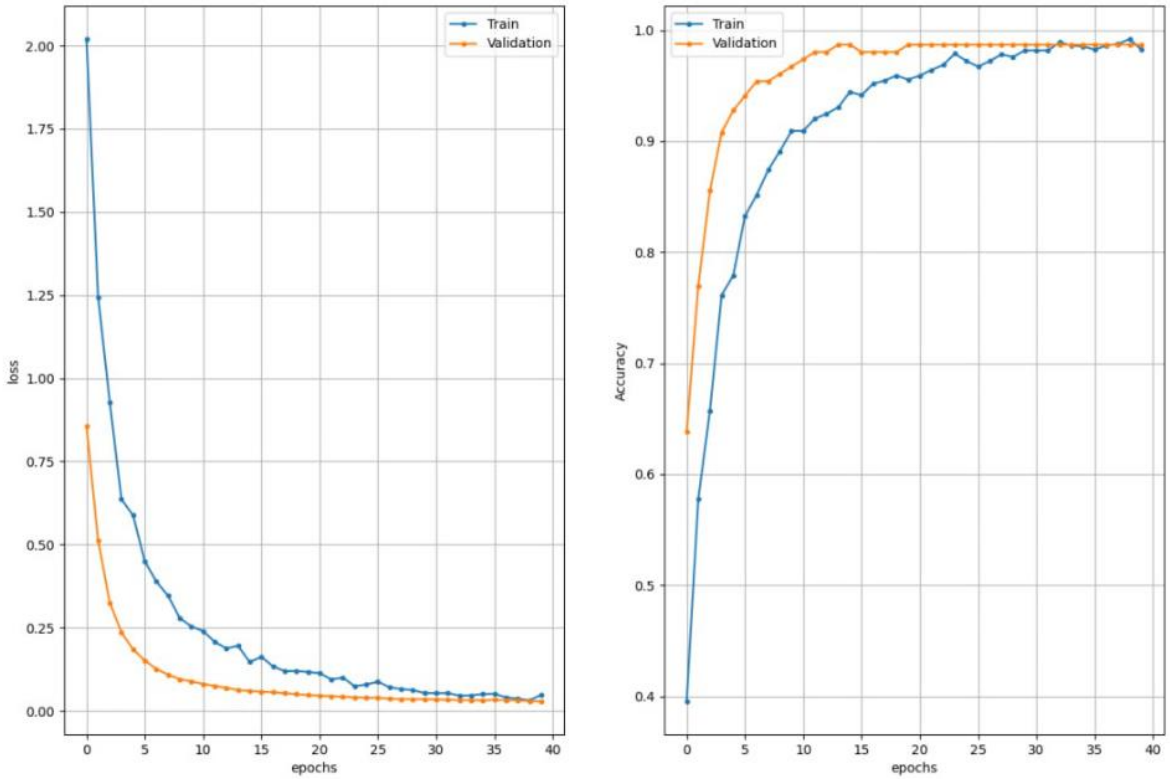


Рисунок 2.8 – Графіки навчання класифікатора зображень

Таблиця 2.1 – Метрики навчання класифікатора зображень

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	1,00	0,98	0,99	92
1	0,99	1	0,99	136
2	0,99	0,99	0,99	152

Як видно з метрик, мережа-класифікатор все ще може бути вдосконалена, але вже виконує завдання класифікації доволі впевнено відрізняючи зображення з фронтальними обличчями та зображення різної щільності.

Рисунок 2.9 – Результати класифікації на тестовій вибірці

2.6 Дослідження моделей Haar, DSFD, MCNN, CSRNET у порівняльному аспекті

Перш ніж інвестувати час в реалізації певного методу, треба дослідити наявні моделі та знайти оптимальні для поточної задачі та поставлених вимог [18]. Для детекції облич існує багато методів і підходів. Для детекції облич в порівняльному аспекті розглянуті Haar Cascade та DSFD.

Haar Cascade швидший за DSFD у виявленні облич. Це пов’язано з тим, що він використовує простішу техніку виділення ознак. DSFD, з іншого боку, вимагає більше обчислювальних ресурсів і обробляє зображення довше.

Однак DSFD є більш надійним, ніж Haar Cascade, коли йдеться про виявлення облич у складних умовах, таких як погане освітлення, оклюзія та різні пози.

Haar Cascade потребує меншого набору даних для навчання порівняно з DSFD. Haar Cascade можна навчити на кількох сотнях позитивних і негативних зображень, тоді як DSFD вимагає для навчання тисячі анотованих зображень облич.

Haar Cascade часто використовується для виявлення облич у реальному часі, наприклад, для відеоспостереження. Таким чином, метод Haar Cascade є більш оптимальним для цього проєкту.

Для підрахунку людей методом теплових мап розглянуто CSRNet та MCNN. Архітектура CSRNet загалом точніше вирішує завдання підрахунку за MCNN (табл. 2.2). Тим не менш MCNN працює швидше, ніж CSRNet. Це пояснюється тим, що MCNN використовує простішу архітектуру з меншою кількістю параметрів [19, 20].

Таблиця 2.2 – Похибка при розрахунку згортковими нейронними мережами

	Part A		**Part B**	
Models	MAE	MSE	MAE	MSE
MCNN	110,2	173,2	26,4	41,3
CSRNet	68,2	131,376	10,6	16,282

Через те, що усі розрахунки будуть відбуватися на сервері, є можливість використовувати більш точний, але більш залежний до гарного обладнання метод [17].

В середньому методи показали результати у швидкості виконання операції, як показано в таблиці 2.3.

Таблиця 2.3 – Середній час виконання методів

Метод	**Час виконання, с**
faceHaar	0,029
DSFDDetector	0,416
CSRNet	0,0035
MCNN	0,0023

Отже, DSFDDetector дійсно занадто повільний для знаходження облич в реальному часі. Щодо методу теплових мап, MCNN значно програє CSRNet в показниках MAE та MSE, тому варто обрати другу архітектуру.

2.7 Розробка алгоритму підрахунку людей

Алгоритм підрахунку людей складається з проходження зображенням модулів, які окремо покликані розрізняти зображення різного наповнення і додаткового модуля, який класифікує зображення що надійшло до системи на вхід.

Формально алгоритм можна описати послідовністю таких кроків:

Крок 1. Запуск системи.

Крок 2. До системи надходить зображення.

Крок 3. Зображення потрапляє в класифікатор, який відносить його до певного класу.

Крок 4. Відповідно до класу зображення надходить в модуль (мережа навчена на частині A/B або детекція облич на основі каскадів Haar.

Крок 5. Отриманий результат виводиться.

Крок 6. Кінець.

Описана вище послідовність дій може бути проілюстрована блок-схемою (рис. 2.10).

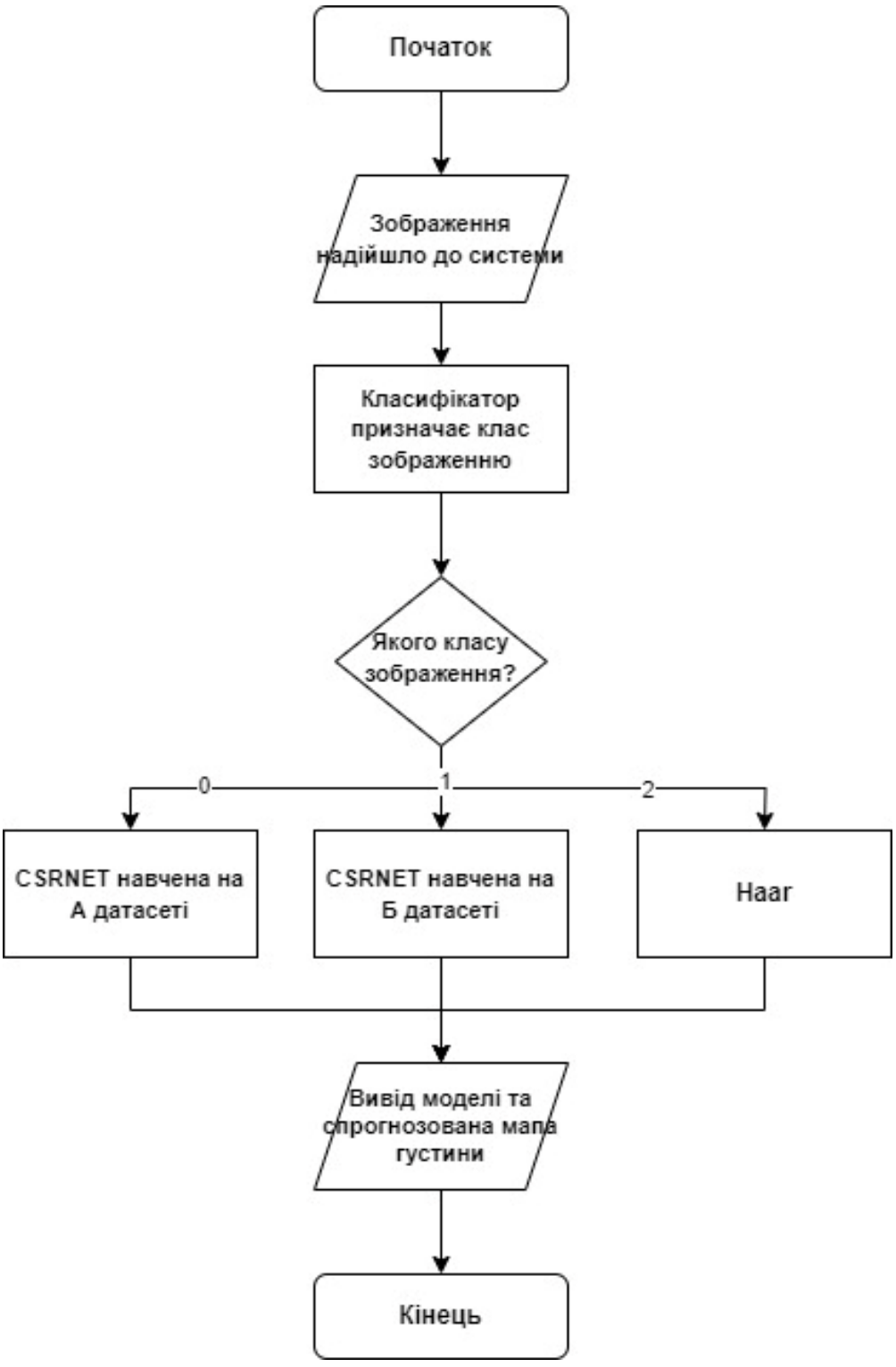


Рисунок 2.10 – Алгоритм логіки серверного компоненту

3 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПІДРАХУНКУ ЛЮДЕЙ У НАТОВПІ

3.1 Налаштування програмного середовища та використані інструменти

У рамках кваліфікаційної роботи було розроблено застосунок для підрахунку натовпу за допомогою згорткових нейронних мереж. Для реалізації серверної частини було обрано мову Python 3.9, на це вплинути такі причини:

– легкість використання: ця мова є простою та зрозумілою, що дозволяє швидко вивчати та розробляти код для нейронних мереж;

– різноманітність бібліотек: він має велику кількість бібліотек для машинного навчання та нейронних мереж, таких як TensorFlow, Keras, PyTorch, та інші – все це дозволяє швидко створювати та розгортати складні моделі нейронних мереж. Рейтинг бібліотек за популярністю наведено у таблиці 3.1;

– крос-платформеність: мова працює на більшості операційних систем, що дозволяє розробникам створювати та запускати моделі нейронних мереж на різних платформах;

– підтримка спільноти: Python має велику та активну спільноту розробників, які розробляють нові бібліотеки, документацію та відповідають на запитання користувачів;

– швидкість розробки: він дозволяє швидко розробляти та тестувати різні моделі нейронних мереж, що дозволяє розробникам швидко прототипувати нові ідеї та експериментувати з різними архітектурами мереж [21].

Таблиця 3.1 – Рейтинг бібліотек за популярністю

Library	Rank	Overall	Github	StackOverflow	Google Results
tensorflow	1	10,87	4,25	4,37	2,24
keras	2	1,93	0,61	0,83	0,48
caffe	3	1,86	1	0,3	0,55
theano	4	0,76	-0,16	0,36	0,55
pytorch	5	0,48	-0,2	-0,3	0,98
sonnet	6	0,43	-0,33	-0,36	1,12
mxnet	7	0,10	0,12	-0,31	0,28
torch	8	0,01	-0,15	-0,01	0,17
cntk	9	-0,02	0,1	-0,28	0,17
dlib	10	-0,60	-0,4	-0,22	0,02
caffe2	11	-0,67	-0,27	-0,36	-0,04
chainer	12	-0,70	-0,4	-0,23	-0,07
paddledapple	13	-0,83	-0,27	-0,37	-0,2
deeplearning4j	14	-0,89	-0,06	-0,32	-0,51
lasagne	15	-1.11	-0,38	-0,29	-0,44
bigdl	16	-1,13	-0,46	-0,37	-0,3
dynet	17	-1,25	-0,47	-0,37	-0,42
apache singa	18	-1,34	-0,5	-0,37	-0,47
nvidia difits	19	-1,39	-0,41	-0,35	-0,64
matconvnet	20	-1,41	-0,49	-0,35	-0,58
tflear	21	-1,45	-0,23	-0,28	-0,94
nervana neon	22	-1,65	-0,39	-0,37	-0,89
openn	23	-1,97	-0,53	-0,37	-1,07

Рушієм процесу навчання буде бібліотека PyTorch 2 версії, спеціальна ревізія, яка підтримує використання дискретної відео карти (завдяки GPU можна суттєво прискорити процес навчання у порівнянні з використанням CPU).

Використання PyTorch – це певний компроміс, де вже є інструменти для розробки, але в користувача є можливість гнучко користуватися ними на відміну від інших аналогічних бібліотек.

Для роботи з бібліотеками у мові Python використовуються менеджери.

pip – це менеджер пакетів для мови програмування Python. Він дозволяє зручно управляти пакетами Python, що використовуються в проєкті, та автоматично встановлює залежності для цих пакетів.

Коли розробник потребує використати певний пакет Python у своєму проєкті, він може встановити його за допомогою pip.

Наприклад, якщо розробник хоче встановити бібліотеку NumPy, яка використовується для роботи з числовими масивами в Python, він може виконати одну просту команду, яка встановлює бібліотеку NumPy з офіційного репозиторію PyPI (Python Package Index) за допомогою pip.

Крім того, pip дозволяє встановлювати пакети з локальних файлів, видаляти пакети та оновлювати їх до новіших версій. Окрім нього існують і інші пакетні менеджери (рис. 3.1):

– conda: менеджер пакетів, що входить до складу Anaconda. Conda може використовуватися для управління залежностями Python, а також для встановлення та управління пакетами, що не пов’язані з Python;

– poetry: відносно новий менеджер пакетів Python, який дозволяє легко управляти залежностями Python та забезпечує зручний інтерфейс командного рядка. Він ще не встиг набути широкого застосування [22];

– easy_install: старіший менеджер пакетів Python, який був створений до появи pip. Він все ще підтримується та може бути використаний для встановлення пакетів Python, але більшість розробників вважають його застарілим і обирають pip, як більш сучасний та зручний;

– pipenv: менеджер пакетів, який комбінує pip та virtualenv в одному інструменті. Він дозволяє розробникам управляти залежностями та віртуальними середовищами Python для проєктів одночасно;

– pyenv: це менеджер середовищ Python, який дозволяє встановлювати та керувати версіями Python на локальному комп’ютері, зазвичай його використовують у зв’язці з іншим менеджером який дозволить встановлювати бібліотеки, наприклад, вищезгаданий poetry.

Щоб зробити комунікації між клієнтом і сервером можливими було обрано використовувати RESTFul методологію, яка є підрозвидом REST.

	Installing python packages	Installing non-python packages	Managing python versions	Managing virtual environments	Environment reproducibility
pip	✅	✖ *			
venv				✅	
piptools					✅
pyenv			✅		
conda	✅	✅	✅ *	✅	
pipenv (+pyenv)	✅	✅	✅	✅	✅
poetry (+pyenv)	✅	✅	✅	✅	✅

Рисунок 3.1 – Порівняльна характеристика менеджерів бібліотек

REST (Representational State Transfer) – це архітектурний стиль, що використовується при проєктуванні розподілених систем, які мають вебсервіси. RESTFul – це підтип REST, який передбачає використання HTTP-протоколу для обміну даними між клієнтом та сервером.

RESTFul використовує різні HTTP-методи (GET, POST, PUT, DELETE тощо), щоб дозволити клієнтам звертатися до ресурсів на сервері та взаємодіяти з ними. RESTFul API (Application Programming Interface) може бути описаний за допомогою стандарту OpenAPI (раніше відомий як Swagger), що дозволяє описувати ресурси, доступні HTTP-методи для кожного ресурсу, параметри запиту та відповіді.

У RESTFul існують певні принципи, які характеризують його.

По-перше, це використання HTTP-протоколу для передачі даних між клієнтом та сервером. По-друге, ресурси повинні бути ідентифіковані за допомогою спеціального URI (Uniform Resource Identifier).

По-третє, використання HTTP-методів для звернення до ресурсів (наприклад, GET для отримання даних, POST для додавання даних, PUT для

оновлення даних, DELETE для видалення даних). А також використання стандартних HTTP-статус-кодів у відповідях на запити (наприклад, 200 OK, 404 Not Found, 500 Internal Server Error).

Використання різних форматів даних для передачі даних між клієнтом та сервером, таких як JSON або XML.

RESTFul API є дуже популярним варіантом комунікації між вебсервісами, оскільки він дозволяє клієнтам легко взаємодіяти з сервером, зменшуючи кількість зайвого коду на боці клієнту (рис. 3.2).

Крім того, використання HTTP-протоколу дозволяє RESTFul API бути масштабованим та стійким до помилок (рис. 3.2).

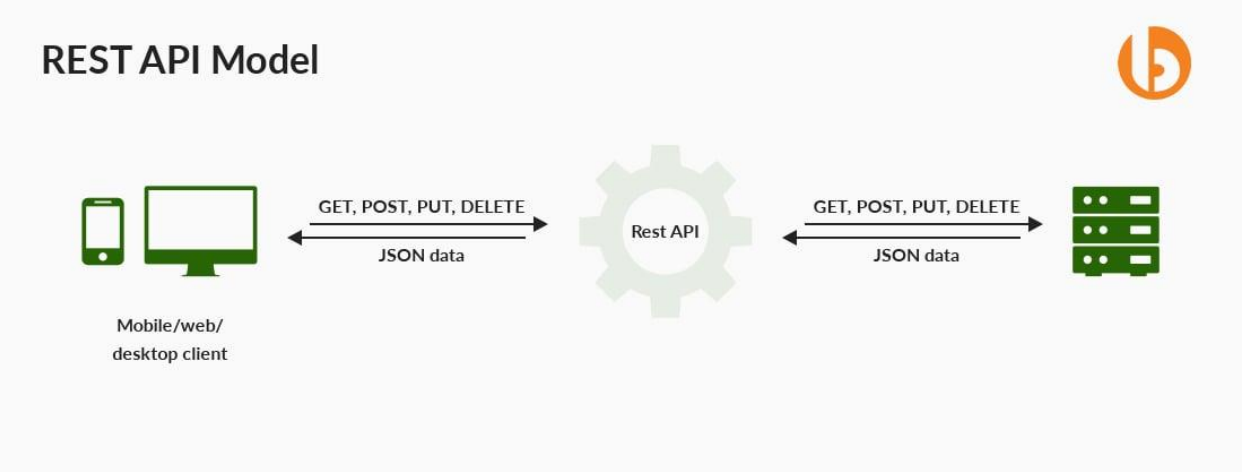


Рисунок 3.2 – Ілюстрація роботи REST API

Так як мережа використовує REST доцільно використовувати певний інструмент, який дозволить тестувати endpoint-посилання серверного компоненту, інша назва URI. Наприклад, якщо на етапі створення серверної частини клієнтського застосунку ще немає, але необхідно протестувати чи працює передача даних.

3.1.1 Інструменти для тестування та відлагодження API

Postman – це інструмент для тестування та відлагодження API. Він надає можливість відправляти HTTP запити до вебсервера та отримувати відповіді. Postman можна використовувати для тестування вебсервісів, RESTful API та інших HTTP заснованих сервісів.

Postman має інтуїтивний інтерфейс, що дозволяє легко налаштувати запити та отримувати відповіді. За допомогою Postman можна відправляти запити з різними параметрами, включаючи заголовки запиту, параметри запиту, тіло запиту тощо.

Крім того, Postman також надає можливість зберігати запити та колекції запитів, що дозволяє повторно використовувати їх для тестування та документування API.

Postman можна використовувати на різних етапах розробки програмного забезпечення. Наприклад, його можна використовувати на етапі тестування, щоб перевірити, чи повертає API очікувані результати. Також Postman можна використовувати на етапі розробки, щоб швидко перевіряти різні сценарії взаємодії з API.

Загалом, Postman є потужним та корисним інструментом для розробки та тестування API, що дозволяє ефективно виконувати різні завдання, пов’язані з розробкою вебсервісів та програмного забезпечення.

Наприклад, успішний запит на перевірку чи працює сервер в Postman буде виглядати, як показано на рисунку 3.3.

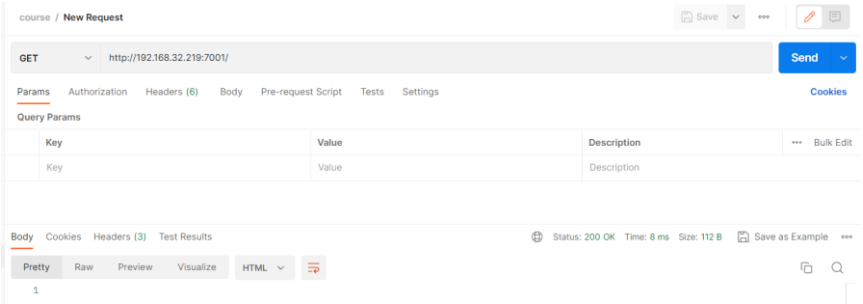


Рисунок 3.3 – Зображення інтерфейсу програми

3.1.2 Інструменти для розробки мобільних застосунків

Для реалізації клієнтської, інтерфейсної частини, було обрано середовище Android Studio, яке дозволяє розробляти застосунки для операційної системи Android, та найголовніше – тестувати їх повною мірою.

Android Studio є офіційним інтегрованим середовище розробки (IDE) для розробки мобільних застосунків на платформі Android. Нижче описані деякі з переваг розробки з використанням Android Studio.

Android Studio побудований на основі IntelliJ IDEA, що забезпечує надійну та швидку роботу. Він також має вбудовану систему відлагодження, яка допомагає виявляти та виправляти помилки в коді.

Android Studio має велику підтримку від Google, тому він містить у собі всі необхідні інструменти для розробки застосунків на платформі Android. Він також містить вбудований емулятор Android, який дозволяє перевіряти застосунки безпосередньо в середовищі розробки.

Наявність шаблонів та інструментів: він містить велику кількість шаблонів та інструментів, що допомагають розробникам швидко створювати нові застосунки та додавати функціональність до існуючих.

Підтримка Kotlin: Android Studio надає повну підтримку мови програмування Kotlin, що дозволяє розробникам створювати більш читабельний та зрозумілий код, а також зменшує кількість помилок у коді.

Легка інтеграція з іншими інструментами: Android Studio легко інтегрується з іншими інструментами розробки, такими як Git, GitHub та Gradle, що дозволяє зручно керувати процесом розробки.

Загалом, Android Studio – це потужне та ефективне середовище розробки, яке допомагає розробникам створювати високоякісні мобільні застосунки на платформі Android.

Розробники Android Studio мають широкий набір інструментів, які використовуються у розробці. Для збірки проєкту разом з усіма бібліотеками

використовується Gradle менеджер. Він відповідає за додаванням бібліотек до проєкту та його збірку.

Задля перевірки написаного коду передбачено декілька видів налагодження. Перша з опції – це використання вбудованого емулятора. Користувач самостійно обирає модель смартфону та версію операційної системи для тестування. Після конфігурації у вікні емулятора з’явиться обраний смартфон.

Таким чином, в розробника є можливість протестувати свій застосунок на різних дисплеях, моделях та версіях девайсів, що може бути життєвою необхідністю для великих застосунків, які розроблюються на декілька девайсів одночасно. До прикладу, так це вікно виглядає, коли користувач запускає емуляцію Pixel смартфону (рис. 3.4).

В розширених налаштуваннях користувач має змогу змінювати кількість заряду батареї, його геолокацію та навіть що робити, якщо користувач заходить у камеру(наприклад, вивід зображення з вебкамери).

Але це не єдиний режим налагодження. Користувачі, які мають фізичні смартфони з операційною систему Android, можуть виконувати налагодження на власному смартфоні, для цього достатньо підключити його по USB до комп’ютера, або під’єднатися по WI-FI, завдяки коду.

Останній спосіб налагодження дуже зручний, коли розробляється застосунок, в якому смартфон активно використовується. Клієнтський застосунок реалізовано на мові Java 8 версії, яка і досі є дуже популярною (рис. 3.5).

Ось деякі переваги та недоліки використання Java 8 для розробки мобільних застосунків:

– широке сприйняття: Java 8 є однією з найбільш популярних версій мови Java, тому вона має велику спільноту розробників та документацію, що допомагає розробникам швидше та ефективніше створювати мобільні застосунки;

– нові можливості: Java 8 має нові функціональні можливості, такі як лямбда-вирази та Stream API, які дозволяють розробникам більш зручно та ефективно писати програмний код;

– сумісність зі старим кодом: Java 8 є сумісною зі старим кодом на мові Java, тому розробники можуть використовувати старий код та покращувати його, додавши нові функціональні можливості.

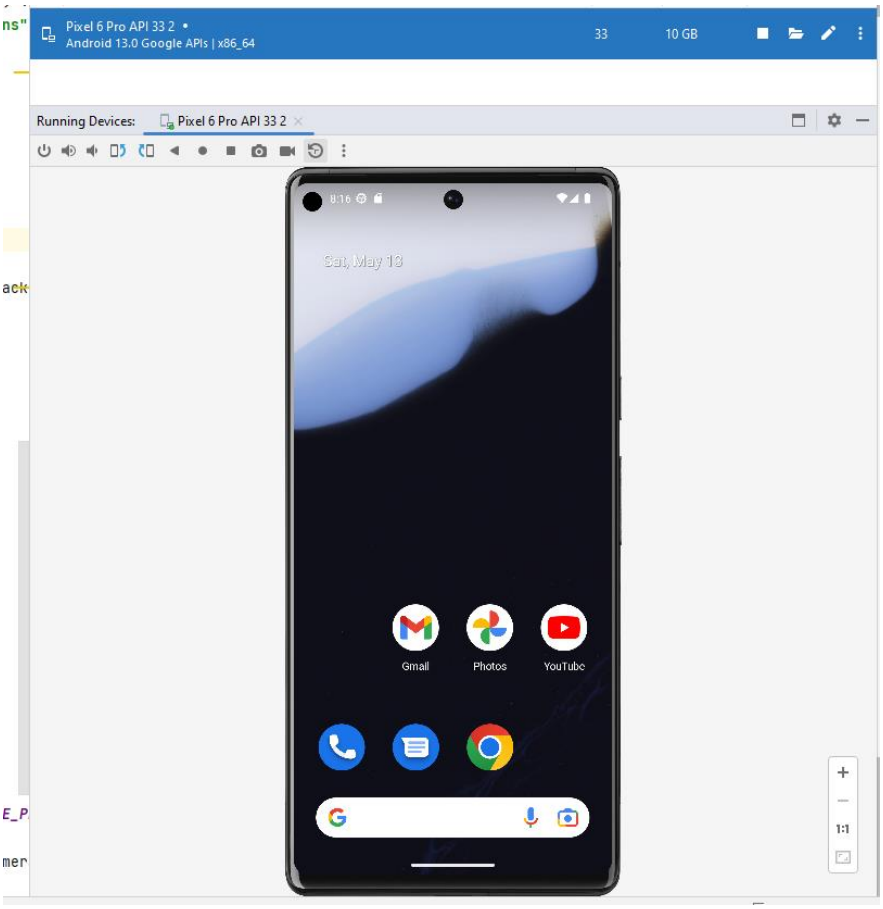


Рисунок 3.4 – Зображення інтерфейсу емулятора

Використання Java 8 має кілька недоліків, особливо у порівнянні з більш новими версіями мови. Ось декілька з них:

– відсутність нових функціональних можливостей. Java 8 не має всіх нововведень, які з'явилися в пізніших версіях, таких як Java 11, Java 12 тощо;

– обмежені можливості використання лямбда-виразів, їх можливості обмежені порівняно з пізнішими версіями мови. Наприклад, в Java 8 відсутні оператори зіставлення шаблонів із значеннями тощо;

– відсутність модульності. Вона дозволяє розбивати програми на модулі з явними залежностями і ізольованою видимістю, що полегшує розробку та підтримку складних застосунків.

Ці недоліки використання конкретної версії мови не впливають на розробляємий застосунок, тому не є критичними.

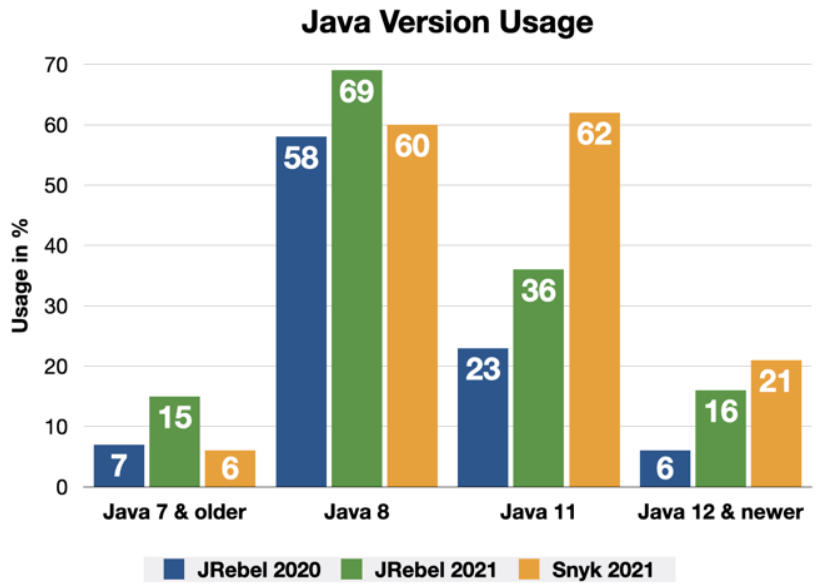


Рисунок 3.5 – Графіки популярності різних версій Java

3.2 Проєктування архітектури системи

Будучі клієнт-серверною, система повинна виконувати базові принципи CRUD методології. CRUD – це абревіатура від англійської фрази Create, Read, Update, Delete, що описує чотири основні операції, які можуть бути виконані над даними у системі:

– Create (створення) – створення нового запису в базі даних;
– Read (читання) – отримання інформації з бази даних;
– Update (оновлення) – зміна інформації, що зберігається в базі даних;
– Delete (видалення) – видалення інформації з бази даних.

Такі операції можуть бути використані для будь-яких типів даних, таких як тексти, зображення, відео та інші. CRUD є стандартом для взаємодії між

застосунками та базами даних і є основою для багатьох вебфреймворків та інших інструментів розробки програмного забезпечення.

У RESTful, HTTP методи використовуються наступним чином для виконання CRUD операцій:

– Create (створення) – використовується метод POST. Клієнт відправляє запит на створення нового ресурсу на сервері;

– Read (читання) – використовується метод GET. Клієнт відправляє запит на отримання ресурсу з сервера;

– Update (оновлення) – використовується метод PUT або PATCH. Клієнт відправляє запит на оновлення існуючого ресурсу на сервері;

– Delete (видалення) – використовується метод DELETE. Клієнт відправляє запит на видалення існуючого ресурсу на сервері.

Операції CRUD можуть бути виконані на будь-якому ресурсі за допомогою відповідних HTTP методів.

Цей підхід до виконання CRUD операцій забезпечує більш зрозумілу та стандартизовану взаємодію між клієнтом та сервером, що дозволяє розробникам створювати більш масштабовані та стійкі системи.

Відповідно з боку проєкту треба реалізувати GET метод, який буде показувати мобільному застосунку, що серверний компонент запущено, він працює справно і готовий обробляти запити. Та POST метод, який буде надсилати дані до серверу і отримувати відповідь.

3.2.1 Проєктування загальної архітектури

Загальна архітектура представлена трьома сутностями: клієнт, серверна частина та фасад серверу. Така архітектура дозволить побудувати гнучку систему, яка не вимагає від користувача надпотужностей для підрахунку натовпу, але надає можливість підрахунку кількості людей на зображенні в

реальному часі. Відповідно кожен компонент системи покликаний, щоб покрити певну частину вимог до системи, а саме:

– клієнтський застосунок виступає в ролі інтерфейсу користувача. Він надає лаконічний і простий інтерфейс і виконує такі функціональні вимоги: отримання зображення з камери смартфону, попередня обробляє їх, кодує і передає та отримує від серверу;

– серверний компонент реалізує наступну логіку: отримання зображення, декодування, класифікація, співставлення до одного з трьох класів, передача до відповідного модуля задля підрахунку та естімації кількості людей, передача проміжних даних до серверного фасаду та відправка до фасаду та застосунку;

– серверний фасад виступає додатковим інструментом у системи і дозволяє аналізувати те, як система опрацьовує данні. Він покликаний отримувати данні з серверного компоненту та відображати їх для швидкого аналізу (перевірка чи правильно система підраховує натовп і генерує теплову мапу, чи правильно знаходить обличчя) [23].

Задля того щоб передавати зображення безпечно треба створити певний алгоритм, який дозволить кодувати його і передавати до серверної частини. В цій роботі було використано метод кодування Base64. Після обробки зображення кодується до рядка, передається до серверу через RESTFul і потім декодується на боці серверу (рис. 3.6).

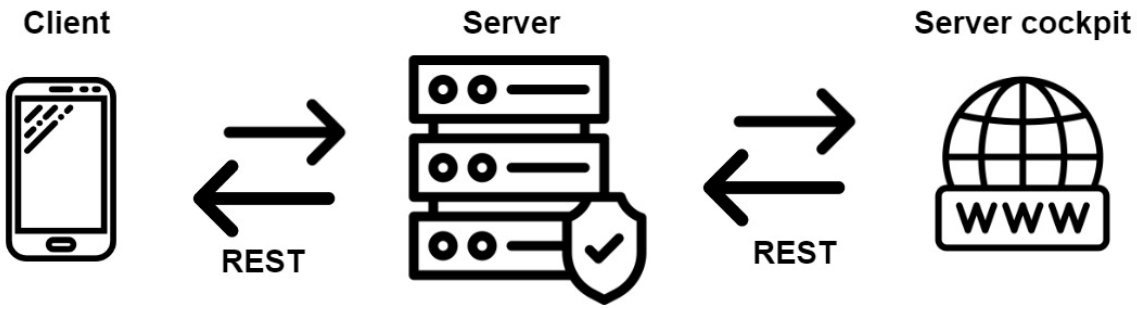


Рисунок 3.6 – Архітектура системи

3.2.2 Покриття проєктом вимог до клієнт-серверних систем

Для клієнт-серверних систем існують і інші вимоги. Вони дозволяють повною мірою показати слабкі та сильні сторони спроєктованої системи.

Функціональність: клієнт-серверні проєкти повинні забезпечувати певну функціональність, яку вимагають відповідні застосунки. Наприклад, якщо це мережевий чат, то проєкт має дозволяти користувачам обмінюватися повідомленнями – в цьому випадку сервер реалізує підрахунок осіб у натовпі, а клієнт віддає та отримує дані.

Клієнт-серверні проєкти повинні бути надійними. Це означає, що вони мають працювати безперебійно в будь-який час і за будь-яких обставин.

Дійсно, мобільний застосунок сам по собі не виконує ніякої логіки, окрім збору та відображення даних з серверу і він не може покривати поставленні вимоги без серверу.

Клієнт-серверні проєкти повинні бути ефективними з точки зору продуктивності. Це означає, що вони мають працювати швидко і витрачати мінімум ресурсів: в цій системі сервер покликаний виконувати лише необхідні розрахунки, тому він не витрачає ресурси без вагомих причин.

Клієнт-серверні проєкти повинні бути масштабовані. Це означає, що вони мають працювати стабільно при великому числі користувачів і при великому обсязі даних. Як бачимо, для підтримки цього пункту необхідно слідкувати за кількістю користувачів та тим, чи система встигає оброблювати їх запити.

У випадку, якщо система не може покривати вимоги користувачів, треба підключити балансер, тобто певну маршрутизацію. В такому випадку створюється декілька серверних компонентів і балансер вирішує куди направити вхідні запити, щоб збалансувати навантаження. Тому при певному вдосконаленні ця система цілком масштабована.

Клієнт-серверні проєкти повинні бути безпечними. Це означає, що вони мають захищати дані користувачів від несанкціонованого доступу і зберігати

їх у безпеці. Вразливою ланкою системи є момент передачі запиту, який теоретично можуть перехопити шахраї. Задля мінімального захисту використовується кодування у Base64-рядок.

Цей момент імплементації за наявності ресурсів треба переробити з використанням більш сучасних алгоритмів шифрування. Але для поточних потреб цього цілком достатньо.

Сумісність: Клієнт-серверні проєкти повинні бути сумісними з різними пристроями. На цих теренах Android має чи не найбільшу сумісність між версіями та пристроями.

3.2.3 Проєктування серверної частини

Серверний компонент складається з декількох модулів. Він інкапсулює в собі наступні компоненти (рис. 3.7):

– HTTP Server – який запроваджує вебінтерфейс для реалізації передачі даних по RESTFul методології. Завдяки цьому компоненту сервер буде приймати та віддавати данні, та оновлювати їх на серверному фасаді;

– Visdom інтеграція – це ще один додатковий вебінтерфейс який дозволяє виводити проміжні результати обрахувань такі як теплові мапи та анотовані обличчя;

– блок з навченими модулями, для виконання задач класифікації та підрахунку [24 – 27].

Такий перелік модулів дозволить покрити поставлені вимоги, та розбити модель на менші модулі, які простіше розробляти, тестувати та підтримувати.

Більше того, подібний підхід дозволяє обмежити кількість використаного коду. Для розробок відведених для команд, багатомодульний підхід дозволяє виконувати розробку паралельно. Проекти написані таким способом є більш розширюваними, що в реаліях сучасного світу дозволяє швидко та гнучко вносити зміни та редагувати функціонал.

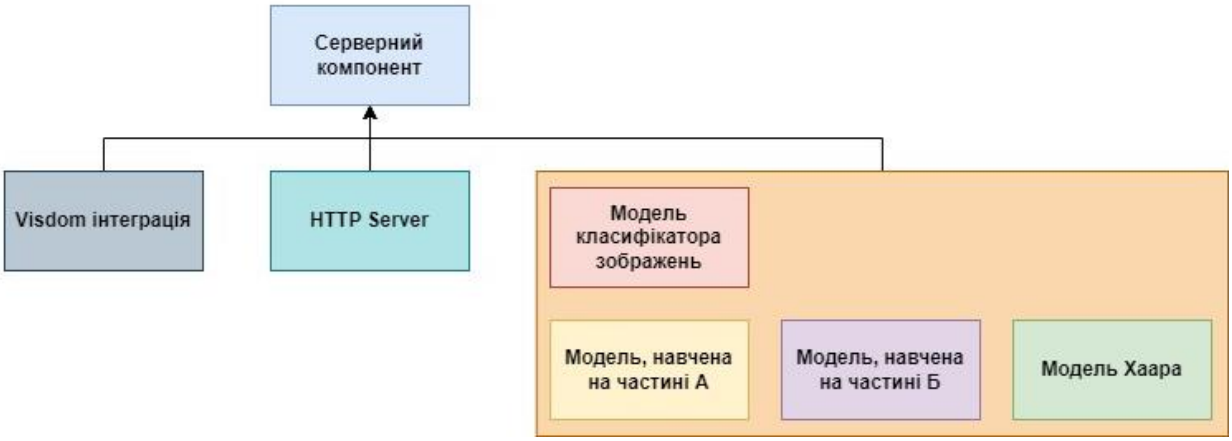


Рисунок 3.7 – Архітектура серверного компоненту

При проєктуванні до уваги бралася MVC архітектура, яка передбачає наявність трьох складових (рис. 3.8):

– модель (Model) надає данні і реагує на команди контролеру, змінюючи при цьому свій;

– представлення (View) відповідає за відображення даних моделі користувачу, оновлюючи їх за потреби, коли модель змінилась;

– контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність вносити зміни.

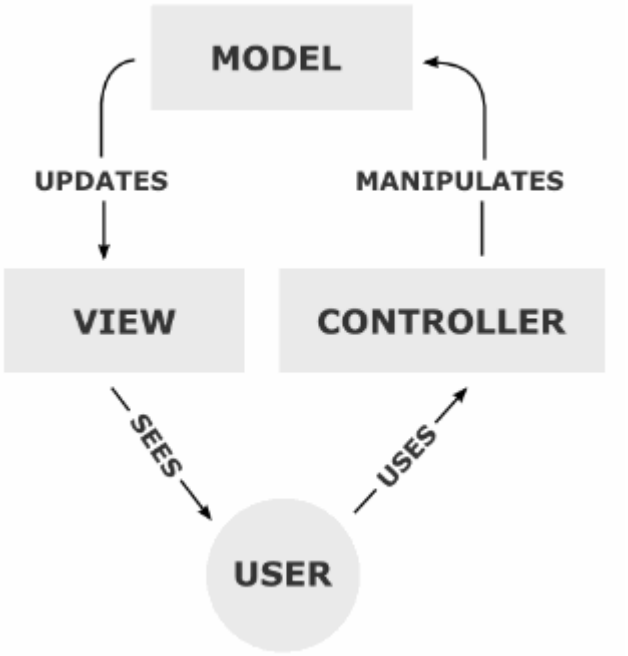


Рисунок 3.8 – Схема MVC архітектура

Такий підхід дозволяє відмежувати відповідальність між модулями та чітко організувати її. Таким чином Controller реалізовано через обгортку HTTP Server, який ініціалізує всі систему та надає канали доступу до неї.

Моделі підрахунку натовпу реалізовують Model. Інформація що надходить до серверного компоненту та повертається ним – це View моделі: відповідно це закодоване Base64 зображення, та наступна відповідь з кількістю людей на переданому зображенні.

3.2.4 Проєктування мобільної частини

Android застосунки складаються або наслідують активність – це базовий клас який дозволяє імплементувати потрібну логіку [28 – 31]. Активність є вхідною точкою застосунку, вона відповідає за ініціалізацію інших модулів та запуск застосунку. Повна архітектура зображена на рисунку 3.9.

MyCameraCaptureSession – це клас, який відповідає за те, щоб отримати зображення з камери та відобразити його на екрані.

AsyncJob – це відповідно, клас, який реалізує логіку застосунку. Він попередньо перевіряє чи працює сервер та коли процес підрахунку запущено, то підхоплює зображення з екрану, і кожний заданий інтервал часу надсилає його до серверного компоненту. Ця активність відбувається асинхронно в новому потоці, нащо вказує назва компонента.

Асинхронна природа цього методу дозволяє не зупиняти застосунок, поки він збирає зображення, кодує його, надсилає і очікує відповідь – навпаки той продовжує роботу, а інтерфейс оновлюється лише коли відповідь надана.

RestService – це реалізація вебінтерфейсу, яка написана на бібліотеці OkHttp3. Описані модулі дозволяють виконувати свою задачу ефективно при цьому зберігаючи доволі просту структуру без зайвих ускладнень.

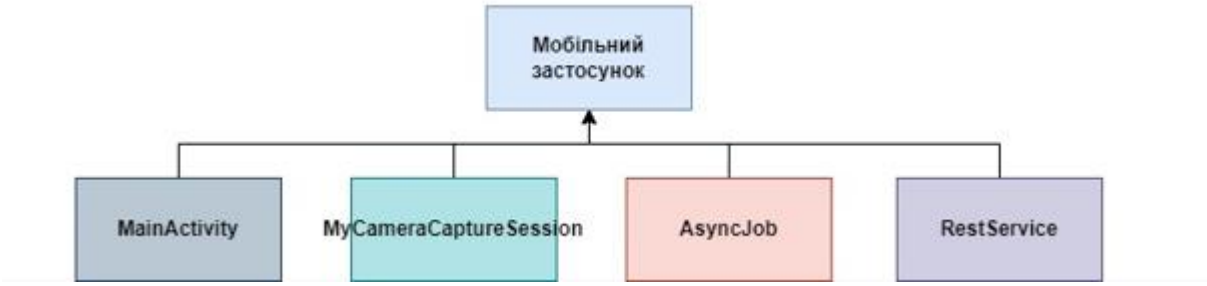


Рисунок 3.9 – Архітектура мобільного застосунку

Інтерфейс застосунку передбачає всього декілька дій, які може зробити користувач: надання доступу при першому запукску, запуск фільмування екрану, початок підрахунку людей [30 – 33]. Для виконання цих вимог спроєктовано користувацький інтерфейс, приклад якого зображено на рисунку 3.10.

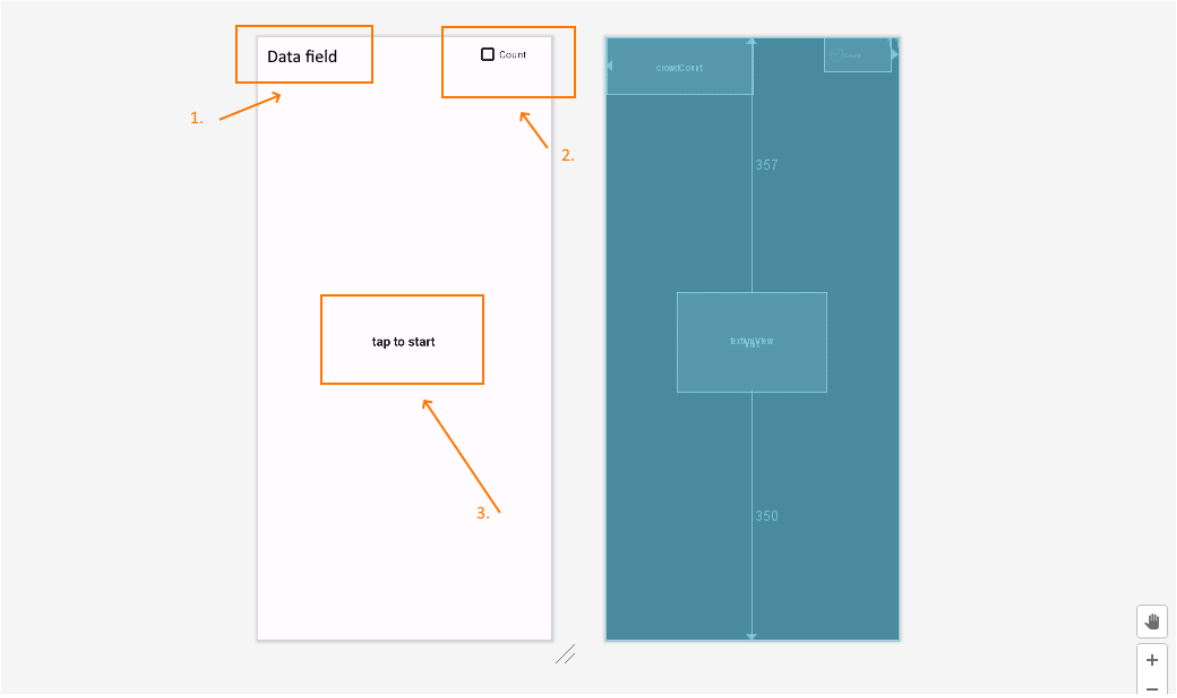


Рисунок 3.10 – Макет мобільного застосунку

Ділянка, нумерована одиницею, покликана відображати поточний метод підрахунку та кількість підрахованих осіб. Прапорець анотований двійкою покликаний на те, щоб показувати чи запущено процес підрахунку зараз чи ні.

Область під цифрою три в центрі екрану запускає процес фільмування та виводу зображення з камери.

Дозволи в Android – це механізм, який дозволяє визначати, які операції можуть виконуватися в застосунку [34, 35]. Цей механізм забезпечує безпеку та конфіденційність даних, які обробляються застосунком.

В Android, дозволи визначаються у вигляді XML-файлів, що містять список операцій, які застосунок може виконувати. Ці операції можуть включати доступ до камери, мікрофону, зображень, контактів, GPS та інших системних ресурсів.

Кожен дозвіл має свій унікальний ідентифікатор, який використовується для запиту дозволу від користувача. Користувач може відхилити запит на доступ до системних ресурсів, і в такому випадку застосунок не отримає доступ до цих ресурсів.

Починаючи з Android 6.0 (Marshmallow), Google змінила механізм дозволів, додавши можливість запитувати дозвіл під час виконання застосунку, а не тільки під час інсталяції. Це означає, що користувач може надати або відхилити дозвіл на доступ до системних ресурсів в будь-який момент [36, 37].

Для конкретної системи необхідно мати дозволи для використання камери та до комунікації через REST, усі дозволи показано на рисунку 3.11.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<!-- For REST part   -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-feature android:name="android.hardware.camera" />
```

Рисунок 3.11 – Дозволи, необхідні дял роботи застосунку

3.3 Ілюстрація роботи застосунку

Перш ніж користуватися застосунком, треба запевнитися, що серверний компонент і серверний фасад працюють справно. Для цього достатньо перевірити два порти на яких працюють сервер та Visdom фасад, відповідно порт 7001 та 8097. У разі роботи, обидва сервіси для GET запиту повернуть 200 статус з кодом OK.

Цю перевірку можна зробити будь-яким зручним способом, наприклад, через Postman. Для цього на машині, на які знаходиться сервер, треба виконати два GET запити (рис. 3.12, 3.13).

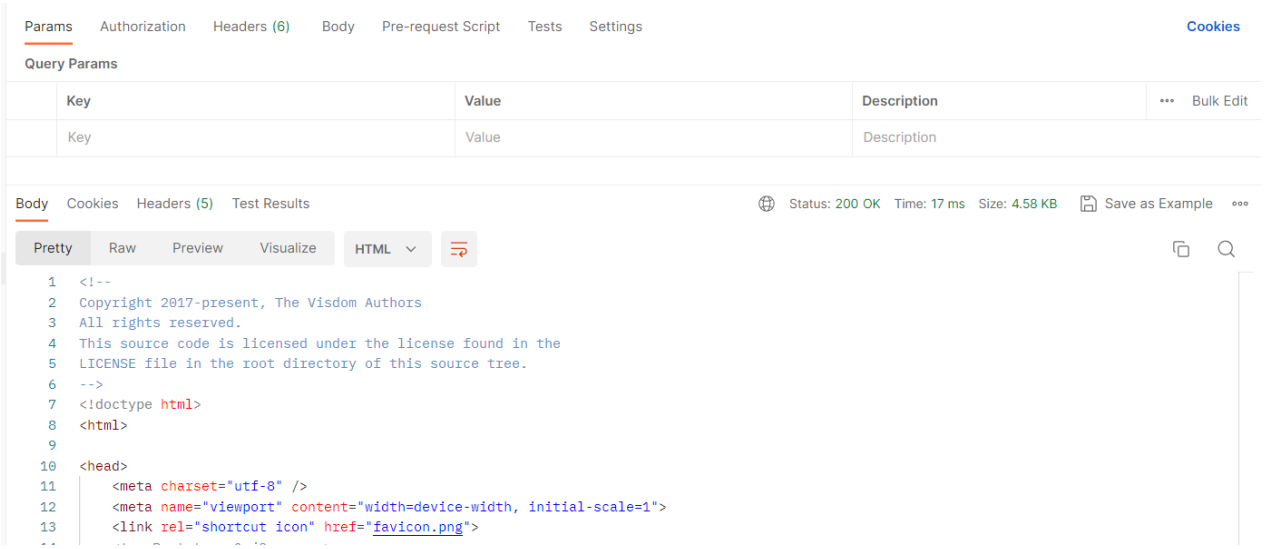


Рисунок 3.12 – Відповідь від серверного фасаду

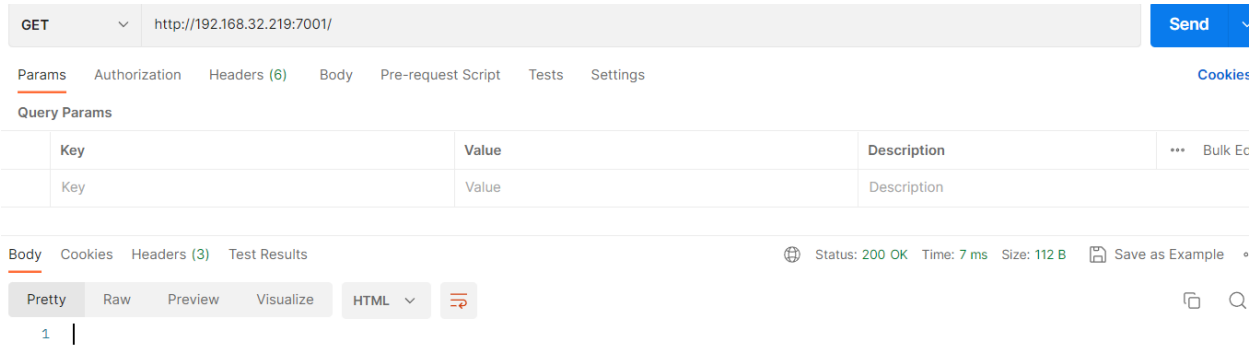


Рисунок 3.13 – Відповідь від серверного компоненту

Після цього можна перейти до клієнтської частини. Необхідно встановити застосунок на будь-який Android смартфон, який підтримує щонайменше Android 11, для смартфонів версією нижче підтримки немає.

Користувач запускає застосунок, де після завантаження треба натиснути на екран (рис. 3.14).

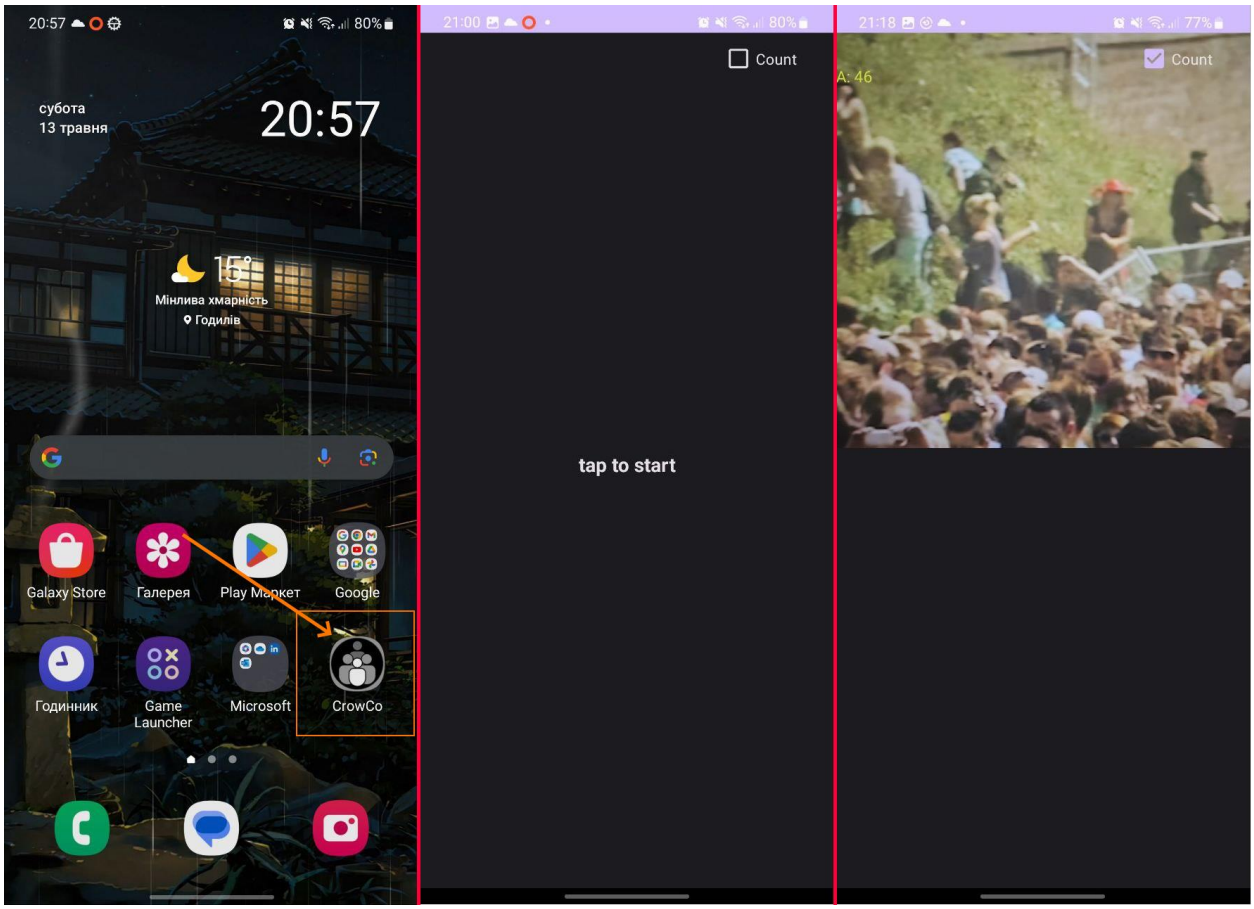


Рисунок 3.14 – Клієнтський застосунок

Після чого увімкнеться фільмування та проєкція камери на екран, як показано у третій секції рисунку. Якщо сервер працює справно, то користувач побачить додаткове повідомлення про це в секції з виводом інформації (рис. 3.15).

Це поле використовується для інформування користувача: туди виводяться повідомлення щодо серверу та зображення, щодо якого виконується підрахунок. Якщо зображення оброблялося певним методом, то це буде відображено в області для повідомлення.

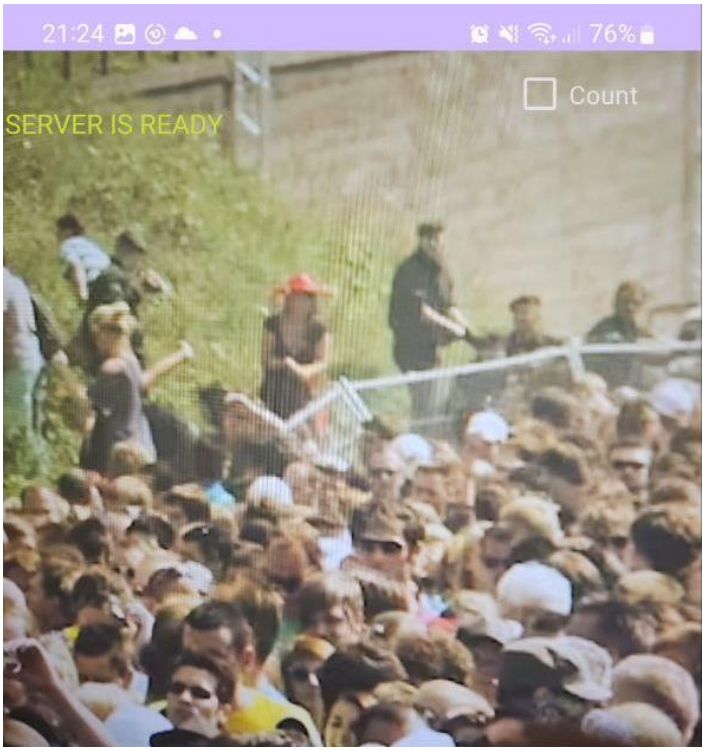


Рисунок 3.15 – Нотифікація про успішне з’єднання з сервером

Після активації прапорця підрахунку людей, застосунок починає асинхронно посилати запити до серверу. За замовчуванням це відбувається кожні 5 секунд, але цей часовий проміжок може бути змінений. Кожне звернення маркується на боці серверу, що дозволяє займатися відладкою системи (рис. 3.16).

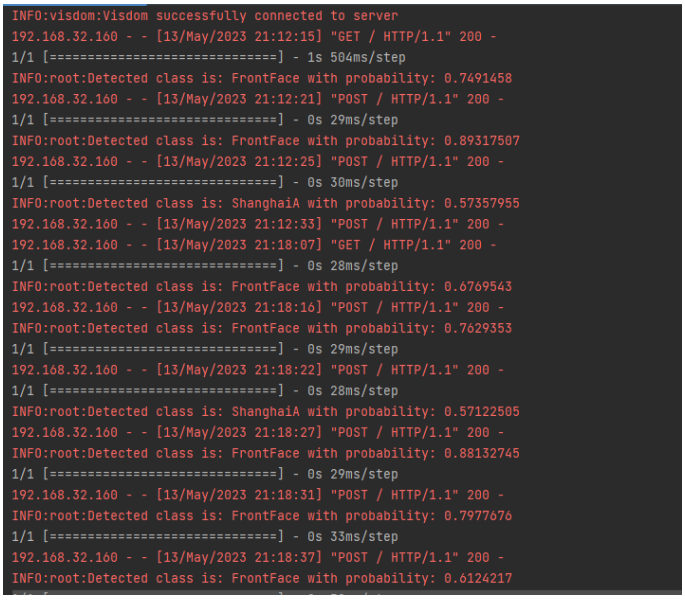


Рисунок 3.16 – Нотифікація надходження запитів

Кожен успішний запит отримує відповідь від серверу і тоді поле для інформації оновлюється.

Якщо користувач має намір проаналізувати проміжні данні, то йому треба зупинити підрахунок осіб і перейти на серверний фасад Visdom.

Щоб користуватися серверним фасадом Visdom, треба звернутися за його портом за замовчуванням – це 8097. Поки жоден клієнт не надіслав запит на обробку, вікно фасаду буде залишатися пустим.

Як тільки користувач запустить процес обробки зображень, проміжні дані почнуть оновлюватися в реальному часі.

Коли серверний компонент оброблює зображення, він звертається до фасаду і офовлює вікна у ньому. Нижче приведені приклади того як це відбуваєистя для зображень різних класів.

Цей серверний фасад представляє великий інтерес, бо через нього потенційно можна виводити дані у майже будь-якій формі і форматі (рис. 3.17).

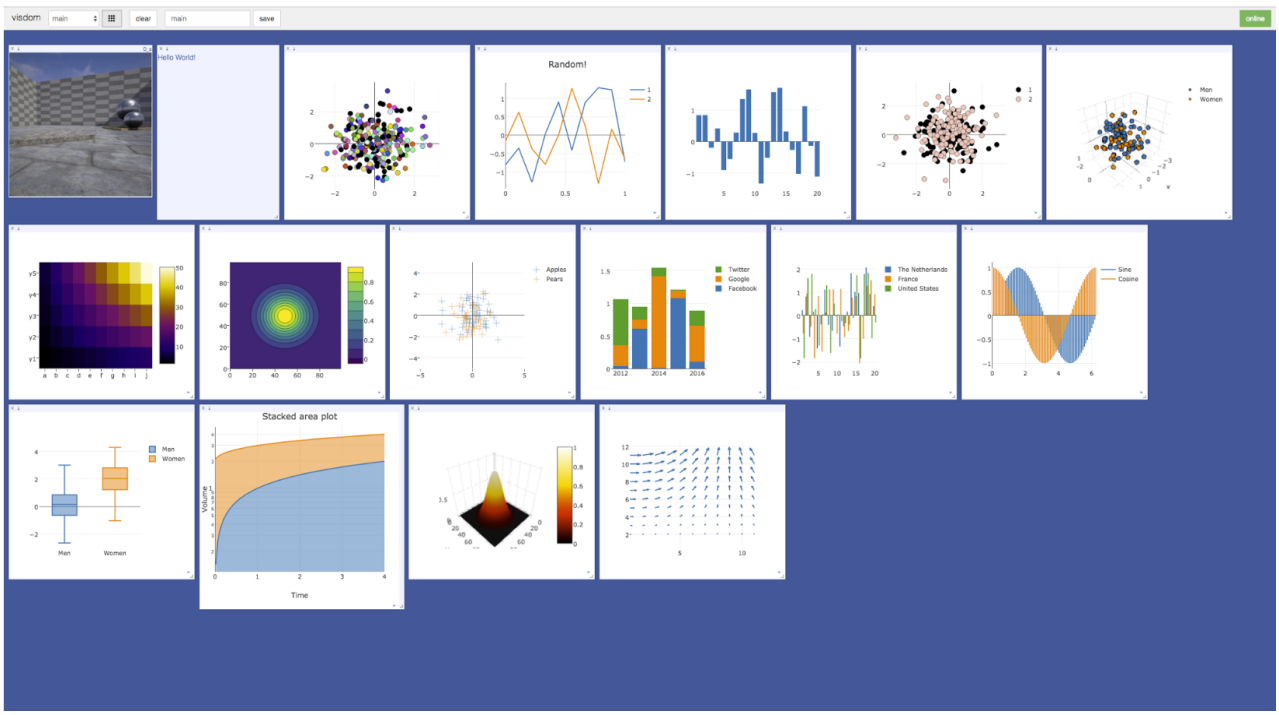


Рисунок 3.17 – Графіки серверного компоненту

Таким чином у майбутньому є можливість реалізувати додаткові графіки статистики, наприклад: скільки зображень було віднесено до якого класу, з якою швидкістю обробляються зображення і таке інше [26]. Тобто цей фасад можна використовувати для аналізу, збору статистики та оцінки продуктивності серверного компоненту.

Коли система обробила декілька зображень і вікно фасаду вже заповнилося проміжними даними, тоді користувач може побачити монохромне зображення оригіналу та мапу густини цього зображення. Приклад того, як це виглядає для зображень типу густого натовпу наведено на рисунку 3.18.

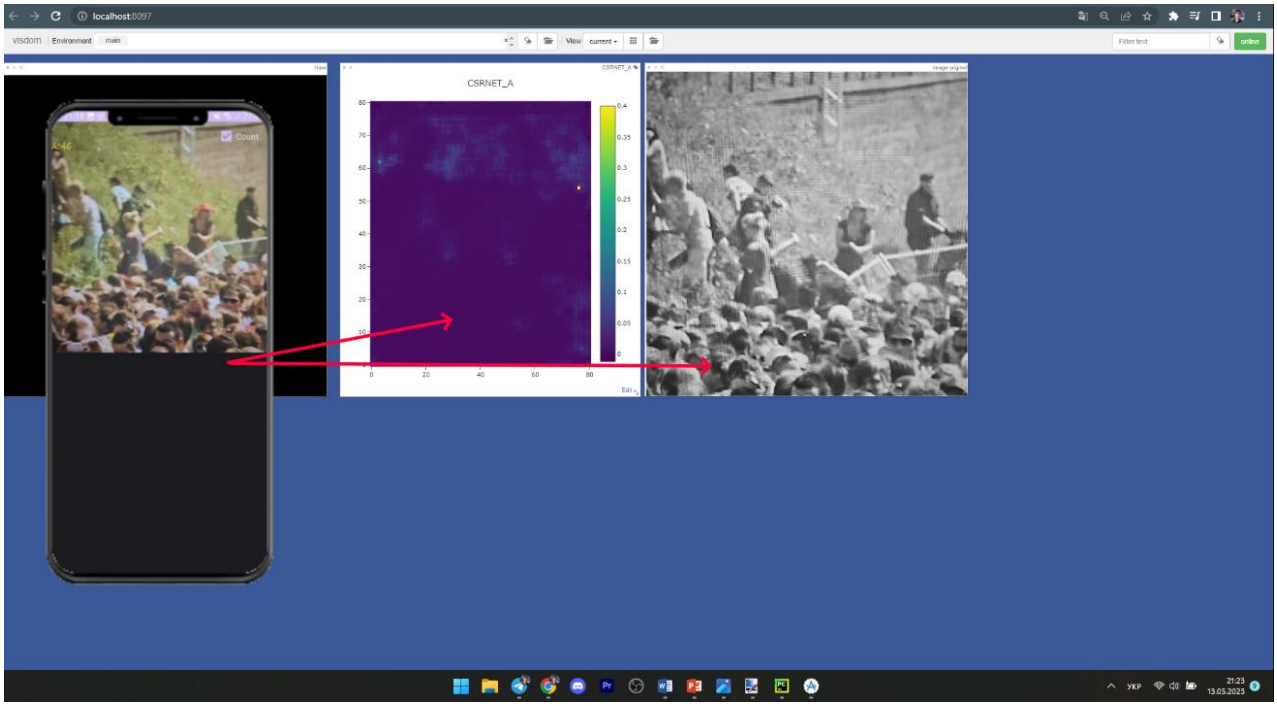


Рисунок 3.18 – Підрахунок темлової мапи для натовпу

Застосунок підраховує не лише густий натовп але і групові фото з фронтальними обличами, які відносятся до окремого класу. Цей функціонал можна протестувати змінивши камеру на фронтальну. Якщо на зображенні що фільмується, будуть обличча крупним планом, тоді класифікатор віднесе його до зображення, на якому треба шукати обличча і обробить зображення методом Хаара. В такому випадку проміжна інформація в серверному

фасаді – це вже не мапи густини, а зображення з анотаційними рамками (рис. 3.19).

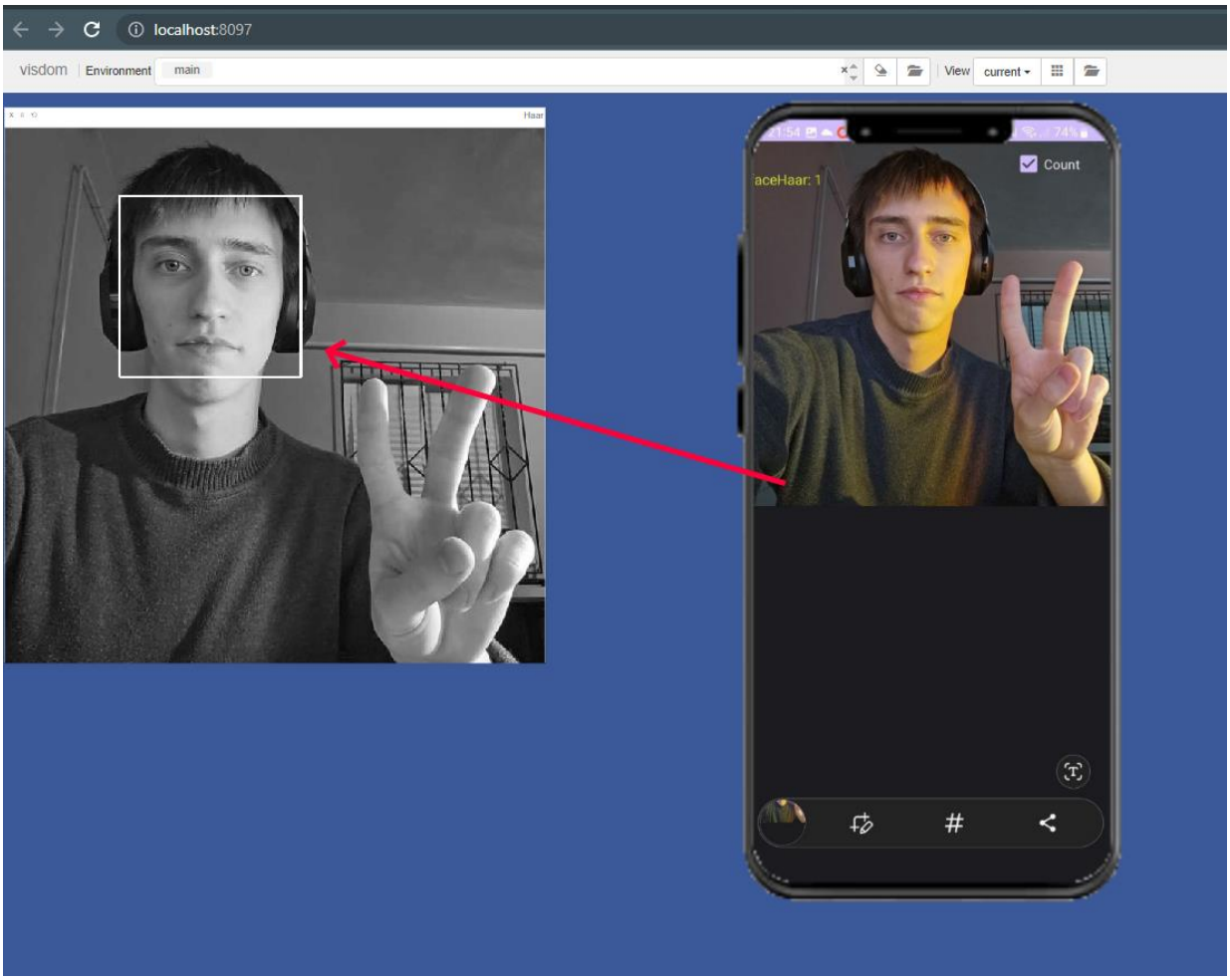


Рисунок 3.19 – Детекція фронтального обличчя

За бажанням користувача роботу підрахунку можна зупинити в будь-який момент і тоді клієнт зупинить зв’язок з серверним компонентом і перестане підраховувати натовп. Більше зображень працюючого застосунку можна знайти в додатку Д.

ВИСНОВКИ

У рамках кваліфікаційної роботи була розроблена і реалізована система підрахунку кількості людей у натовпі основана на згорткових нейронних мережах. Особливу увагу було приділено точності та швидкості передачі та обробки фотозображень й відео. Для виконання цієї задачі було спроєктовано клієнт-серверну систему в межах якої були використані наступні методи та технології:

− опрацьовані основні принципи згорткових нейронних мереж, особливості підготовки даних для них, їх навчання та перенавчання;

– вивчено метрики для оцінки якості навчання нейронних мереж;

– проведено огляд та аналіз існуючих методів підрахунку людей у натовпі, та обрано найбільш сучасну та оптимальну архітектуру CSRNet;

– було проведено процес навчання мережі на датасетах, що містять натовп;

– вивчено відповідні бібліотеки PyTorch та OpenCV, Tensorflow для обробки та аналізу фотозображень та відео;

– спроєктовано структуру застосунку;

– вивчено середовище AndroidStudio для розробки клієнтської частини, та PyCharm для серверної;

– розроблено клієнтську та серверну частини застосунку.

Проведені дослідження нейронних мереж MCNN, CSRNET показали, що мережа CSRNET має більш високу точність підрахунку людей, для неї показник MAE на датасеті ShanghaiTech дорівнює 68,2 та 10,6 відповідно для частини A та частини B. В той час як для MCNN показник MAE – 110,2 та 26,4 для частин A та B відповідно.

Однак мережа MCNN має кращу швидкість: середній час обробки одного зображення склав 2,3 мс, коли як для CSRNET середній час дорівнював 3,5 мс. В застосунку було реалізовано клієнт-серверну архітектуру, де усі

розрахунки відбуватися на сервері, тому несуттєва перевага в швидкодії буде не так важлива ніж точність підрахунку. Тому незважаючи на незначну перевагу у швидкодії мережі MCNN, було вирішено в роботі для підрахунку людей на основі теплових мап використати більш точну мережу CSRNET.

Для питання детектування облич у випадку групових зображень з малою щільністю було розглянуто мережу DSDF та метод на основі каскадів Haar. Детектор DSDF показав себе як дуже точний, але повільний – його швидкість нижче методу на основі каскадів Haar в 14,34 разів. Тому в роботі для підрахунку людей на основі детектування облич було вирішено використати метод на основі каскадів Haar.

У подальшому розвитку роботи необхідно розглянути більш сучасні методи детекції облич, які б могли детектувати не лише фронтальні зображення облич, а ті що знаходяться під кутом або частково закриті іншими об’єктами.

Результати роботи апробовано у вигляді 2 тез доповідей під час 27-го Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ В XXI СТОЛІТТІ» [18] та 14-ої Міжнародної науково-практичної конференції «FREE AND OPEN SOURCE SOFTWARE» [14].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. *Харків: ХНУРЕ*.

2. Гороховатський, В.О., & Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

3. Кобилін, О.А., & Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ.

4. Yakovleva, O., & Nikolaieva K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

5. SytossResearch. (n.d.). SytossResearch/descriptorbasednormalization. GitHub. (2023) https://github.com/SytossResearch/DescriptorBasedNormalization

6. Gorokhovatskyi, V., & Tvoroshenko I. (2020). Image classification based on the Kohonen network and the data space modification.

7. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

8. Ковтуненко, А.Р., Яковлева, О.В., Любченко, В.А., & Янголенко, О.В. (2020). Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників.

9. Cherednichenko, O., Kanishcheva, O., Yakovleva, O., & Arkatov, D. (2020). Collection and Processing of a Medical Corpus in Ukrainian. In COLINS (pp. 272-282).

10. Cherednichenko, O., Vovk, M., Yanholenko, O., & Yakovleva, O. (2020). Towards the technology of employers' requirements collection development. In Integrated Computer Technologies in Mechanical Engineering: Synergetic Engineering (pp. 228-239). Cham: Springer International Publishing.

11. Cherednichenko, O., Yanholenko, O., Iakovleva, O., & Kustov, O. (2014). Models of research activity measurement: web-based monitoring implementation. In Information Systems: Education, Applications, Research: 7th SIGSAND/PLAIS EuroSymposium 2014, Gdańsk, Poland, September 25, 2014. Proceedings 7 (pp. 75-87). Springer International Publishing.

12. Cherednichenko, O., Yanholenko, O., Liutenko, I., & Iakovleva, O. (2013). Monitoring and Evaluation Problems in Higer Education-Comprehensive Assessment Framework Development. In CSEDU (pp. 455-460).

13. Yanholenko, O., Cherednichenko, O., Yakovleva, O., & Arkatov, D. (2020). A Model for Estimating the Security Level of Mobile Applications: a Fuzzy Logic Approach. In IntelITSIS (pp. 252-266).

14. Гречишкін, Д. С. (2023). ОГЛЯД ДАТАСЕТІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ПІДРАХУНКУ ЛЮДЕЙ. Матеріали XIV-ої Міжнародної науково-практичної конференції «Free and Open Source Software», Харків, 14-16 лютого 2023 р.–Харків: Харківський національний економічний університет імені Семена Кузнеця, 110 с., 53.

15. Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 589-597).

16. Li, Y., Zhang, X., & Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1091-1100).

17. Kobylin, O., & Lyashenko, V. (2020). Time series clustering based on the k-means algorithm.

18. Гречишкін Д.С., Яковлева О.В. (2023), Аналіз у порівняльному аспекті моделей MCNN та CSRNet для вирішення задачі підрахунку людей у натовпі. 27-ий міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ», С. 48-49.

19. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In Explainable Deep Learning AI (pp. 125-146). Academic Press.

20. V. Gorokhovatskyi, I. Tvoroshenko (2020). Image Classification Based on the Kohonen Network and the Data Space Modification, in: CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020), pp. 1013–1026.

21. Y. Daradkeh, V. Gorokhovatskyi, I. Tvoroshenko, S. Gadetska, M. Al-Dhaifallah, (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components, IEEE Access 9 92964–92973.

22. P. Viola, M. Jones, (2001). Rapid Object Detection Using a Boosted Cascade of Simple Features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001).

23. N. Dalal, B. Triggs, (2005). Histograms of Oriented Gradients for Human Detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05).

24. H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, (2015). A convolutional neural network Cascade for face detection, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5325–5334.

25. R. Girshick, J. Donahue, T. Darrell, J. Malik, (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587.

26. Cherednichenko, O., Yanholenko, O., & Iakovleva, O. (2013). Web-Based monitoring and evaluation: research activity assessment case study (Doctoral dissertation, EDIS Publishing Institution of the University of Zilina).

27. S. Ren, K. He, R. Girshick, J. Sun, (2017). Faster R-CNN: Towards real-time object detection with region proposal networks, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1137–1149.

28. K. Zhang, Z. Zhang, Z. Li, Y. Qiao, (2016). Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Processing Letters 23(10) 1499–1503.

29. S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, S. Z. Li, (2017). FaceBoxes: A CPU real-time face detector with high accuracy, in: 2017 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–9.

30. S. Ren, K. He, R. Girshick, J. Sun, (2016). Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6) 1137–1149.

31. J. Deng, J. Guo, E. Ververas, I. Kotsia, S. Zafeiriou, (2020). Retinaface: Single-shot multi-level face localisation in the wild, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5203–5212.

32. E. Zhang, Y. Zhang, (2009). Average precision, Encyclopedia of Database Systems 192–193.

33. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System (COLINS-2023). In CEUR Workshop Proceedings (Vol. 3403). pp. 69-86.

34. Pretrained Pytorch Face Detection (MTCNN) and facial recognition (InceptionResnet) models. URL: https://github.com/timesler/facenet-pytorch (дата звернення 24.04.2023).

35. A high-performance pytorch implementation of face detection models, including RetinaFace and DSFD. URL: https://github.com/hukkelas/DSFD-Pytorch-Inference (дата звернення 23.04.2023).

36. Star-Clouds/Centerface: Face detection. URL: https://github.com/Star-Clouds/CenterFace (дата звернення 25.04.2023).

37. InsightFace: 2D and 3D Face Analysis Project. URL: https://github.com/deepinsight/insightface/ (дата звернення 27.04.2023).