Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформацій

Кафедра Радіотехнологій інформаційно-комунікаційних систем

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

ГЮІК.ХХХХХХ.000ПЗ
(позначення документа)

Сучасна соціальна мережа з використанням глибокого аналізу контенту і обробки даних за допомогою нейромереж
(тема)

Виконав:
студент II курсу, групи ІКТм -20-1
Гранкін О. О.
(прізвище, ініціали)

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

Тип програми
освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Інформаційно-комунікаційні технології
(повна назва освітньої програми)

Керівник доцент Бітченко О.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри РТІКС _____ Цопа О.І.
(підпис) (прізвище, ініціали)

2021 р.

Не містить відомостей заборонених для відкритого публікування.

Студент О. О. Гранкін

Керівник О.М. Бітченко

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Інформаційних радіотехнологій та технічного захисту інформацій
Кафедра Радіотехнологій інформаційно-комунікаційних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Тип програми Освітньо-професійна
Освітня програма Інформаційно-комунікаційні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові **ГРАНКІНУ Олександру Олександровичу**
(прізвище, ім'я, по батькові)

1. Тема роботи **СУЧАСНА СОЦІАЛЬНА МЕРЕЖА З ВИКОРИСТАННЯМ ГЛИБОКОГО АНАЛІЗУ КОНТЕНТУ І ОБРОБКИ ДАНИХ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖ**

затверджена наказом по університету від **5 листопада 2021 р. № 1648Ст**

2. Термін подання студентом проекту (роботи) **5 грудня 2021 р.**

3. Вихідні дані до проекту (роботи)
3.1 Розробити серверну частину з відкритим API інтерфейсом
3.2 Розробити мобільний додаток
3.3 Навчити нейронну мережу для класифікації об'єктів на зображенні

4. Перелік питань, що потрібно опрацювати в роботі
Вступ. 4.1 Огляд та аналіз соціальних та нейронних мереж. 4.2 Аналіз предметної області. 4.3 Аналіз і обґрунтування вибору технологій і мов програмування. 4.4 Дослідження використання нейронної мережі на мобільному і хмарному просторі. 4.5 Проектний розділ. 4.6 Розробка мобільного додатку Висновки. Перелік посилань. Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Комп'ютерна презентація

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
Основна частина	доц. Бітченко Олександр Миколайович		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	1.09-6.09	Виконано
2	Огляд та аналіз соціальних та нейронних мереж	11.09-20.09	Виконано
3	Аналіз предметної області	21.09-30.09	Виконано
4	Аналіз і обґрунтування вибору технологій і мов програмування	1.10-20.10	Виконано
5	Дослідження використання нейронної мережі на мобільному і хмарному просторі	21.10-1.11	Виконано
	Проектний розділ	2.11-10.11	Виконано
	Розробка мобільного додатку	12.11-16.11	Виконано
6	Реферат	17.11-19.11	Виконано
7	Перелік умовних позначень, символів, одиниць, скорочень і термінів	20.11-21.11	Виконано
8	Висновки	22.11-24.11	Виконано
9	Оформлення пояснювальної записки	25.11-30.11	Виконано
10	Оформлення презентації	1.12-11.12	Виконано
11	Подання роботи на кафедру	05.12.2021	Виконано

Дата видачі завдання **1 вересня 2021 р**

Студент _____
(підпис)

Керівник роботи_____ доц. Бітченко О.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить 97 сторінок тексту, 64 рисунки, 17 джерел посилання та 3 додатки.

СОЦІАЛЬНА МЕРЕЖА, АНАЛІЗ ЗОБРАЖЕНЬ, КЛАСИФІКАЦІЯ ОБ'ЄКТІВ, МОБІЛЬНИЙ ДОДАТОК

У першій частині пояснювальної записки були розглянуті принципи роботи нейронних мереж, проаналізовано різні алгоритми аналізу зображень, класифікації об'єктів, розглянуто готові моделі даних, проаналізовано існуючі соціальні мережі та їх використання нейронних мереж, розглянуто поточні технології розробки мобільних додатків. А також обґрунтовано вибір мови та технологій для розробки соціальної мережі з використанням нейромережі на операційній системі Android.

У другій частині пояснювальної записки було спроектовано структуру системи, інтегрована нейронна мережа з використанням алгоритму аналізу зображень для класифікації об'єктів YOLOv5, проведена настройка комунікації програми з хмарним сервісом зберігання даних Firebase Realtime Database, розроблений інтерфейс програми користувача. А також розроблено автоматичний процес збірки.

ABSTRACT

The explanatory note of the attestation work contains 97 pages of text, 64 figures, 17 literary sources and 3 applications.

SOCIAL NETWORK, IMAGE ANALYSIS, OBJECT CLASSIFICATION, MOBILE APP

In the first part of the explanatory note, the principles of operation of neural networks were considered, various algorithms for image analysis, object classification were analyzed, ready-made data models were considered, existing social networks and their use of neural networks were analyzed, and current technologies for developing mobile applications were considered. And also the choice of language and technologies for the development of a social network using a neural network on the Android operating system has been grounded.

In the second part of the explanatory note, the structure of the system was designed, the neural network was integrated using the image analysis algorithm to classify YOLOv5 objects, the communication of the application with the cloud storage service Firebase Realtime Database was configured, and the user interface of the program was developed. And also an automatic assembly process has been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

c – параметр, що визначає крутизну графіка порогової функції;

w – «вага» відповідного синапс;

x – вхідний сигнал;

C_m – параметр спонтанної активності нейрона;

ADT (Android Development Tools) – інструменти розробки Android;

BoF (bag of freebies) –

BoS (bag of specials) –

Clean Architecture – Чиста Архітектура ()

Dependency Rule – правила залежностей ().

IDE – інтегроване середовище розробки (

Lint – статичний аналізатор коду (),

MVVM (Model-View-ViewModel) –

RBF-мережі – радіально-базисні мережі;

SDK (Software Development Kit) –

YOLO (You Only Look Once) – сучасний детектор об'єктів;

БД – база даних

ВСТУП

Інтернет давно вже грає величезну роль в житті людини. Зараз складно уявити собі людину, яка не чула про такі поняття як «пошукова система», «сайт» або «веб-сторінка». Користувачів інтернету з кожним роком стає все більше, а значить, що і число пропонованих їм інтернет-послуг зростає. Один з найцікавіших елементів сучасного інтернету є соціальні мережі, які поєднують досить багато варіантів взаємодії користувача та інтернет площадки і мабуть, найбільш популярні серед користувачів.

Соціальні мережі були поширені ще до появи Інтернету. Соціальні мережі – це соціальна структуру, що складається з групи людей або організацій і зв'язків між ними, це не тільки сайти в інтернеті, а будь-які спільноти людей пов'язані між собою спільними інтересами. Якщо ми говоримо про соціальну мережу в інтернеті, то це – віртуальна мережа, яка є засобом надання послуг, пов'язаних з встановленням зв'язків між її користувачами, а також між різними користувачами і відповідними інформаційними ресурсами, розміщеними на веб-сайтах глобальної мережі. Простіше кажучи, це сайти з можливістю надати деяку інформацію про себе (школа, інститут, дата народження тощо), яку інші члени мережі можуть використовувати для пошуку. Зв'язок здійснюється через внутрішню поштову службу або систему миттєвих повідомлень.

Соціальні мережі стали головним дозвіллям користувачів в інтернеті. Вдалий формат заснований на профілях користувачів переконав інвесторів вкладати в ці сайти великі гроші, що дало величезний стрибок у розвитку даної технології. Соціальні мережі замінили користувачам сховища фотографій і свого часу витіснили з ринку багато сервісів для зберігання фотографій. Вони почали пропонувати користувачам необмежений віртуальний простір для знімків, контентом було легко ділитися, з'явилася можливість легко його редагувати. Оскільки фото подобалися користувачам більше, ніж текст, незабаром з'явилися окремі соціальні сервіси, побудовані виключно навколо зображень, наприклад, такі як Instagram.

Соціальні мережі зробили бізнес зручніше і ближче. Коли в соціальних мережах з'явилися мільйони користувачів, туди прийшли і бізнесмени - від власників скромних крамничок до транснаціональних корпорацій. Соціальні медіа дозволили бізнесу побудувати навколо свого продукту або послуг цілі співтовариства людей. Платформи підлаштувалися і стали пропонувати компаніям різні маркетингові інструменти.

В кінці 2000-х років соціальні мережі почали вводити функцію за допомогою якої можна висловлювати ставлення до постів і тим самим навчили користувачів оцінювати контент за допомогою лайків і дизлайків. Тоді стало остаточно ясно, що UGC, тобто контент, вироблений користувачами, - це головна сила, яка підтримує життя в соцмережах. Система оцінки постів, яка дозволяла залишити позитивний відгук, стимулювала аудиторію частіше писати і викладати фотографії.

На цьому розвиток системи оцінювання контенту в соціальних мережах зупинився в своєму розвитку. Користувач може зробити оцінку зі списку запропонованих реакцій на весь контент в цілому, але він не може оцінити конкретно окремий елемент. Наприклад, на фотографії зображено три машини, користувач не може віддати свою перевагу одній або двом конкретним машинам і дати оцінку тільки їм, він тільки може оцінити всю фотографію в цілому.

Метою даної кваліфікаційної роботи магістра є розробка соціальної мережі з використанням нейронних мереж для аналізу, визначення та класифікації контенту на основі якого буде повністю перероблена поточна система оцінювання контенту.

Для реалізації зазначеної мети необхідно вирішити такі завдання:

– навчити нейронну мережу для класифікації об'єктів на зображенні;

– розробити серверну частину з відкритим API інтерфейсом, для обробки запитів користувачів;

– розробити мобільний додаток з інтуїтивним інтерфейсом і використанням Material Design.

1 ОГЛЯД ТА АНАЛІЗ СОЦІАЛЬНИХ ТА НЕЙРОННИХ МЕРЕЖ

1.1 Нейронні мережі

1.1.1 Основні поняття та визначення

Нейронна мережа складається з штучних нейронів, кожен нейрон являє собою спрощену модель біологічного нейрона, які певним чином пов'язані один з одним і зовнішнім середовищем за допомогою зв'язків, кожна з яких має певний коефіцієнт, на який множиться та надходить через нього значення (ці коефіцієнти називають вагами).

Основна роботу яку виконує штучний нейрон – це приймання сигналів з багатьох входів, обробка їх єдиним чином і передача результату на багато інших штучних нейронів, тобто робить роботу аналогічну з нейроном біологічного виду (рисунок 1.1).

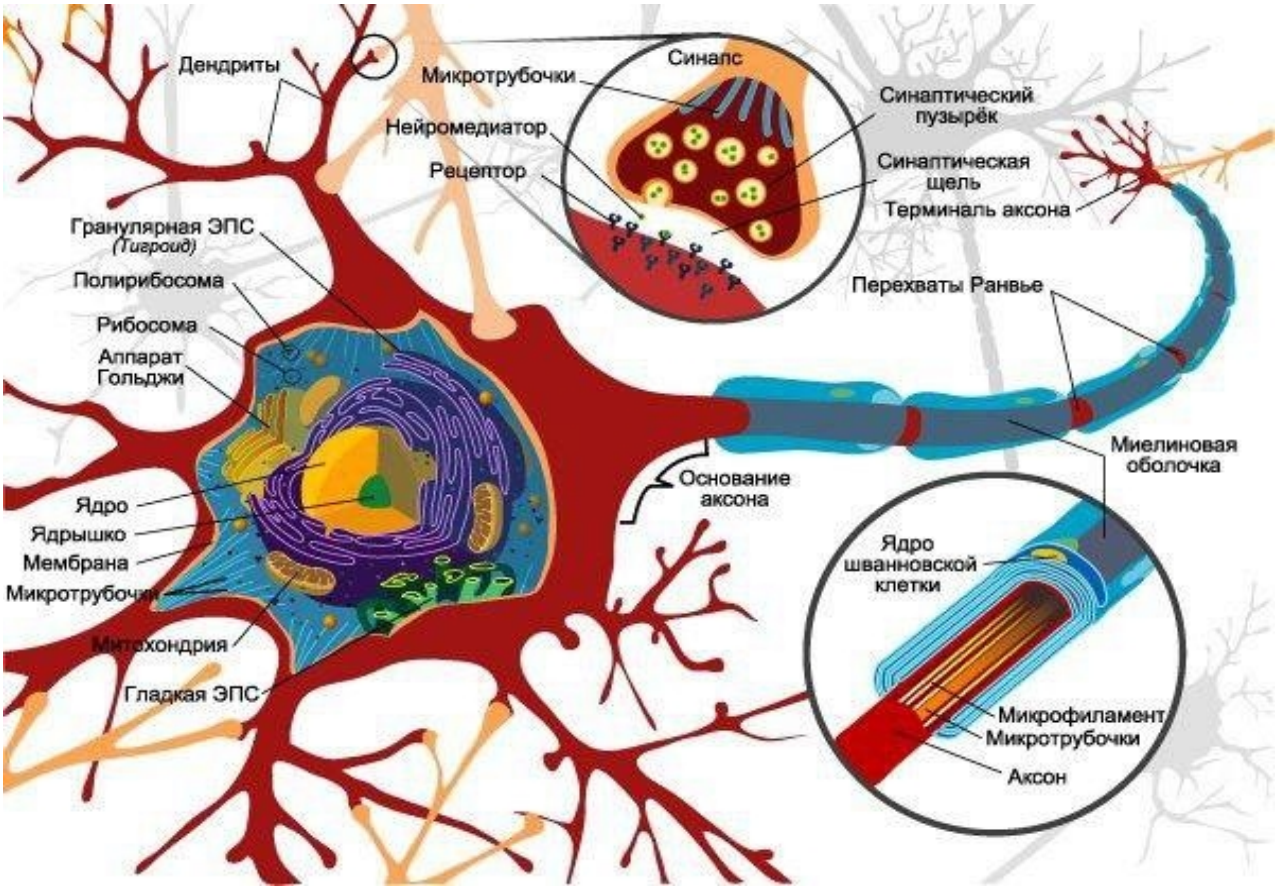

Рисунок 1.1 – Зображення біологічного нейрона

Біологічні нейрони пов'язані між собою аксонами, місця стиків називаються синапсами. У синапсах відбувається посилення або ослаблення електрохімічного сигналу. Зв'язки між штучними нейронами називаються синоптичними, або просто синапсами. У синапсу є один параметр – ваговий коефіцієнт, залежно від його значення відбуває ту чи іншу зміну інформації, коли вона передається від одного нейрона до іншого. Саме завдяки цьому вхідна інформація обробляється і перетворюється в результат, а навчання нейронної мережі засноване на експериментальному підборі такого вагового коефіцієнта для кожного синапсу, який і призводить до отримання необхідного результату [1].

Нейронні мережі можуть бути реалізовані апаратно, наприклад нейрочіпи та нейрокомп'ютери, і так само можуть бути реалізовані програмно. В процесі своєї життєдіяльності в нейронній мережі здійснюється перетворення даних, конкретний вид якого визначається вагами міжнейронних зв'язків, властивістю активаційної функції нейронів, конфігурацією мережі та архітектурою [2].

Найбільш часто нейронні мережі використовуються для вирішення складних завдань, які вимагають аналітичних обчислень подібних тим, які робить людський мозок. Найпоширенішими завданнями, для вирішення яких застосовуються нейронні мережі, це:

– прийняття рішень і управління. На виході мережі в результаті повинна з'явитися ознака рішення, яке вона прийняла. При цьому в якості вхідних сигналів використовуються різні критерії стану керованої системи [2].

– розпізнавання образів. У вигляді образів можуть використовуватися різні об'єкти: символи, зображення, зразки звуків, тексту тощо. У наш час це найбільш часто використовувана область застосування нейронних мереж. Для прикладу можна навести Google, коли користувач шукає фото, або коли камера смартфона визначає положення особи і виділяє його, і в багатьох інших додатках [2].

– стиснення даних і асоціативна пам'ять. Здатність нейромереж до виявлення взаємозв'язків між різними параметрами дає можливість представити дані більш компактно, якщо дані тісно пов'язані між собою. Зворотний процес - від-

новлення вихідного набору даних по частині інформації - називається (авто) асоціативної пам'яттю. Асоціативна пам'ять дає можливість також відновлювати вихідний сигнал / образ з зашумлених / пошкоджених вхідних даних [2].

– класифікація – розподіл даних за параметрами. Наприклад, на вхід нейромережі подається набір даних про людей і потрібно вирішити, кому можна давати кредит, а кому ні. Цю роботу може виконати нейронна мережа, аналізуючи таку інформацію, як платоспроможність, кредитна історія, вік тощо [2].

– кластеризація. Під кластеризацією мається на увазі розбиття множини вхідних сигналів на класи, при цьому ні ознаки, ні кількість класів заздалегідь не відомі. Після навчання нейронна мережа здатна визначати, до якого класу належить вхідний сигнал. Таким чином, подібна мережа здатна виявляти нові, невідомі раніше класи сигналів [2].

– прогнозування. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення і виділення прихованих залежностей між вхідними та вихідними даними. Після навчання нейронна мережа може передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень і (або) якихось існуючих зараз чинників [2].

Нейронні мережі також використовуються для вирішення оптимізаційних задач, аналізу даних, орієнтації в просторі, знаходження патернів у великих обсягах даних тощо [2].

1.1.2 Елементи та типи нейронних мереж

Нейронна мережа являє собою структуру взаємопов'язаних клітинних автоматів, що складається з наступних основних елементів:

– нейрон – елемент, що перетворює вхідний сигнал по функції

$$f(x) = \frac{x}{c + |x|} + C_m, \tag{1.1}$$

де x – вхідний сигнал;

c – параметр, що визначає крутизну графіка порогової функції;

C_m – параметр спонтанної активності нейрона.

– суматор – елемент, який здійснює підсумовування сигналів надходять на його вхід

$$f = \sum_{i=1}^{N} x_i \,. \tag{1.2}$$

– синапс – елемент, який здійснює лінійну передачу сигналу

$$f(x) = w \cdot x, \tag{1.3}$$

де w – «вага» відповідного синапсу.

Структура найпростішої нейронної мережі представлена на рисунок 1.2. До її складу входять нейрони вхідного, прихованого та вихідного шарів.

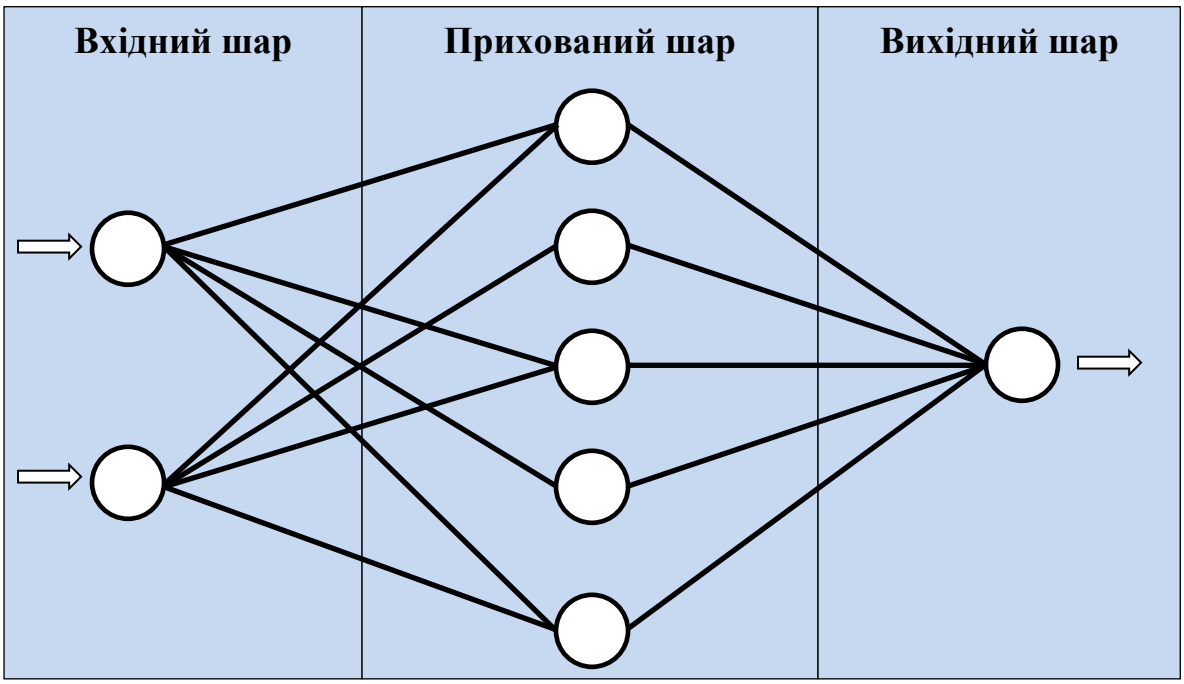

Рисунок 1.2 – Структура найпростішої нейромережі

Нейрони вхідного шару приймають дані ззовні (наприклад, від сенсорів системи розпізнавання осіб) і після їх обробки передають сигнали через синапси нейронів наступного шару. Нейрони другого шару (другий шар називають прихованим, так як він безпосередньо не пов'язаний ні з вхідним, ні з вихідним шарами нейромережі) обробляють отримані сигнали і передають їх нейронам вихідного шару. Оскільки мова йде про імітацію нейронів, то кожен процесор вхідного рівня пов'язаний з декількома процесорами прихованого рівня, кожен з

яких, в свою чергу, пов'язаний з декількома процесорами рівня вихідного. Така, найпростіша нейромережа здатна до навчання і може знаходити прості взаємозв'язки в даних [1].

Нейромережа здатна знаходити не тільки прості взаємозв'язки, а й взаємозв'язки між взаємозв'язками, маючи набагато складнішу структуру. У ній може бути кілька прихованих шарів нейронів, які виконують складні логічні перетворення. Кожен наступний шар мережі шукає взаємозв'язки в попередньому. Дані нейромережі здатні до глибокого (глибинного) навчання. Саме завдяки переходу на використання нейромережі з глибоким навчанням компанія Google змогла різко підвищити якість роботи свого популярного продукту "Перекладач" [1].

На початок 2019 р сайт вікіпедія налічує 26 типів нейромереж. 12 з них називаються на честь їх творців, у решти були такі назви як осциляторна, сіамська, хаотична, адаптивного резонансу тощо. Для того щоб якось систематизувати вже наявні і майбутні нейромережі, робляться спроби їх класифікації [1]:

– за типом вхідних даних: аналогові (на вході дійсні числа), виконавчі (на вході двійкові числа) і образні (на вході знаки, ієрогліфи, символи) нейронні мережі;

– за характером налаштування синапсів: мережі з фіксованими зв'язками (вагові коефіцієнти нейронної мережі вибираються відразу, виходячи з умов задачі), мережі з динамічними зв'язками (у цих мереж в процесі навчання відбувається настройка синаптичних зв'язків);

– за часом передачі сигналу: синхронні мережі (час передачі для кожної синаптичного зв'язку дорівнює або фіксованій постійній, або нулю), асинхронні мережі (час передачі для кожного зв'язку між елементами свій, але теж постійний);

– за характером навчання: навчання з учителем (вихідний простір рішень нейронної мережі відомо), навчання без вчителя (вихідний простір рішень формується тільки на основі вхідних впливів, а також вони мережі називають самоорганізації); навчання з підкріпленням (використовується система призначення штрафів і заохочень, що отримуються в результаті взаємодії ІНС з середовищем);

– за характером зв'язків: мережі прямого поширення (всі зв'язки направлені строго від вхідних нейронів до вихідних), рекурентні мережі (сигнал з вихідних нейронів або нейронів прихованого шару не повністю передається назад на входи нейронів вхідного шару).

Так само, використовуються радіально-базисні мережі (або RBF-мережі), самоорганізовані карти (зокрема, самоорганізована карта Кохонена) і мережі інших класів, ще не цілком сформовані [1].

1.1.3 Переваги та недоліки нейромереж

Основними перевагами нейронних мереж перед звичайними обчислювальними методами є:

– рішення задач в умовах невизначеності. За рахунок можливості навчання, нейронна мережа дозволяє вирішувати завдання з невідомими закономірностями і обчислювати залежності між вхідними та вихідними даними, це дозволяє працювати з неповними даними;

– гнучкість. Компоненти нейрокомп'ютерів (зв'язки між нейронами і самі нейрони) можна комбінувати різними способами. За допомогою цього один нейрокомп'ютер можна застосовувати для вирішення різних завдань, які ніяк не пов'язаних між собою;

– швидкодія. Нейронні мережі вирішують завдання швидше, ніж більшість інших алгоритмів, завдяки тому що вхідні дані обробляються багатьма нейронами одночасно;

– стійкість до шумів у вхідних даних. Відпадає необхідність в попередньому аналізі вхідних даних так як нейронна мережа може самостійно виявляти неінформативні для аналізу параметри і робити їх відсів;

– відмовостійкість. На несприятливу зміну умов нейромережа відповідає лише малим зниженням продуктивності. Тільки серйозні пошкодження структури можуть істотно вплинути на працездатність нейромережі, ця особливість пояснюється розподіленим характером зберігання інформації в нейромережі;

– адаптація. Нейронні мережі, навчаючись на даних, здатні підлаштовуватися під зміни навколишнього середовища (наприклад, під зміни ситуації на ринку, якщо завдання нейромережі – прогнозування коливань цін на біржі). Якщо необхідно вирішувати якесь завдання в умовах нестаціонарного середовища, то можуть бути створені нейромережі, які перенавчаються в режимі реального часу. Отже, чим вище адаптивні здібності системи, тим більш стійкою буде її робота в нестаціонарному середовищі.

До недоліків нейромереж можна віднести наступні:

– відповідь, що видається нейромережею, завжди приблизна. Завдання, в яких необхідно застосовувати нейромережу і одночасно отримувати точні відповіді, зустрічаються досить рідко;

– тривалість навчання. Для того щоб нейронна мережа могла коректно вирішувати поставлені завдання, потрібно навчити її на десятках мільйонів наборів вхідних даних;

– нездатність вирішувати обчислювальні завдання;

– нездатність прийняття рішень в кілька етапів. Нейронна мережа не може вирішувати завдання, які вимагають послідовного виконання декількох кроків; вона здатна вирішувати завдання тільки "в один захід". Тому нейромережа не може, наприклад, довести математичну теорему.

1.2 Огляд існуючих соціальних мереж з використанням нейронних мереж

1.2.1 Соціальна мережа LinkedIn

LinkedIn – це соціальна мережа, основним завданням якої є пошук нових ділових контактів. Тут працівники і роботодавці можуть знайти один одного, незалежно від того, в якій країні вони знаходяться.

Для початку користування соціальною мережею LinkedIn необхідно зареєструватися і авторизуватися. Авторизацію можна провести за коштами використанням Google аккаунта.

Основні види діяльності на яких базується соціальна мережа LinkedIn:

– ділове листування;
– рекрутинг;
– вивчення брендів;
– вивчення конкурентів.

Профілі в LinkedIn розроблені з урахуванням специфіки та ділової спрямованості. Вони включають такі пункти:

– світлина;
– навички та кваліфікація;
– контакти;
– місцезнаходження здобувача;
– попередні місця роботи;
– співтовариства;
– сфера діяльності;
– освіта.

У профілі є розділ "Загальні відомості". У ньому користувачі заповнюють важливі дані про спеціалізацію, додаткових якостях, загальну інформацію про себе тощо (рисунок 1.3).

Рисунок 1.3 – Профіль користувача LinkedIn

У LinkedIn для повноцінного використання соціальної мережі необхідно додати собі мінімум 50 контактів. Дану операцію можна виконати кількома способами. Перший – пошук по друзях і знайомих. Поле пошуку розташоване у верхній частині інтерфейсу (рисунок 1.4). Введіть ім'я людини і натисніть кнопку "Встановити контакт". Лінкедін пропонує вибрати параметри пошуку, що спрощує знаходження потрібних людей.

Після додавання 4-5 користувачів соціальна мережа використовуючи натреновану нейронну мережу підбере "можливих знайомих" серед них. Розширюючи цей список, ви побачите більше людей в списку. Також соціальна мережа аналізує дані про місця роботи, навчання, іншу інформацію, зазначену в профілі, і на підставі аналізу пропонує варіанти. Другий варіант додати контакти – імпорт. Функція дозволяє додати контакти з широкого кола, не шукати людей по одному. Імпортувати контакти можна з Outlook або електронної пошти. Опція розташована в верхньому меню, вкладці "Контакти".

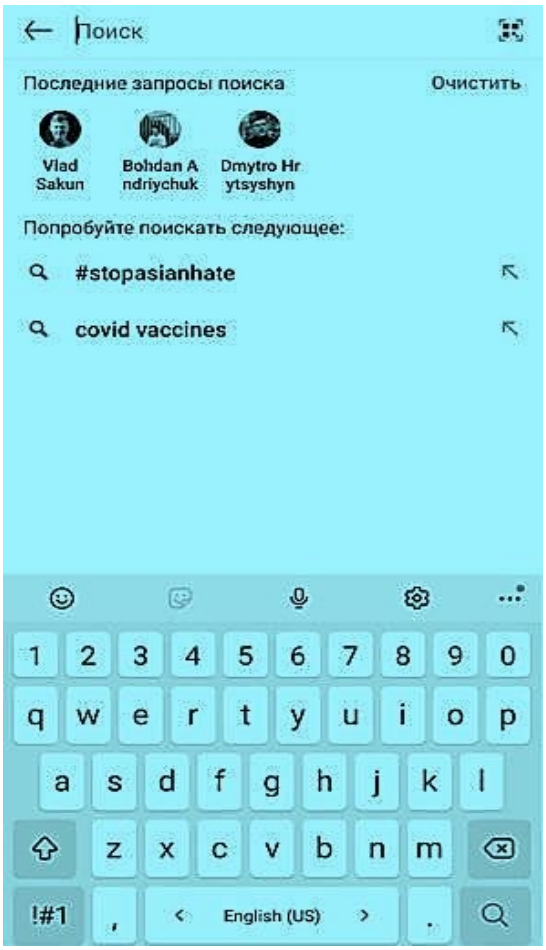

Рисунок 1.4 – Пошук контактів

При реєстрації персонального аккаунта соціальна мережа пропонує вказати не тільки адресу електронної пошти, а й пароль. Після надання інформації, LinkedIn автоматично проаналізує та розпізнає всі контакти, з якими користувач листувався з e-mail, і після закінчення реєстрації сайт пропонує додати профілі цих людей в свою адресну книгу і розіслати запити на додавання до контактів. Ця функція має сенс у випадку коли, користувач вказує адресу робочої електронної пошти і використовується лише для ділового листування.

Після реєстрації профілю у користувача є можливість подавати заявки і вступати в співтовариства і групи. Групи в даній соціальній мережі відрізняються від аналогічних на Facebook або інших майданчиках. У LinkedIn вони професійні та ділові. Це спільноти співробітників компанії (не офіційна сторінка організації, а саме група працівників), випускників факультетів, університетів, курсів підвищення кваліфікації, галузеві групи тощо (рисунок 1.5).

Рисунок 1.5 – Група співробітників компанії PNN Soft

Дані співтовариства використовуються для моніторингу новин в галузі або компанії між співробітниками, для узгодження дружніх або ділових зустрічей,

для підказок і порад новачкам. Групи можуть бути відкритими, вільними для доступу, або закритими, рішення про прийом користувача бере адміністратор групи.

У соціальній мережі є офіційні сторінки організацій або підприємців, які виступають в ролі роботодавців. Профілі підприємств на сайті відрізняються від сторінок на інших майданчиках обмеженими можливостями для реклами та брендингу. У Facebook, Instagram профілі та спільноти компаній використовуються для просування бренду, в LinkedIn це обмежена. Якщо організація намагається просунути товари і послуги за допомогою соціальної мережі, можливе видалення сторінки.

Корпоративні сторінки в LinkedIn використовуються для публікації пресрелізів і корпоративних новин. Компанії можуть публікувати список співробітників, що працюють в штаті. Відкриті вакансії розміщують у відповідному розділі. Встановити контакт зі сторінкою організації, як з користувачем, неможливо. Але можна відслідковувати оновлення, підписавшись на сторінку компанії. Моніторинг сторінок проводять не тільки претенденти. Відстежують оновлення, щоб бути в курсі подій конкурентів, придивлятися до потенційних партнерів, спостерігати за розвитком лідера галузі тощо. Дуже часто компанії використовують LinkedIn для пошуку співробітників. Є система фільтрів яка дозволяє відсіяти кандидатів, які не відповідають вимогам з тих чи інших причин. Інформація, зазначена в профайлах користувачів частіше виявляється правдивою, ніж в резюме.

Зареєструватися в LinkedIn можна безкоштовно, але після створення профілю соціальна мережа пропонує відкрити платні опції. Платний профіль відкриває додаткові можливості для користувача:

– переглядати сторінки людей, які заходили на його сторінку;

– відкривається розширений доступ до профілів інших користувачів. Інформація може бути корисна при встановленні ділових контактів;

– можливість писати людям, яких немає в списку контактів;

– пошук розширюється додатковими фільтрами;

– додається опція збереження профілів в спеціальну папку на сайті, додавати до них позначки, примітки і контактні відомості.

Основною причиною яка залучає нових користувачів в LinkedIn є вакансії. Фільтри і алгоритми обробки інформації користувачів засновані на нейронних мережах дозволяють LinkedIn автоматично підбирати пропозиції, відповідні кваліфікації та спеціалізації здобувача. Розділ "Вакансії" розташований в нижньому меню (рисунок 1.6).

Якщо пропозиції, які надіслала соціальна мережа, не зацікавили здобувача, він може самостійно провести пошук. Для цього необхідно вибрати поле пошуку, вказати, яку вакансію необхідно знайти, встановити параметри у фільтрах, і вибирати із запропонованих варіантів.

Коли користувач шукає роботу, але ще не звільнився зі старої, і поточний роботодавець зареєстрований у соціальній мережі, можуть виникнути проблеми. Щоб роботодавець не знав про те, що співробітник шукає інше місце працевлаштування або підробіток, можна приховати активність користувача. Опція знаходиться в розділі "Конфіденційність та налаштування" (рисунок 1.7).

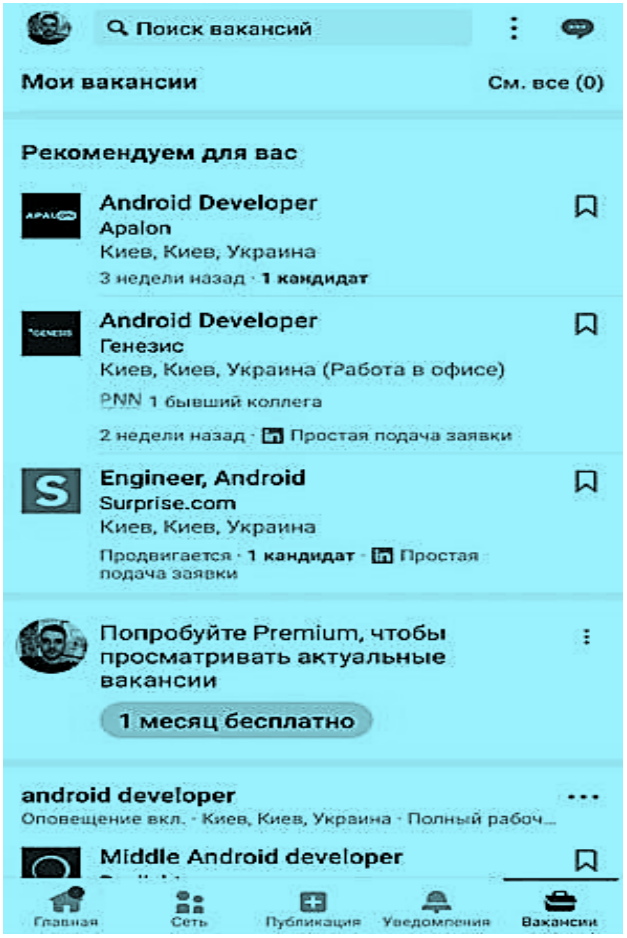

Рисунок 1.6 – Розділ "Вакансії"

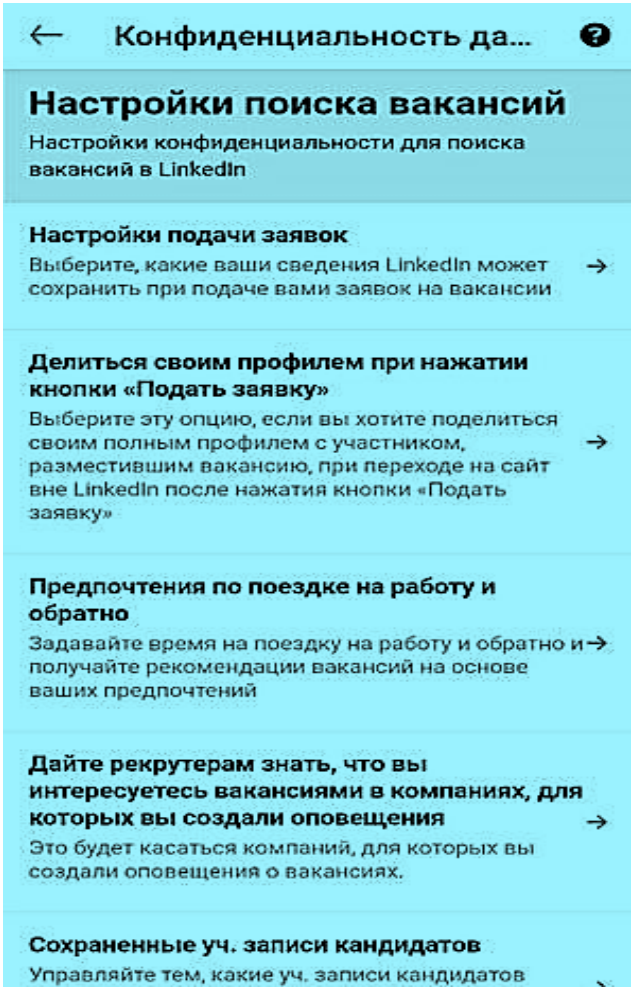

Рисунок 1.7 – Конфіденційність і настройки

При правильному використанні, сервіс може стати відмінним інструментом для пошуку співробітників з необхідними якостями або компанії, яка шукає фахівця на певну посаду. Правильна настройка профілю, публікації, професійні спільноти – все це зроблено для того, щоб пошук був максимально простим і зручним.

Нейромережі в даній соціальній мережі використовують для можливості пошуку роботодавцю нових співробітників і навпаки.

Основними мінусами даної соціальної мережі є вузько направлений профіль, дана соціальна мережа не призначена для звичайного користувача. Масовий користувач не буде використовувати цю соціальну мережу, так як в ній немає багатьох функцій, наприклад такий як можливість викладати "stories".

1.2.2 Огляд та аналіз мережі Instagram

Instagram – додаток основною функцією якого є обмін фотографіями і відеозаписами. Соціальна мережа дозволяє знімати фотографії та відео, обробляти їх за допомогою різних фільтрів, а також поширювати їх через свій сервіс і ряд інших соціальних мереж. Додаток сумісний зі специфікацією iPad, iPhone, і iPod Touch на операційній системі iOS 4.3 і вище, а також з телефонами на Android 2.2 і вище з підтримкою OpenGL ES 2 [3].

Основними можливості Instagram:

- фотографії високої роздільної здатності;
- живі фільтри;
- миттєва зміна нахилу фотографії;
- поворот одним кліком;
- відмітка на фотознімках себе, своїх друзів, цікаві місця тощо.

Після того як користувач зареєструвався, він потрапляє в стрічку де будуть з'являтися останні публікації всіх користувачів, на яких підписаний користувач. Там же будуть з'являтися і публікації які сам користувач публікує (рисунок 1.8).

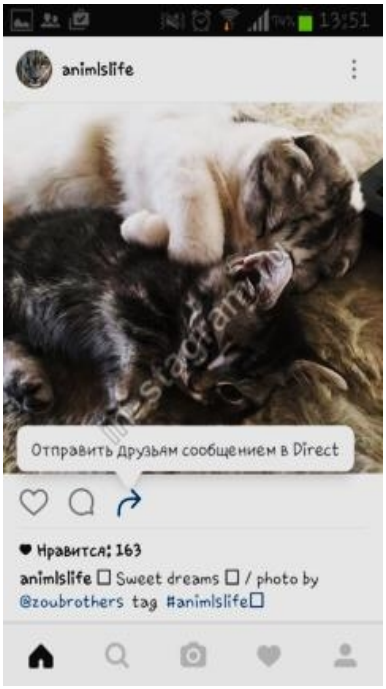

Рисунок 1.8 – Стрічка

Користувач може поставити лайк або залишити коментар (рисунок 1.9).

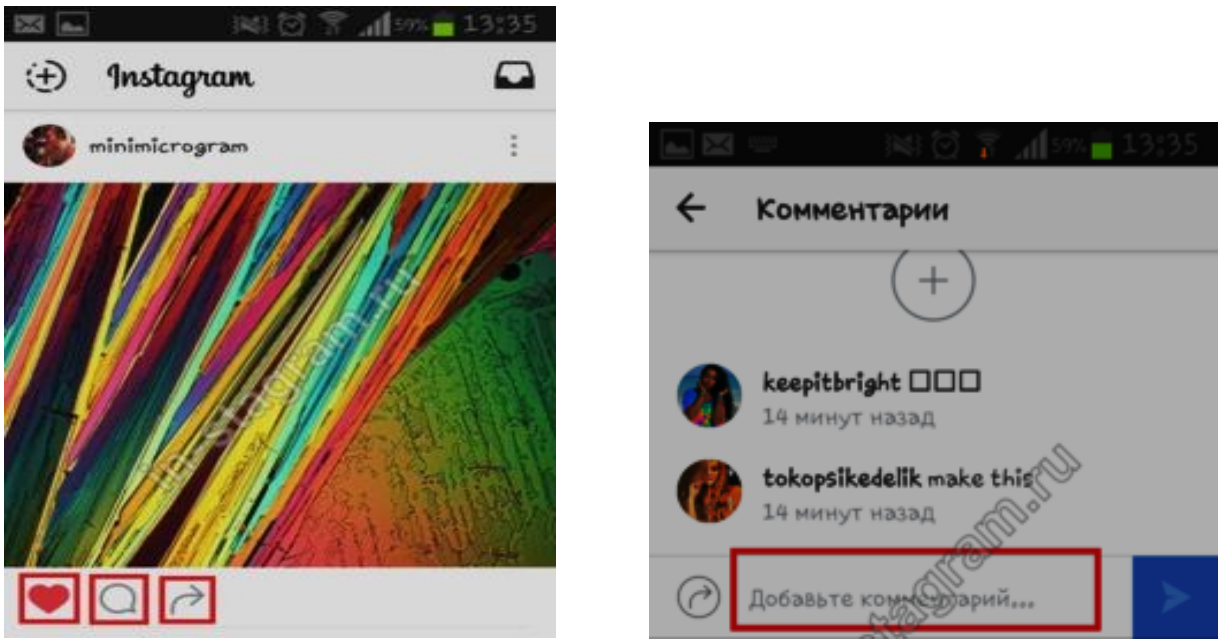

Рисунок 1.9 – Взаємодія

Для цього потрібно натиснути на сердечко під контентом. Для того щоб залишити коментар потрібно натиснути на хмару. Ще можна ділитися контентом через Direct (внутрішній месенджер Instagram) зі своїми друзями якщо натиснути на стрілку.

Щоб зробити фотографію, необхідно натиснути на маленьке зображення камери внизу екрану. Спочатку буде перенаправлення в галерею (звідки також можна вибрати знімок), для того щоб сфотографувати щось прямо зараз, потрібно натиснути на «Фото», а потім - на велику круглу кнопку внизу (рисунок 1.10).

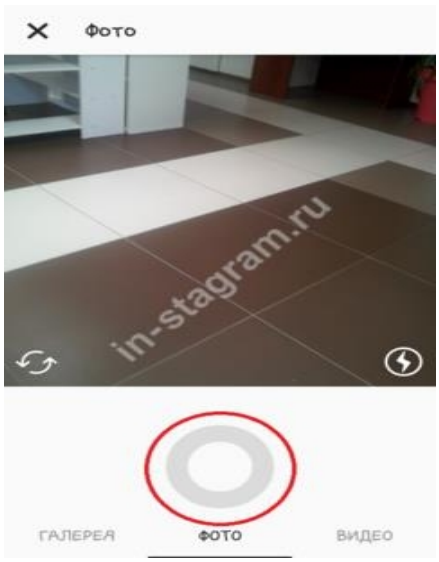

Рисунок 1.10 – Створення знімка

Після того як користувач зробив знімок, він буде перенаправлений на вибір фільтрів (рисунок 1.11). Щоб користуватися фільтрами, потрібно натискати на ту чи іншу вікно під фотографією, фільтр буде застосований відразу ж на фотографії.

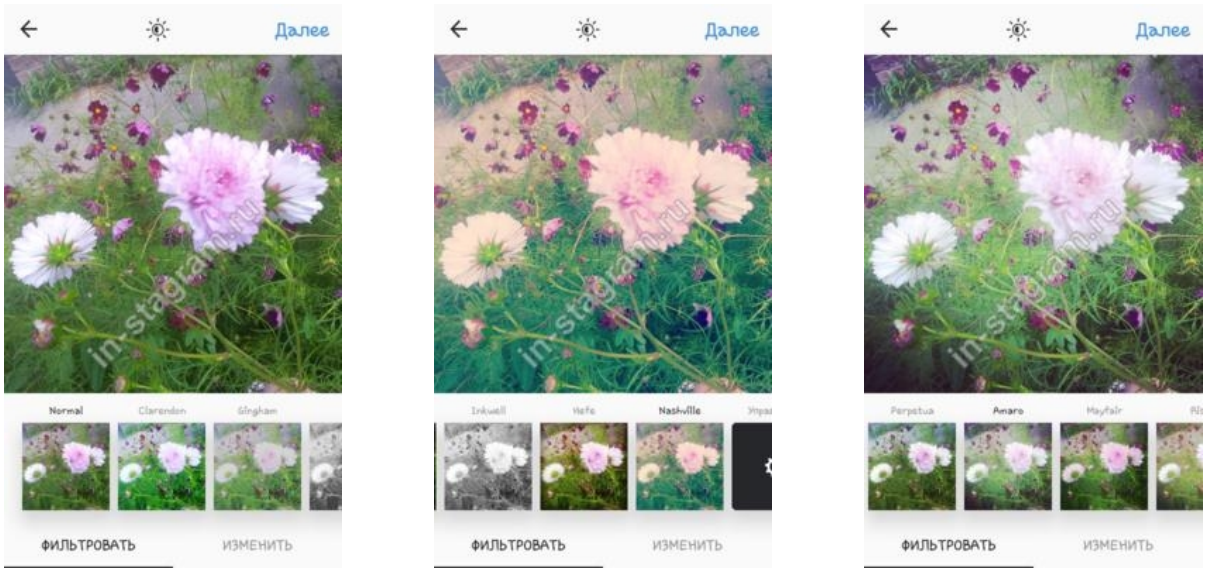

Рисунок 1.11 – Фільтри

Після того як фільтрація фото здійснена, натискаємо кнопку "Далі" і переходимо до публікації. В поле праворуч можна ввести звичайний текст і текст з хештегом (рисунок 1.12).

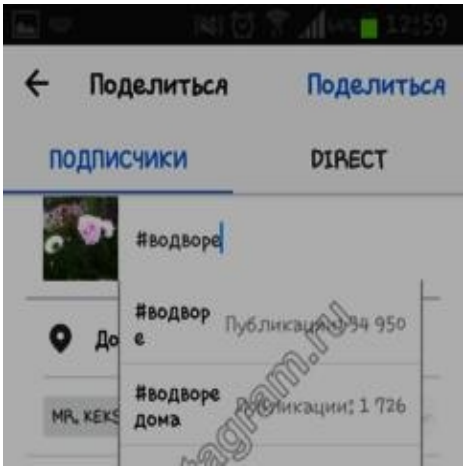

Рисунок 1.12 – Опис знімка

Крім фотографій в Instagram існує функція запису відеороликів

Максимальна тривалість відеоролика становить 30 секунд. На запис так

само як і на фотографію можна накладати різні фільтри, передбачені сервісом. Ще для відеоконтенту можна вибрати обкладинку, яку користувачі будуть бачити коли відео з'явитися в їх стрічці.

Розглянемо як користуватися історіями в Instagram. Історії відображаються в верхній частині, над усіма публікаціями в стрічці. Там можна побачити маленькі аватарки тих, хто опублікував історію, обведені кольоровим колом (рисунок 1.13).

Рисунок 1.13 – Перегляд історії

Щоб подивитися обрану історію, потрібно натиснути на неї. Якщо кілька людей, на яких підписаний користувач, запишуть історії, вони вишикуються в ряд (рисунок 1.14). Якщо запустити одну з них, вони будуть автоматично відтворюватися один за одним.

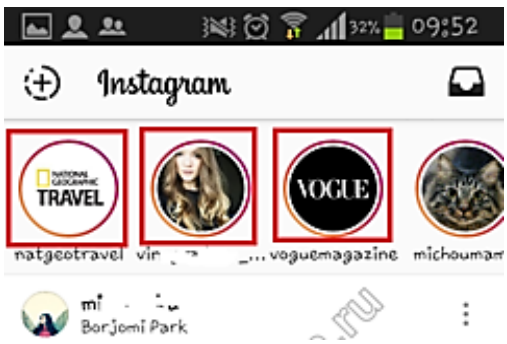

Рисунок 1.14 – Список історій

Після перегляду різнокольоровий коло навколо історії пропаде як зображено на рисунку 1.15.

Рисунок 1.15 – Історія переглянута

Для того щоб записати власну історію, необхідно натиснути на кнопку у верхній частині екрану з лівого боку (рисунок 1.16).

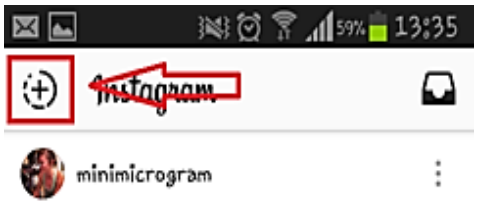

Рисунок 1.16 – Створення історії

Відкривається видошукач. Можна зняти фото чи відео. Щоб сфотографувати, потрібно натиснути на круглу кнопку внизу екрану. Щоб записати відео – затиснути її. Одне відео може бути не більше 30 секунд. Після 30 секунд запис обривається. Після того як фотографія зроблена, зверху екрану з'являються два значка (рисунок 1.17). Натиснувши на перший, з'являється функція малювання. Натиснувши другий – додавання тексту.

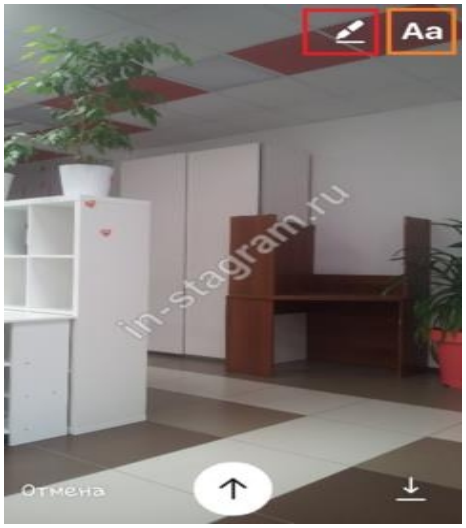

Рисунок 1.17 – Ікони редагування фотографії

Після того як фотографія зроблена і відредаговано, вона з'явиться на головній сторінці серед інших опублікованих "історій" (рисунок 1.18).

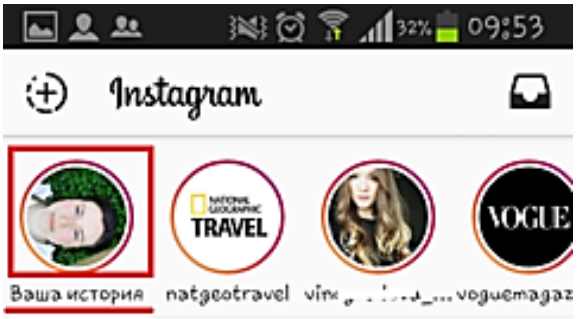

Рисунок 1.18 – Додана історія

У створенні "історій" є одна особливість. Вона висить опублікована тільки 24 години, після закінчення яких видаляється. Користувач може додавати нові "історії", їх час життя буде також становити 24 години.

Нейросеть в даній соціальній мережі використовують для того щоб надати користувачеві новинну стрічку на основі його переваг, а також для пошуку його потенційних знайомих.

Основними мінусами є те що, можна дати тільки загальну оцінку контенту користувача. Якщо користувач викладає фотографію на якій зображено кілька людей, можна поставити лайк всієї фотографії. Не можна оцінити одного контрактного людини.

1.3 Висновки та постановка задачі

Нейромережа вдає із себе математичну модель, яка намагається імітувати роботу людського мозку. Нейронна мережа складається з різних шарів нейронів. Кожен шар отримує на вхід інформацію від попереднього шару, обробляє її і передає на вхід наступного шару. Кожен нейрон це своя мініатюрна модель з набором вхідних особливостей, ваг і поведінкою.

Нейронні мережі вже давно глибоко використовують в сучасному інтернеті і соціальні мережі не є винятком. Нейронні мережі в соціальних мережах

допомагають вирішувати безліч проблем, в яких використання звичайних алгоритмів може бути дуже дорогим тривалим і в цілому неефективним заняттям. Нейронні мережі допомагають користувачам в пошуку контенту, в пошуку знайомих. Вони створюють комфортні умови для проведення часу на майданчику, і тим самим генеруючи прибуток для власників цих майданчиків.

Метою даної роботи є розробити соціальну мережу з використанням нейронної мережі для системи оцінювання контенту.

Були розглянуті соціальні мережі які в своїй роботі використовують навчені нейронні мережі. Але не одна з перерахованих вище майданчиків не надає можливості оцінки контенту на основі нейронних мереж.

Для досягнення поставленої мети необхідно навчити нейронну мережу на готової моделі даних для класифікації об'єктів на зображенні, а також розробити мобільний додаток яке буде представляти із себе соціальну мережу.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Мета даної роботи полягає у розробці соціальної мережі з використанням нейронної мережі для аналізу контенту. Нейронна мережа повинна обробляти зображення, виявляти на них об'єкти та класифікувати їх.

Використання у роботі класичних нейронних мереж для обробки зображень утруднено, це пов'язано з великою розмірністю вектора вхідних значень нейронної мережі, великою кількістю нейронів у проміжних шарах і, як наслідок, чималими витратами обчислювальних ресурсів на обчислення мережі та навчання. Для згорткових нейронних мереж меншою мірою властиві описані вище недоліки.

Згорткова нейронна мережа – це спеціальна архітектура штучних нейронних мереж, яку запропонував Ян Лекун. Вона орієнтована на ефективну обробку зображень. Ця нейронна мережа перебуває у складі технологій глибокого навчання. Ця технологія побудована на основі принципів роботи зорової кори головного мозку, в ньому були відкриті так звані прості клітини, вони реагують на прямі лінії під різними кутами та складні клітини, реакція яких пов'язана з активацією певного набору простих клітин. Таким чином, основна ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів та субдискретизуючих шарів (шарів підвиборки). Структура згорткової нейронної мережі – односпрямована (не має зворотних зв'язків), багатошарова. Використовуються стандартні методи навчання, переважно метод зворотного поширення помилки. Функція активації нейронів (передавальна функція) – будь-яка, на вибір дослідника. Архітектура згорткової нейронної мережі зображено рисунку 2.1.

Основним моментом для розуміння згорткових нейронних мереж необхідно поняття «ваг, що розділяються», частина нейронів деякого розглянутого шару всієї нейронної мережі може використовувати одні і ті ж вагові коефіцієнти. Нейрони, які використовують ті самі ваги, об'єднуються в карти ознак, і будь-який нейрон з карти ознак пов'язаний з частиною нейронів, які знаходяться в попередньому шарі.

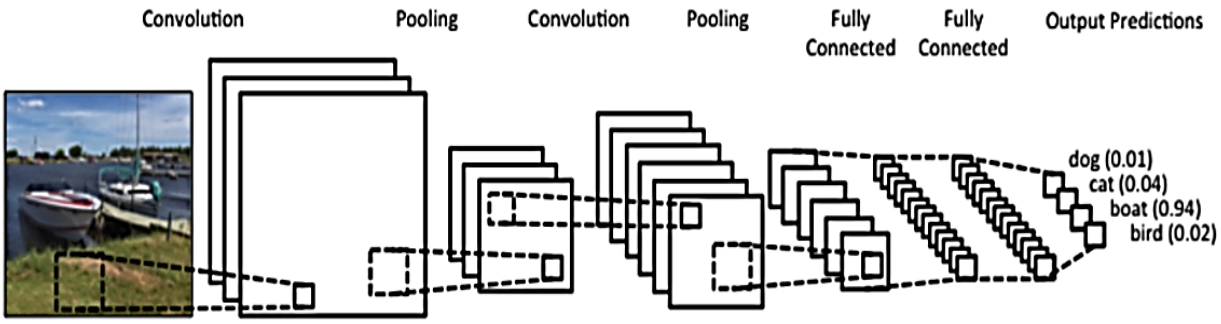

Рисунок 2.1 – Архітектура згорткової нейронної мережі

Під час обчислення мережі відбувається так, що кожен нейрон виконує операцію конволюції (згортка) деякої області попереднього шару (визначається безліччю нейронів, пов'язаних з 14 даним нейроном). Шари, які були побудовані таким чином, називають згорткові шари. Також у згортковій нейронній мережі можуть бути шари, що виконують функції зменшення розмірності простору карт ознак (шари субдискретизації) та повнозв'язкові шари (вихідний шар, в основному завжди повнозв'язковий). Всі три види шарів можуть чергуватись у довільному порядку, це дає можливість складати карти ознак з карт ознак, на практиці це дає здатність розпізнавання складних ієрархій ознак.

Різні нейронні мережі використовують різні алгоритми реалізації своєї роботи. Існує безліч алгоритмів зі своїми перевагами та недоліками. Нижче буде проаналізовано роботу алгоритмів сімейства YOLO. Буде розглянуто їх оригінальність, авторство, продуктивність та основні покращення.

YOLO (You Only Look Once) – це сучасний детектор об'єктів, який може виконувати виявлення об'єктів у реальному часі з гарною точністю. Приклад використання даного алгоритму можна побачити рисунку 2.2.

Алгоритм YOLO був уперше представлений світу у 2016 році. Він став важливою віхою у дослідженнях щодо виявлення об'єктів. Все завдяки своїй здатності виявляти об'єкти в режимі реального часу з більшою точністю. Автором алгоритму є Джозеф Редмон, випускник Вашингтонського університету. Документ із описом YOLO отримав приз OpenCV People's Choice Award на Конференції з комп'ютерного зору та розпізнавання образів (CVPR) у 2016 році.

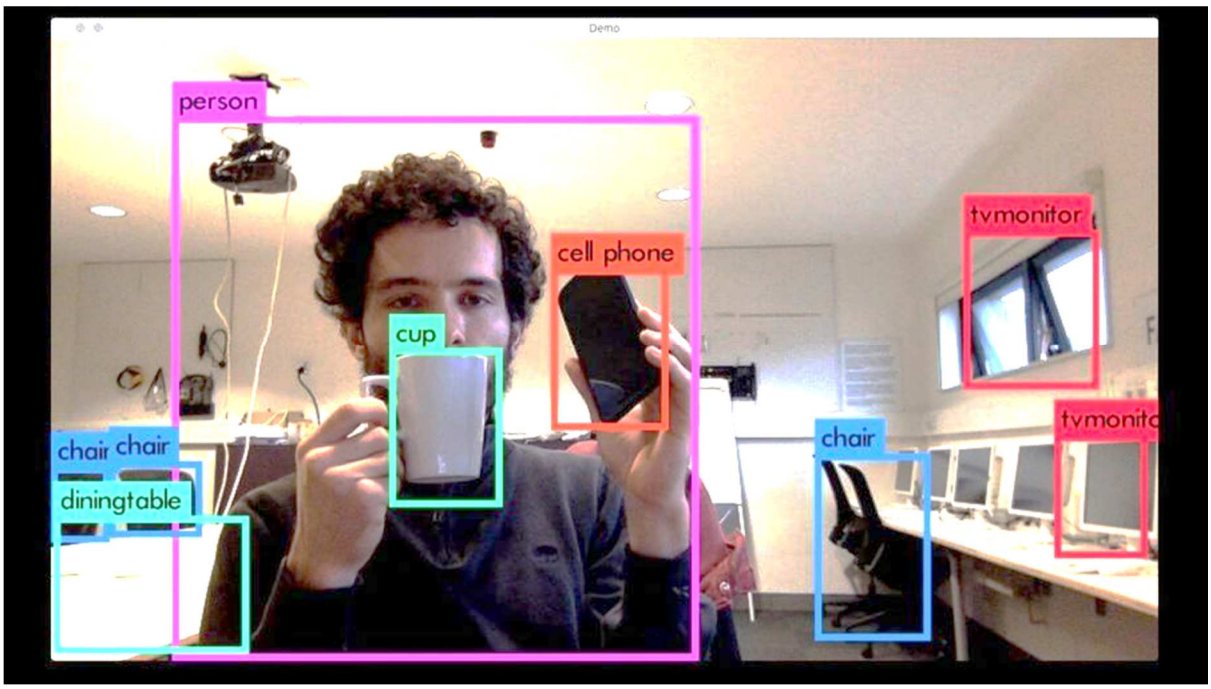

Рисунок 2.2 – Виявлення об'єктів на зображенні

Алгоритм YOLOv2 був випущений у 2017 році, він може обробляти зображення зі швидкістю 40-90 кадрів за секунду. Але в наступному році світу був представлений алгоритм YOLOv3, який надавав можливість легко знаходити компроміс між швидкістю і точністю, просто змінюючи розмір моделі без будь-якого перенавчання. Архітектура алгоритму YOLOv3 представлена рисунку 2.3

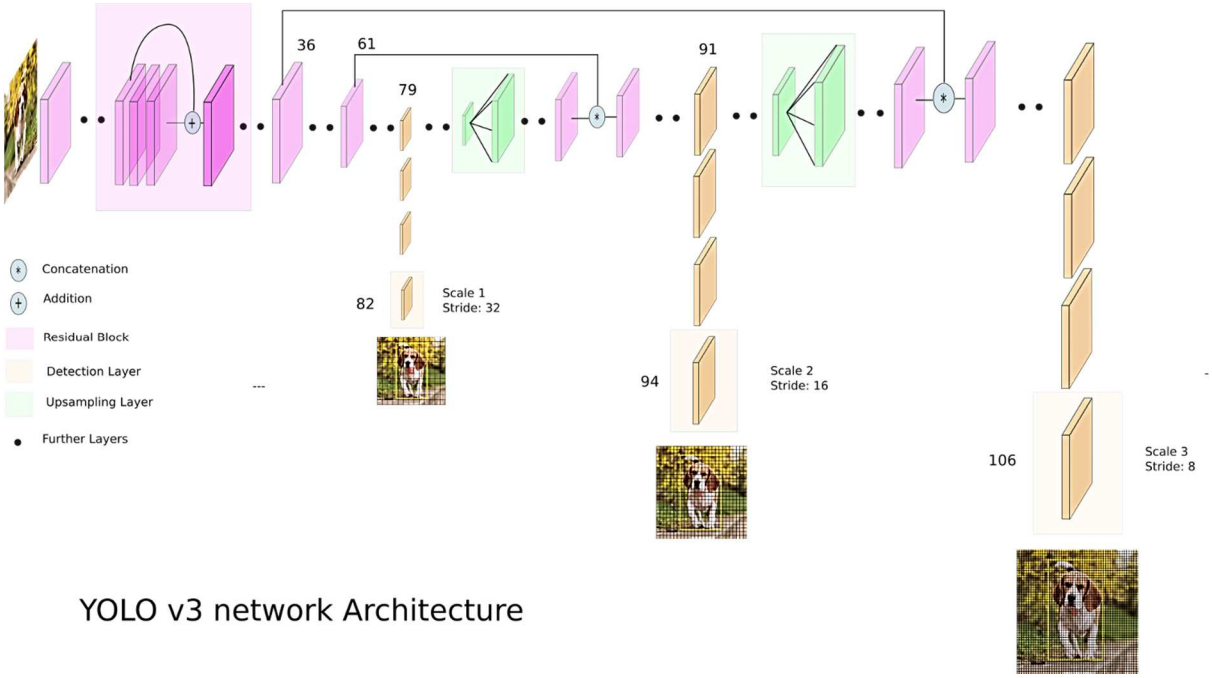

Рисунок 2.3 – Архітектура алгоритму YOLOv3

У таблиці 2.1 представлено порівняння алгоритмів YOLOV2 і YOLOV3 на різних моделях даних.

Таблиця 2.1 Продуктивність алгоритмів YOLOv2 та YOLOv3

Model	Train	Test	mAP(Точність)	FPS
YOLO	VOC 2007+2012	2007	63.4	45
Fast YOLO	VOC 2007+2012	2007	52.7	155
YOLOv2	VOC 2007+2012	2007	76.8	67
Tiny YOLO	VOC 2007+2012	2007	57.1	207
Tiny YOLO	COCO trainval	-	-	200
YOLOV3-320	COCO trainval	test-dev	51.5	45
YOLOV3-tiny	COCO trainval	test-dev	33.1	220
YOLOV3-ssp	COCO trainval	test-dev	60.6	20

Основна реалізація YOLO заснована на Darknet, фреймворку нейронної мережі з відкритим вихідним кодом, написаним на C і CUDA. Darknet описує базову архітектуру мережі та використовується як основа для навчання YOLO. Ця реалізація представлена Редмоном, вона швидка, проста в установці та підтримує обчислення на CPU та GPU.

Алгоритм YOLO v4 використовує сучасний BoF (bag of freebies) та кілька BoS (bag of specials). BoF підвищує точність виявлення об'єктів без часу роботи алгоритму. Вони лише збільшують вартість ресурсів, необхідну для навчання нейромережі. З іншого боку, BoS трохи збільшують вартість виведення, проте вони значно підвищують точність виявлення об'єктів. YOLOv4 також розроблений на фреймворку Darknet, показуючи швидкість у реальному часі 65 FPS на моделі Tesla V100, випереджаючи найшвидші та найточніші детектори з точки зору швидкості та точності. Порівняно з YOLOV3 FPS збільшилися на 10% і 12% відповідно.

Алгоритм YOLOv5 відрізняється від своїх попередніх випусків, оскільки ця реалізація зроблена на PyTorch, а не на Darknet як попередні версії. Як і

YOLOv4, YOLOv5 має магістраль CSP та шию PA-NET. Основні покращення YOLOv5 містять у собі збільшення даних мозаїки та автоматичне навчання прив'язок обмежуючого прямокутника. YOLOv5 відноситься до архітектури One-Stage detector - підхід, який передбачає координати певної кількості bounding box'ів з результатами класифікації та ймовірності знаходження об'єкта, і надалі коригуючи їх місцезнаходження. Архітектура алгоритму YOLOv5 представлена рисунку 2.4

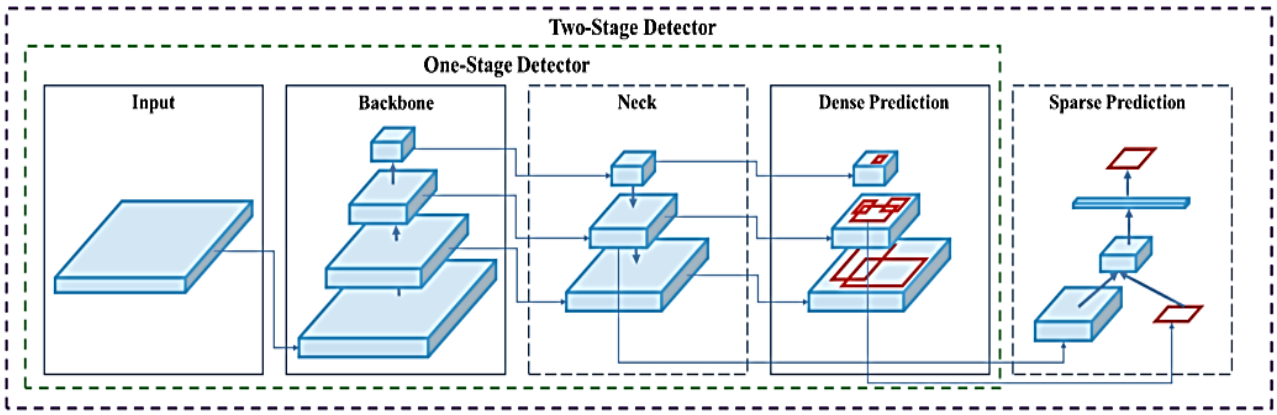

Рисунок 2.4 – Архітектура алгоритму YOLOv5

Алгоритм YOLOv5 скейліт вихідне зображення в кілька feature map'ів з використанням skip-connection та інших архітектурних особливостей. Отримані карти ознак наводяться в один дозвіл за допомогою апсемплінгу та конкатенуються. Потім передбачаються класи та bounding box'и для об'єктів, далі для кожного об'єкта вибирається найімовірніший bounding box за допомогою Non-Maximum Suppression.

Переваги вибору алгоритму YOLOV5

– це перший власний випуск моделей у сімействі YOLO, написаний на PyTorch, а не на Darknet. Darknet – це неймовірно гнучка дослідницька структура, але вона не розрахована на виробниче середовище;

– алгоритм YOLOv5 швидкий у тестах на моделі даних Tesla P100 час виведення становив до 0,007 секунди на зображення, що означає 140 кадрів на секунду;

– алгоритм YOLOv5 точний у тестах набору даних з підрахунку та виявлення клітин крові (BCCD) був досягнутий результат приблизно 0,895 середньої точності (mAP) після навчання всього за 100 епох;

– алгоритм YOLOV5 маленький. Зокрема, файл ваги для YOLOv5 складає 27 мегабайт. Файл ваги для YOLOv4 (з архітектурою Darknet) складає 244 мегабайти. YOLOV5 майже на 90 відсотків менше, ніж YOLOV4. Це означає, що YOLOV5 набагато простіше розгорнути на вбудованих пристроях.

Для виконання цієї роботи було виявлено такі функціональні вимоги:

– розроблена система має розпізнавати людей, тварин, машини та інші об'єкти;

– розроблена система має класифікувати зображення з прийнятною точністю;

– розроблена система повинна зберігати навчену модель її подальшого використання.

3 АНАЛІЗ І ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ І МОВ ПРОГРАМУВАННЯ

3.1 Об'єктно-орієнтована мова програмування Kotlin

Kotlin (Ко́тлін) – об'єктно-орієнтована, статично типізована мова програмування, яка працює поверх віртуальної машини Java. Був придуманий та розроблений компанією JetBrains. Kotlin може бути компілюємо JavaScript і виконуваний код ряду інших платформ через інфраструктуру LLVM. Назва мови було обрано на честь острова Котлін у Фінській затоці, де знаходиться місто Кронштадт [4].

Основна мета творців мови полягала у створенні більш лаконічної та типобезпечної мови, ніж Java, і більш простої, ніж Scala. Мова краще підтримується в IDE ніж Scala і порівняно зі Scala відбувається більш швидка компіляція. Kotlin сумісний з Java, що дозволяє Java розробникам поступово перейти до його використання; зокрема, Kotlin вбудовується в Android, що дозволяє для існуючого android-програми впроваджувати нові функції на Kotlin без переписування програми цілком [4].

Синтаксис мови Kotlin використовує елементи з TypeScript, Haxe, F#, JavaScript, Паскаля, Go та Scala, C++, PL/SQL Java, C#, Rust і D. Для оголошення параметрів і змінних типи даних вказуються після назви (розділювач — двокрапка). Не обов'язкова точка з комою, як роздільник операторів (як у Groovy, Scala та JavaScript); в основному перекладу рядка достатньо, щоб компілятор зрозумів, що вираз закінчився. Крім об'єктно-орієнтованого підходу Kotlin також підтримує процедурний стиль з використанням функцій. Як і Ci, C++ і D, точка входу в програму – функція main, що приймає масив параметрів командного рядка. Підтримується виведення типів. Також програми написані на Kotlin підтримують perl-і shell-стиль інтерполяції рядків (змінні, включені в рядок, замінюються на вміст) [4].

3.2 Об'єктно-орієнтована мова програмування Java

Java – мова загального призначення, що є суворо типізованою об'єктно-орієнтованою мовою програмування, була розроблена компанією Sun Microsystems (надалі компанія була викуплена компанією Oracle). Розробка мови здійснюється спільнотою, організованою через Java Community Process; Java та основні технології, що реалізують її, поширюються за ліцензією GPL. Правами дану торгову марку має корпорації Oracle [5].

Основна перевага способу виконання програм написаних на Java полягає в тому, що відбувається повна незалежність байт-коду від операційної системи та обладнання, на якому це виконується, це дозволяє виконувати програми написані на Java на будь-якому пристрої, для якого існує відповідна віртуальна машина. Ще однією важливою особливістю технології Java є гнучка система безпеки, у рамках якої виконання програми повністю контролюється віртуальною машиною. Усі взаємодії та операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних чи з'єднання з іншим комп'ютером), викликають негайне переривання [5].

Android активно використовує мову Java для створення мобільних програм під свою операційну систему. При цьому програми компілюються в нестандартний байт-код для використання їхньою віртуальною машиною Dalvik (з версії Android 5.0 Lollipop JVM замінили на ART). Для цього компанія Google розробила спеціальний інструмент Android SDK (Software Development Kit). Розробку програм можна вести в середовищі Eclipse, використовуючи плагін Android Development Tools (ADT), в середовищі Android Studio, NetBeans або IntelliJ IDEA. Версія JDK [5].

3.3 Високорівнева мова програмування Python

Python – високорівнева мова програмування загального призначення з автоматичним управлінням пам'яттю та динамічною строгою типізацією, орієнтована на підвищення продуктивності розробника, читання коду та його якості, а

також на забезпечення переносимості написаних на ньому програм. Python використовує об'єктно-орієнтовану парадигму програмування, все в Python складається з об'єктів. З основних особливостей мови Python можна виділити відсутність розділення блоків коду фігурними дужками, для цього використовуються пробіли та відступи. Як кажуть користувачі, синтаксис ядра мінімалістичний, рахунок цього рідко виникає необхідність звертатися до документації. Мова часто використовують для написання скриптів, так як він є інтерпретованим. Основним недоліком можна відзначити нижчу швидкість роботи і більш високе споживання пам'яті порівняно з аналогічним кодом, написаним компілюваними мовами, таких як Сі або C++. Python є мультипарадигмальною мовою програмування, що підтримує імперативне, об'єктно-орієнтоване програмування, структурне, процедурне, метапрограмування та функціональне програмування. Аспектно орієнтоване програмування частково підтримується через декоратори, для більш повноцінної підтримки необхідно використовувати додаткові фреймворки. Основні архітектурні риси - динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора, високорівневі структури даних. Завдання узагальненого програмування вирішуються рахунок динамічної типизации. Такі методики як контрактне та логічне програмування можна реалізувати за рахунок розширень та бібліотек. Програми, написані на Python, можуть розділитися на модулі і об'єднаються в пакет [6].

Інтерпретатор CPython є еталонною реалізацією Python. Він підтримує більшість платформ, які активно використовуються і є стандартом мови. CPython розповсюджується під вільною ліцензією Python Software Foundation License. Ця ліцензія дозволяє використовувати інтерпретатор без обмежень у будь-яких програмах, включаючи пропрієтарні. Робота CPython полягає в тому, що він компілює вихідні тексти у високорівневий байт-код, а він у свою чергу виконується у стіковій віртуальній машині. До інших трьох основних реалізацій мови відносяться Jython (для JVM), IronPython (для CLR/.NET) та PyPy. PyPy підвищує шви-

дкість виконання програм за рахунок використання JIT-компіляції, він написаний на підмножині мови Python (RPython) та розроблявся як альтернатива CPython. 2020 року закінчилася підтримка версії Python 2 [6].

3.4 Система автоматичного збирання Gradle

Gradle – система призначена для автоматичного складання, спосіб роботи заснований на принципах Apache Ant і Apache Maven, взаємодія здійснюється на мовах Groovy і Kotlin замість традиційної XML-подібної форми представлення конфігурації проекту. На відміну від Apache Maven, заснованого на концепції життєвого циклу проекту, та Apache Ant, у якому порядок виконання завдань (targets) визначається відносинами залежності (depends-on), Gradle заснований на спрямованому ациклічному графі визначення порядку виконання завдань. Gradle був придуманий і розроблений для багатопроектних збірок, що розширюються. Gradle підтримує каскадну (waterfall) модель розробки, може визначати, які компоненти дерева складання не змінилися і які завдання, залежні від цих частин, не вимагають перезапуску. Gradle свій власний Gradle Daemon – фоновий процес для прискорення складання проекту. Основні плагіни призначені для розробки та розгортання Java, Groovy та Scala програм, але є плагіни для інших мов програмування: Kotlin C++, Swift [7].

3.5 Розширювана мова розмітки XML

XML – являє собою мову розмітки, що розширюється. Його рекомендує для використання Консорціумом Всесвітньої павутини (W3C). Специфікація XML описує XML-документи та частково описує поведінку XML-процесорів (програм, які читають XML-документи та забезпечують доступ до їхнього вмісту). Основний задум при розробці мови XML був у тому, щоб можна було передавати дані простим формальним синтаксисом, зручним для створення та обробки документів як людиною, так і програмами, в основному для використання

в Інтернеті. Мова є розширюваною, тому що вона не фіксує розмітку, яка використовується в документах: розробник може сам створити розмітку відповідно до своїх потреб, основні обмеження буде лише в синтаксичних правилах мови. Існують такі речі як розширення XML - це конкретний набір правил граматики, що визначають, які атрибути та елементи можуть входити до складу інших елементів, створених на базі XML. Розширення включає словник тегів та їх атрибутів. Поєднання простого формального синтаксису, зручності для людини, розширюваності, а також базування на кодуваннях Юнікод для того щоб відобразити зміст документів призвело до широкого використання як, власне, XML, так і безлічі похідних спеціалізованих мов на базі XML у найрізноманітніших програмних засобах [8]

3.6 Компактна вбудована СУБД SQL

SQLite – це портативна СУБД, що вбудовується. Вихідний код цієї бази даних було передано на громадське надбання. 2005 року проект отримав нагороду Google-O'Reilly Open Source Awards. SQLite не схоже на інші бази даних за рахунок того, що не використовує клієнт-серверну парадигму, движок SQLite не є окремо працюючим процесом в системі, з яким взаємодіє програма користувача, SQLite являє собою бібліотеку, з якою програма входить у симбіоз, і движок SQLite стає складовою програми. Як протокол обміну даними між SQLite та програмою використовуються виклики функцій (API). За рахунок такого підходу можна зменшити час відгуку, накладні витрати і це спрощує програму. У комплекті з бібліотекою також йде функціональна клієнтська частина у вигляді sqlite3, за допомогою цього файлу показується реалізація функцій основної бібліотеки. Клієнтська частина є кросплатформною утилітою командного рядка. Бібліотеку можна використовувати як на виділених машинах з гігабайтними масивами даних так і на системах, що вбудовуються. Також бібліотека зберігає у собі всю базу даних (включаючи індекси, визначення, дані та таблиці) в єдиному ста-

ндартному файлі на тій системі, на якій виконується програма. Витонченість реалізації бібліотеки полягає в тому, що перед виконанням транзакції запису весь файл, що зберігає базу даних, блокується; ACID-функції досягаються за рахунок того, що відбувається створення файлу журналу. З однієї бази можуть одночасно читати дані кілька процесів чи потоків без будь-яких проблем. Взаємодія з базою, а саме запис можна виконати тільки якщо жодних інших запитів на цей час не обслуговується; інакше спроба запису закінчується невдачею, і програму повертається колбек з кодом помилки. Також можна настроїти автоматичне повторення спроб запису протягом заданого інтервалу часу [9].

3.7 База даних Firebase у реальному часі

DataBase Firebase Realtime – це база даних, яка розміщена у хмарному просторі. Дані користувачів зберігаються у JSON форматі та синхронізуються в реальному часі з кожним підключеним клієнтом. Коли клієнт створює кросплатформні програми за допомогою платформ Android, Apple та SDK для JavaScript, всі користувачі клієнта можуть використовувати один екземпляр бази даних у реальному часі та автоматично отримувати оновлення з новітніми даними. Замість звичайних HTTP-запитів база даних Firebase Realtime використовує синхронізацію даних, щоразу, коли хтось змінює дані, будь-який підключений пристрій отримує це оновлення протягом мілісекунд. Це дозволяє забезпечити спільну роботу не дбаючи про мережевий код. Програми Firebase залишаються чуйними навіть коли клієнт знаходиться в автономному режимі, оскільки Firebase Realtime Database SDK зберігає дані на диску. Після відновлення підключення клієнтський пристрій отримує всі пропущені зміни, синхронізуючи їх із поточним станом сервера. Отримання доступу до баз даних Firebase здійснюється безпосередньо з мобільного пристрою або веб-браузера; немає потреби в сервері програм.

Безпека та перевірка даних доступні через правила безпеки бази даних Firebase у реальному часі, правила на основі виразів, які виконуються під час читання або запису даних. Немає потреби в сервері програм. Безпека та перевірка

даних доступні через правила безпеки бази даних Firebase у реальному часі, правила на основі виразів, які виконуються під час читання або запису даних. Немає потреби в сервері програм. Безпека та перевірка даних доступні через правила безпеки бази даних Firebase у реальному часі, правила на основі виразів, які виконуються під час читання або запису даних [10].

3.8 Інтегроване середовище розробки Android Studio

Android Studio – інтегроване середовище розробки (IDE) для роботи з платформою Android. Це середовище розробки засноване на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, будучи офіційним засобом розробки Android додатків. Android Studio доступна для macOS, Windows та GNU/Linux. Android Studio підтримує мову програмування Kotlin, що було анонсовано 17 травня 2017 року, на щорічній конференції Google I/O. Далі Kotlin був обраний офіційною мовою програмування для платформи Android на додаток до Java і C++. В Android Studio немає можливості відключення автозбереження файлів. Це важлива безальтернативна позиція розробників цієї IDE. Розробники вважають, що такий підхід ефективніший, ніж той, який комусь зручний і звичний [11].

На даний момент в Android Studio доступні такі функції [11]:

– розширений редактор макетів: WYSIWYG, здатність працювати з UI компонентами за допомогою Drag-and-Drop, функція попереднього перегляду макета на декількох конфігураціях екрана;

– складання додатків, заснована на Gradle;

– вбудована підтримка Google Cloud Platform, яка включає інтеграцію з сервісами Google Cloud Messaging і App Engine;

– різні види збірок та генерація декількох .apk файлів;

– рефакторинг коду;

– статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій та інше;

– вбудований ProGuard та утиліта для підписування додатків;

– шаблони основних макетів та компонентів Android;
– підтримка розробки програм для Android Wear та Android TV.

3.9 Інтегроване середовище розробки PyCharm

PyCharm – це інтегроване середовище розробки мови програмування Python. Дана IDE надає графічний налагоджувач, засоби для аналізу коду, інструмент для запуску юніт-тестів та підтримує веб-розробку з використанням фреймворку Django. PyCharm була розроблена на основі IntelliJ IDEA компанією JetBrains. PyCharm є кросплатформовим середовищем розробки, IDE сумісна з macOS, Windows, Linux. PyCharm Community Edition (безкоштовна версія) знаходиться під ліцензією Apache License, а PyCharm Professional Edition (платна версія) є пропрієтарним програмним забезпеченням.

На даний момент у PyCharm доступні такі функції:

– налагодження коду за допомогою PyDev;
– підтримка Git, SVN, Mercurial та інших систем контролю за версіями;
– автодоповнення коду;
– рефакторинг коду.

PyCharm має можливість писати власні плагіни, тим самим можна розширювати можливості PyCharm. Деякі плагіни з інших JetBrains IDE можуть працювати з PyCharm. Існує більше тисячі плагінів, сумісних з PyCharm [12].

3.10 Бібліотека Retrofit2

Retrofit – це REST клієнт для Java та Android. Ця бібліотека дозволяє дуже просто отримати та завантажити JSON (або інші структуровані дані) через вебсервіс на основі REST. У Retrofit налаштовується конвертер для серіалізації даних. Для JSON зазвичай використовується GSon, але також можете додавати власні конвертери для обробки XML чи інших протоколів. Бібліотека Retrofit полегшує взаємодію з REST API сайту. Авторами бібліотеки Retrofit є розробники з

компанії Square, які написали безліч корисних бібліотек, наприклад Okhttp і Picasso. Бібліотекою зручно користуватися для запиту до різних веб-сервісів із командами GET, POST, PUT, DELETE. Бібліотека може працювати в асинхронному режимі, що позбавляє зайвого коду [13].

3.11 Бібліотека Dagger2

Dagger 2 – це повністю статичний фреймворк для впровадження залежностей у Java та Android, що працює під час компіляції. Dagger 2 – це адаптація створеного раніше компанією Square фреймворку Dagger, яку підтримує компанія Google.

Плюси даггера порівняно з іншими бібліотеками:

- генерує код нескладний для розуміння та налагодження;
- не створює проблем при використанні proguard;
- перевіряє залежність на етапі компіляції.

3.12 Бібліотека Glide

Glide – бібліотека призначена для асинхронного завантаження зображень із ресурсів, мережі або файлової системи. Бібліотека виконує кешування та відображення зображень. Glide є найближчим конкурентом іншої популярної бібліотеки – Picasso, але є більш популярною завдяки своїй оптимізації на роботі з великими зображеннями. Синтаксис і принцип роботи дуже схожий. Бібліотека підтримує функціональність у вигляді підтримки анімованих GIF-файлів та відео [14].

Google дуже високо оцінює якість бібліотеки і також включив її до складу одного із шаблонів серед програмування Android Studio [14].

3.13 Бібліотека Compressor

Compressor – це легка та потужна бібліотека для стиснення зображень для Android. Компресор надає можливість використовувати функціонал для стиснення зображень зображення меншого розміру з дуже незначною або зовсім незначною втратою якості.

3.14 Висновки

Всі перелічені мови програмування, інструменти, бібліотеки та технології будуть використані для розробки мобільного додатка.

4 ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ НЕЙРОННОЇ МЕРЕЖІ НА МОБІЛЬНОМУ І ХМАРНОМУ ПРОСТОРІ

4.1 Нейронна мережа знаходиться у хмарному просторі AWS Elastic Beanstalk

AWS Elastic Beanstalk – це простий у використанні сервіс для розгортання та масштабування інтернет-додатків та сервісів, розроблених за допомогою Java, .NET, PHP, Node.js, Python, Ruby, Go та Docker, на серверах Apache, Nginx, Passenger та IIS .

Для звернення до нейронної мережі було написано спеціальний API на фреймворку Falsk. Насправді цей метод є клієнт серверну архітектуру. Є якийсь сервіс, який постійно працює у хмарі, до нього надходять запити, він їх обробляє та надсилає відповіді. Даний сервіс доступний для користування за наступною URL-адресою:http://jarvistiny-env-1.eba-gkqdmjmm.us-east-2.elasticbeanstalk.com/.

Архітектура даного сервісу зображено рисунку 4.1.

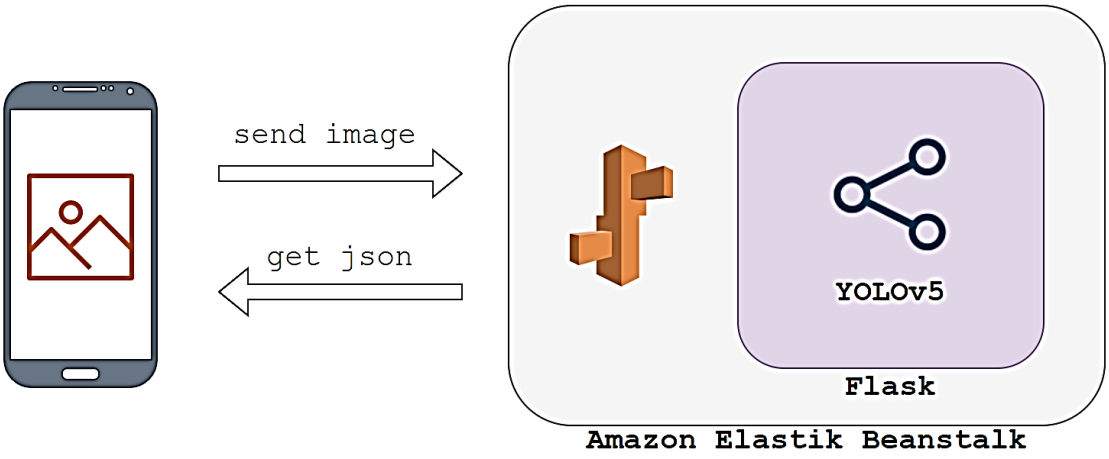

Рисунок 4.1 – Архітектура роботи сервісу

На практиці взаємодія з даним сервісом відбувається таким чином, наприклад, ми ходимо ідентифікувати людину зображену на рисунку 4.2.

Рисунок 4.2 – Приклад фотографії для класифікації

Для цього ми надсилаємо зображення за URL-адресою: http://jarvistiny-env-1.eba-gkqdmjmm.us-east-2.elasticbeanstalk.com/detections

Як приклад будемо використовувати програму для взаємодії з різними адресами API POSTMAN (рисунок 4.3).

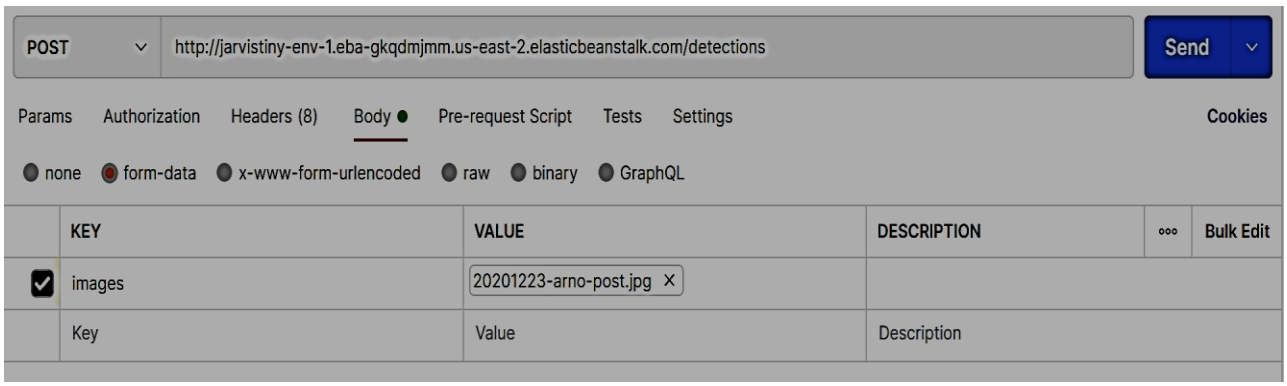

Рисунок 4.3 – Надсилання зображення через POSTMAN

Після того як сервіс отримав зображення та обробив його, він повертає нам відповідь. У статусі ми бачимо код 200, це означає, що обробка пройшла успішно. А в тілі відповіді ми бачимо формат JSON. (Рисунок 4.4)

Рисунок 4.4 – Відповідь сервісу після обробки зображення

У JSON відповіді можна побачити такі значення:

– detections – масив з усіма знайденими об'єктами на зображенні;
– class – назва класу об'єкта (Людина, машина, кіт, собака);
– confidence – точність передбачення;
– x1y1x2y2 – масив координат рамки навколо знайденого об'єкта;
– width/height – ширина висота зображення;
– image – ім'я зображення.

Також в даному сервісі є URL-адреса за якою можна отримати готове зображення з виділеними об'єктами. Наприклад відправимо зображення людини (див. рисунок 4.1) за цим URL:http://jarvistiny-env-1.eba-gkqdmjmm.us-east-2.elasticbeanstalk.com/image.

У відповідь отримаємо зображення рисунку 4.5.

Рисунок 4.5 – Фото після класифікації

Плюс використання такого підходу можна виділити кілька факторів. Перше, це те, що процес обчислень нейронної мережі відбувається не на стороні клієнта, а на стороні сервера. У такому разі не має значення, якими характеристиками володіє мобільний пристрій клієнта. Другим чинником можна назвати легке масштабування, оскільки Amazon має дуже багато призначених при цьому сервісів.

4.2 Нейронна мережа знаходиться на мобільному пристрої

Другий спосіб полягає в тому, що нейронна мережа буде на мобільному пристрої. Для здійснення поточного завдання необхідно використати Tencent.

Tencent - це високопродуктивне обчислювальне середовище для використання нейронних мереж з відкритим вихідним кодом. Середовище було розроблено щоб бути оптимізованою для мобільних терміналів. Цей інструмент дозво-

ляє розробникам легко перенести алгоритми глибокого навчання на мобільні телефони для ефективного виконання. На даний момент середовище підтримує більшу частину алгоритмів класифікації об'єктів, наприклад:

- Classical CNN: VGG AlexNet GoogleNet Inception;
- MobileNetV2-SSDLite MobileNetV3-SSDLite;
- Detection: Faster-RCNN R-FCN …;
- Practical CNN: ResNet DenseNet SENet FPN;
- Light-weight CNN: SqueezeNet MobileNetV1/V2/V3 ShuffleNetV1/V2;
- MNasNet;
- Segmentation: FCN PSPNet UNet YOLACT;
- Pose Estimation: SimplePose;
- Face Detection: MTCNN RetinaFace;
- Detection: VGG-SSD MobileNet-SSD SqueezeNet-SSD;
- Detection: YOLOV2 YOLOV3 MobileNet-YOLOV3 YOLOV4;
- включаючи алгоритм, який був обраний у цій роботі, а саме YOLOV5.

Після того як Tencent була встановлена в середовищі розробки Android Studio, а також була скачана і встановлена модель даних. Проект було збережено та скомпільовано. На практиці взаємодія з програмою відбувається наступним чином. Користувач робить знімок або вибирає його зі своєї галереї. Програма аналізує знімок та показує результат. Наприклад візьмемо рисунок 4.2. та подивимося на результат (рисунок 4.6).

Як можна бачити вище, програма змалювала обраний рисунок, проаналізувала його та обвела знайдений об'єкт у рамку. Вище рамки можна побачити клас об'єкта, у разі це “Person”, і навіть точність передбачення, що становить 87,8 відсотків.

Плюсом використання такого підходу можна відзначити, що немає необхідності відправляти зображення на сервер, можна обробляти його безпосередньо на мобільному пристрої, це може бути корисно при поганому інтернет з'єднанні. Також немає необхідності робити компресію зображення, щоб зменшити

розмір файлу, це може значно підвищити точність виконання роботи нейронної мережі.

Рисунок 4.6 – Результат обробки фотографії на мобільному пристрої

4.3 Порівняння двох способів

Для об'єктивної оцінки двох способів необхідно вибрати критерії оцінювання. Враховуватимуться такі параметри:

– час виконання обчислень в секундах, від моменту коли зображення було вибрано, до того коли був отриманий результат;

– кількість знайдених об'єктів;

– середня точність передбачень;

– кількість хибних передбачень;

– розмір зображення.

Як фотографії для тестування вибрано 20 випадкових фотографій із сайту www.shutterstock.com. На фотографіях зображені різні об'єкти, такі як: люди, машини, тварини, предмети, їжа тощо.

Перше тестування полягає в класифікації об'єктів на зображення в оригінальному розмірі, якість картинки не змінено.

Таблиця 4.1 – Результат обробки оригінальних зображень першим способом

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
1	1.02	3	78.4	0	2.5
2	0.21	5	89.2	1	2.1
3	1.16	1	67.5	0	2.0
4	0.15	2	78.9	0	2.1
5	0.45	4	67.8	2	2.1
6	0.32	5	89.9	0	2.6
7	0.32	8	90.5	1	1.7
8	0.54	2	98.6	0	1.5
9	0.31	1	99.8	0	2.1
10	1.22	1	87.9	0	2.4
11	0.43	1	98.8	0	1.4
12	0.56	3	78.6	1	1.0
13	0.56	4	89.6	0	1.5
14	0.32	0	0	0	1.6
15	0.26	3	89.9	0	1.4
16	1.45	2	87.2	0	2.7
17	1.02	1	98.3	0	2.8
18	2.45	7	67.2	1	2.1
19	1.19	5	67.2	1	1.8
20	0.21	2	87.2	0	1.9

Таблиця 4.2 – Результат обробки оригінальних зображень другим способом

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
1	0.18	3	78.4	0	2.5
2	0.20	4	89.1	1	2.1
3	0.09	1	67.5	0	2.0
4	0.03	2	78.9	0	2.1
5	0.23	4	65.8	2	2.1
6	0.23	5	89.9	0	2.6
7	0.29	6	89.5	1	1.7
8	0.10	2	98.6	0	1.5
9	0.09	1	99.8	0	2.1
10	0.14	1	87.9	0	2.4
11	0.13	1	98.8	0	1.4
12	0.16	3	78.6	1	1.0
13	0.16	4	89.6	0	1.5
14	0.15	0	0	0	1.6
15	0.16	3	89.9	0	1.4
16	0.15	2	87.2	0	2.7
17	0.17	1	98.3	0	2.8
18	0.24	7	66.9	1	2.1
19	0.20	5	67.2	1	1.8
20	0.17	2	87.2	0	1.9

Таблиця 4.3 – Порівняння результатів тестів оригінальних зображень

Спосіб №	Середній час у секундах	Середня точність передбачень	Середня кількість помилок	Середній розмір зображень у мегабайтах
1	1.70	80.62	0.35	1.96
2	0.16	80.45	0.35	1.96

Як можна побачити в таблиці вище, перший спосіб сильно поступається другому за часом виконання. Це пов'язано з тим, що перед тим як нейронна мережа почне обробляти і класифікувати зображення, його необхідно відправити на сервер, а оригінальні зображення мають великий розмір, це сильно сповільнює весь процес. Плюс відіграє такий фактор як внутрішні процеси роботи хмарної платформи Amazon, на які неможливо вплине ззовні. На рисунку 4.7 представлений графік, на якому порівнює час виконання.

Рисунок 4.7 – Графік порівняння часу виконання

На рисунку 4.8 представлено графік точності передбачень.

Рисунок 4.8 – Графік порівняння точності передбачень

Як можна бачити на графіці, незначні відмінності, спосіб номер 2 поступається способу 1, але це не критично.

Друге тестування полягає у класифікації об'єктів на зображенні після його стиснення. Розмір зображення зменшено у 6 разів, візуально для ока людини значних змін якості картинки немає. Вхідними даними виступатимуть ті самі 20 картинок із першого тесту.

Таблиця 4.4 – Результат обробки стислих зображень першим способом

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
1	0.40	3	78.2	0	0.41
2	0.18	5	89.2	1	0.35
3	0.46	1	67.5	0	0.33
4	0.10	2	78.8	0	0.34
5	0.24	4	67.9	2	0.34
6	0.16	5	89.9	0	0.4
7	0.18	8	90.5	1	0.3
8	0.34	2	98.4	0	0.41

Продовження таблиці 4.4

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
9	0.21	1	99.7	0	0.2
10	0.42	1	87.9	0	0.17
11	0.33	1	98.8	0	0.25
12	0.44	3	78.6	1	0.27
13	0.39	4	89.6	0	0.23
14	0.25	0	0	0	0.45
15	0.21	3	89.8	0	0.47
16	0.45	2	87.2	0	0.35
17	0.49	1	98.2	0	0.3
18	1.15	7	67.2	1	0.32
19	0.51	5	67.1	1	0.47
20	0.18	2	87.1	0	0.35

Таблиця 4.5 – Результат обробки стислих зображень другим способом

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
1	0.17	3	78.4	0	0.41
2	0.21	4	89.1	1	0.35
3	0.09	1	67.3	0	0.33
4	0.03	2	78.9	0	0.34
5	0.21	4	65.9	2	0.34
6	0.23	5	89.9	0	0.4
7	0.28	6	89.5	1	0.3
8	0.10	2	98.6	0	0.41

Продовження таблиці 4.5

№	Час у секундах	Кількість об'єктів	Середня точність передбачень	Кількість помилок	Розмір зображень у мегабайтах
9	0.09	1	99.7	0	0.2
10	0.14	1	87.9	0	0.17
11	0.13	1	98.8	0	0.25
12	0.16	3	78.6	1	0.27
13	0.15	4	89.5	0	0.23
14	0.15	0	0	0	0.45
15	0.15	3	89.8	0	0.47
16	0.15	2	87.2	0	0.35
17	0.17	1	98.3	0	0.3
18	0.24	7	66.8	1	0.32
19	0.18	5	67.2	1	0.47
20	0.17	2	87.2	0	0.35

Таблиця 4.6 – Порівняння результатів тестів стислих зображень

Спосіб №	Середній час у секундах	Середня точність передбачень	Середня кількість помилок	Середній розмір зображень у мегабайтах
1	0.35	80.5	0.35	0.33
2	0.16	80.45	0.35	0.33

Як можна бачити за результатами вище, стиснення картинок допомогло зменшити кількість часу для класифікації зображень першим способом. Для другого способу істотних змін виявлено. Також стиснення зображень майже не вплинуло на точність передбачень і кількість помилок, що допускаються. Але перший спосіб так само поступається другому за часом виконання як мінімум у 2 рази. Це можна побачити на графіку зображеному рисунку 4.8.

Рисунок 4.8 – Графік порівняння часу виконання

На рисунку 4.9 ми бачимо, що точність передбачень обох способів залишається майже однаковою. Не має значення, використовуємо ми оригінальні картинки або стислі.

Рисунок 4.9 – Графік порівняння часу виконання

4.4 Висновок

У ході цього дослідження було зроблено порівняння двох різних способів взаємодії програми з нейронною мережею. Використання першого способу має на увазі, що нейронна мережа знаходиться на віддаленому хмарному просторі. Основним мінусом такого підходу є велика кількість часу очікування обробки зображення, це може бути пов'язано з низькою швидкістю інтернету, великим розміром файлу, що обробляється, або внутрішньою роботою хмарної системи, на яку не в силах вплинути сторонній розробник. Використання другого способу передбачає, що нейронна мережа знаходиться на мобільному пристрої. Основним мінусом цього підходу і те що користувача може бути пристрій з низькими системними характеристиками що у світі зустрічається дуже рідко. Було з'ясовано, що другий спосіб у два рази швидше виконує пошук та класифікацію об'єктів. Для виконання поставленого завдання у цьому проекті було прийнято рішення використати реалізацію другого підходу. У майбутньому при масштабуванні системи для поліпшення системи можна буде комбінувати обидва підходи, що дозволить перекривати мінуси один одного.

5 ПРОЕКТНИЙ РОЗДІЛ

5.1 Розробка структури проекту

5.1.1 Етапи розробки

Розробка мобільного додатка включає наступні етапи:

- аналітика;
- проектування;
- розробка;
- тестування;
- підтримка та розвиток.

Аналітика була проведена в першому розділі даної дипломної роботи, було розглянуто вже готові рішення на ринку, проаналізовано їх сильні та слабкі сторони.

Проектування полягає у виборі архітектури програми. Було прийнято рішення використовувати підхід розробки, заснований на підході чистої архітектури (Clean Architecture).

Розробка буде виконуватися в інтегрованому середовищі розробки Android Studio. Вона включає всі інструменти, які можуть знадобитися в процесі розробки, такі як: візуальний редактор інтерфейсу, вбудований термінал для роботи з системою контролю версій, автоматичний збирач файлів та можливість роботи з базою даних.

Тестування буде виконане з використанням unit тестів для перевірки роботи з API а також буде проведено мануальне тестування інтерфейсу користувача, для пошуку багів.

Підтримка та розвиток полягає у швидкому реагуванні на фідбек користувачів для виконання hot фіксів.

5.1.2 Чиста Архітектура (Clean Architecture)

Clean Architecture об'єднала в собі ідеї кількох інших архітектурних підходів, які сходяться на тому, що архітектура повинна:

- не залежати від UI;
- не залежати від БД, зовнішніх фреймворків та бібліотек;
- бути тестованою.

Такий підхід може бути досягнутий за рахунок поділу програми на шари, а також дотриманням правила залежностей (Dependency Rule).

Dependency Rule закликає до того, щоб внутрішні шари ніколи не повинні залежати від зовнішніх. Вся бізнес-логіка і логіка програми не в жодному разі не залежати від інтерфейсу користувача, презентерів, баз даних тощо, це правило зображено стрілками, що вказують всередину на рисунку 5.1 [15].

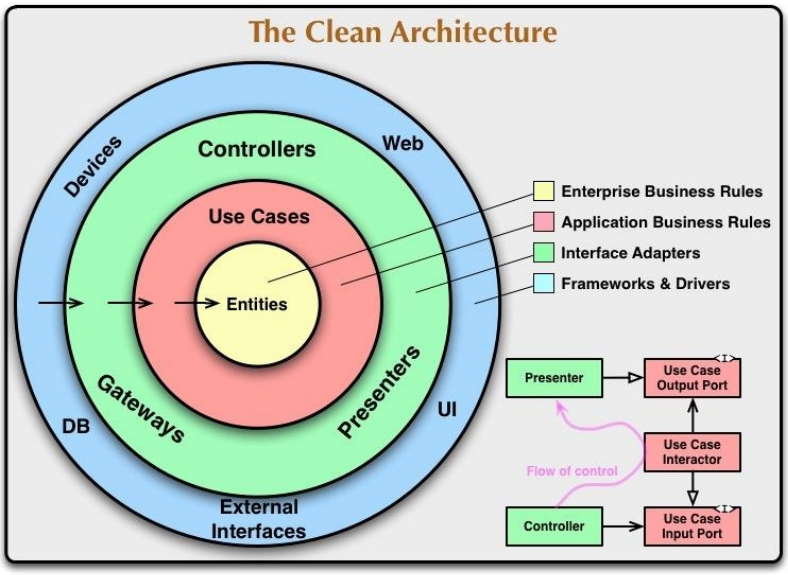

Рисунок 5.1 – Схема Clean Architecture

Імена сутностей (функцій, класів, змінних, чого завгодно), які були оголошені у зовнішніх шарах, не повинні зустрічатися в коді внутрішніх верств. Цей підхід дозволяє будувати системи, які буде простіше підтримувати, тому що зміни у зовнішніх шарах не торкнуться внутрішніх шарів [15].

В основному виділяють 4 шари (рисунок 5.2):

– Entities - бізнес-логіка, загальна для багатьох програм;

– Use Cases (Interactors) - логіка програми;

– Interface Adapters – адаптери між Use Cases та зовнішнім світом. Тут можуть використовуватися Presenter'и з патерну MVP, а також Gateways (популярніша назва репозиторії);

– Frameworks - зовнішній шар, тут лежить все інше (інтерфейс користувача, база даних, http-клієнт тощо).

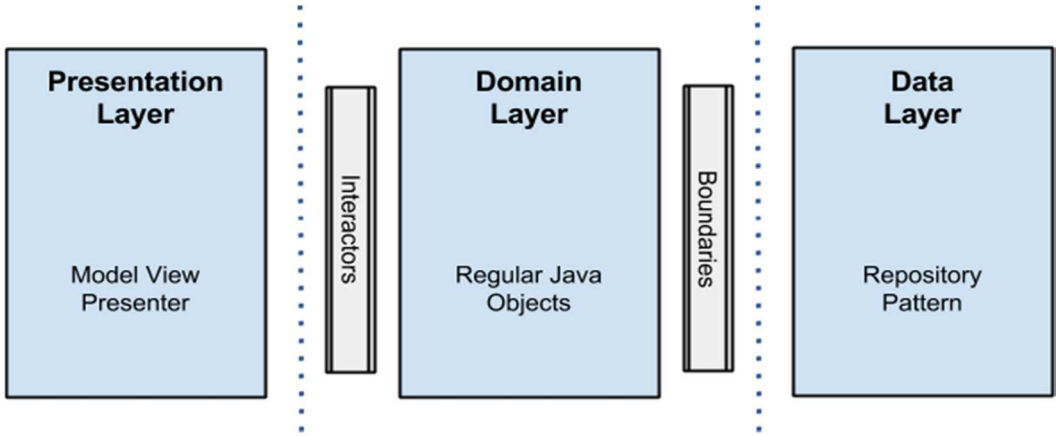

Рисунок 5.2 – Схема шарів

Перехід між шарами здійснюється через Boundaries, тобто через два інтерфейси: один для запиту і один для відповіді. Вони потрібні, щоб внутрішній шар не залежав від зовнішнього (дотримуючись Dependency Rule), але при цьому міг передати йому дані [15].

Коли чиста архітектура застосовується для розробки мобільного додатка, вона має трохи інший вигляд, який відрізняється від оригіналу (рисунок 5.3). Кольори позначають шари. А стрілка внизу означає Dependency Rule [15].

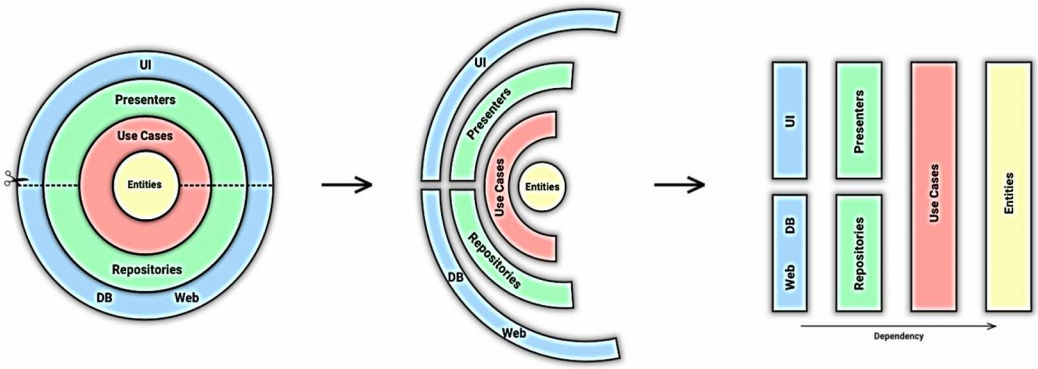

Рисунок 5.3 – Чиста архітектура для мобільного додатка

На рисунку 5.4 можна побачити, до якої категорії належить кожен шар.

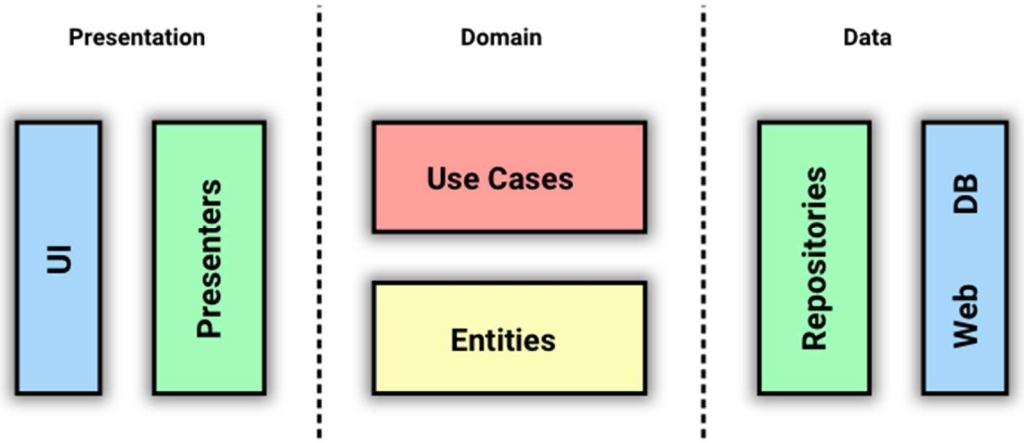

Рисунок 5.4 – Шари, розташовані за категоріями

Для наочності представлення роботи програми з даними можна подивитися на рисунок 5.5. На схемі, що вийшла, можна наочно побачити перебіг даних, яке позначено червоними стрілками. Дані йдуть від інтерфейсу користувача до бази даних або сервера і назад.

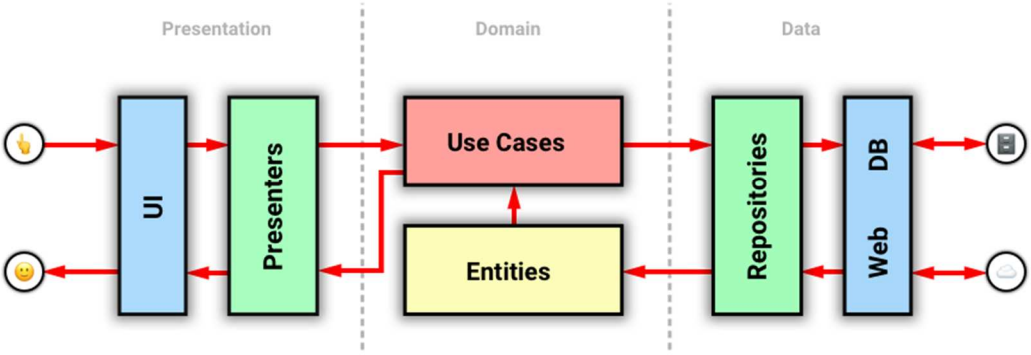

Рисунок 5.5 – Перебіг даних у Clean Architecture

Подія користувача йде в Presenter, який у свою чергу передає в Use Case. Use Case робить запит у Repository. Repository отримує дані і потім створює Entity, передає його в Use Case. Так Use Case отримує всі необхідні йому Entity. Потім, застосувавши їх та свою логіку, отримує результат, який передає назад у Presenter. А той, своєю чергою, відображає результат в UI. На переходах між шарами використовують Boundaries, описані раніше [15].

5.1.3 Паттерн MVVM

Було прийнято вибудовувати інтерфейс користувача з використанням патерна MVVM. З MVVM (Model-View-ViewModel) процес розробки графічного інтерфейсу користувачів ділиться на дві частини. Перша частина полягає у роботі з мовою розмітки або кодом GUI. Друга це розробка бізнес-логіки або логіки бекенда (модель даних). Частина View model у MVVM – це конвертер значень. Це означає, що viewmodel відповідає за конвертування об'єктів даних з моделі в такий вид, щоб з об'єктами було легко працювати. Якщо дивитися з такої сторони, то viewmodel це швидше модель, ніж уявлення. Вона контролює більшу частину логіки відображення. Модель вистави може реалізовувати патерн медіатор. Для цього організується доступ до логіки бекенд навколо набору юз-кейсів, що підтримуються поданням (рисунок 5.6).

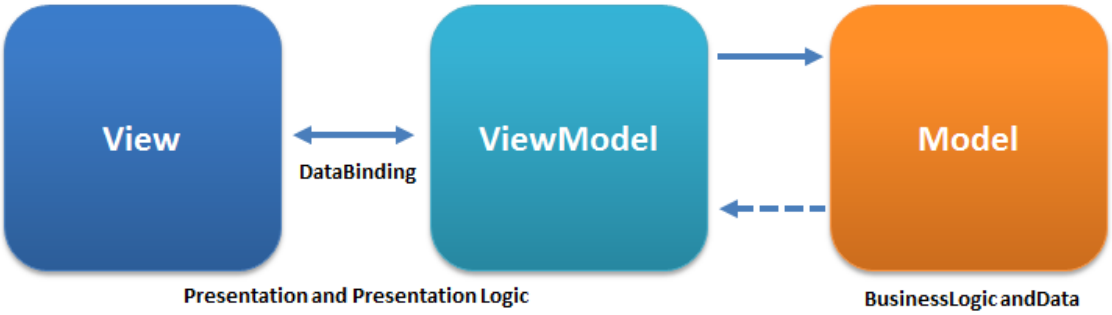

Рисунок 5.6 – Паттерн MVVM

Архітектуру MVVM можна розділити на три різні частини:

– уявлення – зберігає структурне визначення того, що користувачі можуть бачити на своїх екранах. Сюди може бути поміщений статичний та динамічний вміст (анімація, зміна стану та будь-яка робота з UI). Тут не повинно бути жодної бізнес-логіки програми. В android тут знаходиться активність або фрагмент [16];

– модель вистави – компонент, який пов'язує модель і виставу. Він відповідає за керування посиланнями даних та можливих конверсій. Тут з'являється

біндінг. В Android немає необхідності турбуватися про це, тому що можна безпосередньо використовувати клас AndroidViewModel або ViewModel [16];

– модель – рівень бізнес-даних, він не пов'язаний з жодним особливим графічним уявленням. В Android, згідно з Clean Architecture, модель може містити клас бізнес-логіки, базу даних, репозиторій. Взаємодія між компонентами можна побачити рисунку 5.7 [16].

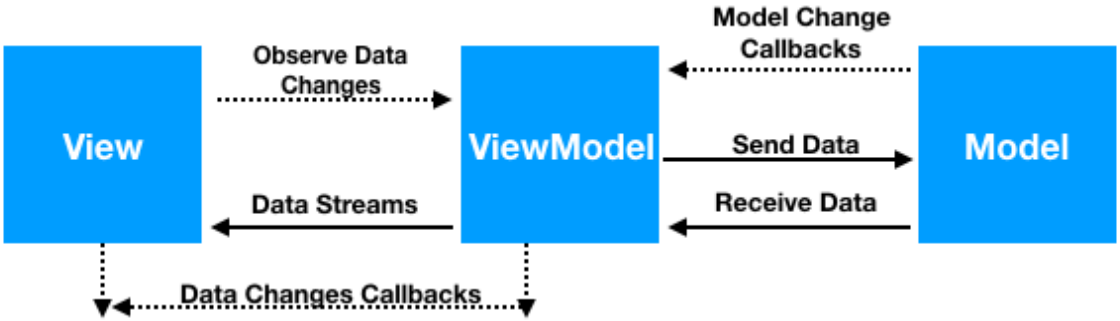

Рисунок 5.7 – Взаємодія компонентів MVVM

Завдяки використанню такої архітектури програми, його можна легко підтримувати та тестувати.

5.2 Розробка Бази даних

5.2.1 Підключення локальної бази даних

Android надає відразу кілька способів підключення локальної бази даних SQLite. Перший спосіб полягає в успадкування кожної Activity від ORMLiteBaseActivity. Це дозволяє не стежити за життєвим циклом з'єднання з базою даних. Мінус такого підходу в тому, що немає можливості отримати доступ до інших класів програми. Другий спосіб пропонує вирішення цієї проблеми, необхідно створити клас, який ініціалізуватиме помічника для створення базою даних та роботи з нею. Звернення до нього буде виконуватися на самому початку та наприкінці виконання життєвого циклу мобільного додатка.

5.2.2 Підключення Firebase Realtime Database

Підключення до віддаленої бази даних, яка працює в реальному часі виконується за кілька кроків, необхідно:

– підключити відповідні бібліотеки до конфігураційних файлів збирача проекту Gradle;

– на сайті firebase створити конфігураційний файл бази даних та імпортувати його в проект;

– отримати інстанс класу FirebaseDatabase для подальшої роботи.

5.3 Розробка підходів до масштабування

Планувати масштабування програми від початку її розробки – це чудова бізнес-стратегія. Додаток необхідно створювати таким чином, щоб припускати, що він може пропускати через себе тисячі та мільйони користувачів. Ось деякі питання, на які потрібно відповісти одразу:

– в середньому, скільки людей будуть використовувати додаток у найближчі 6-12 місяців?;

– які дані генерує користувач?;

– яким є приблизний часовий діапазон для розміщення всіх клієнтів на одному сервері?

Важливу роль масштабування відіграє наявність архітектури в додатку, це дозволяє легко підтримувати проект у належному стані, а також дає можливість проводити оптимізацію без особливих зусиль. Насамперед варто проаналізувати проект і подумати, а чи вже вдалося оптимізувати. Потім потрібно перевірити, чи виконуються оптимальні запити до БД (аналіз EXPLAIN, використання індексів). Виконати аналіз чи правильно зберігаються дані у базі. Аналіз використання у додатку кешування даних. А також виконати перевірку чи оптимальні алгоритми обробки даних та налаштування оточення [17].

Після того, як оптимізація проведена, можна подумати про рознесення бекенд частини по кількох хостах, з метою збільшення загальної продуктивності програми за рахунок збільшення доступних ресурсів. Такий підхід має офіційну назву – "масштабування" (scale) програми. Точніше, під «масштабованістю» (scalability) називається можливість системи збільшувати свою продуктивність зі збільшенням кількості виділених їй ресурсів. Розрізняють два способи масштабування: вертикальне та горизонтальне. Вертикальне масштабування передбачає збільшення продуктивності програми при додаванні ресурсів (процесора, пам'яті, диска) у межах одного вузла (хоста). Горизонтальне масштабування притаманно розподілених додатків і має на увазі зростання продуктивності докладання при додаванні ще одного вузла (хоста) [17].

Важливо те, що в проекті як база даних використовується Firebase Realtime Datebase. Ця база даних знаходиться на серверах Google. Google надає всі інструменти масштабування, а за окрему плату можна делегувати це завдання безпосередньо Google.

6 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

Коли користувач натискає на іконку програми у головному меню свого телефону, він потрапляє на екран авторизації. У цьому додатку спрощена система реєстрації, немає необхідності вводити величезну кількість інформації, так як за статистикою, складна реєстрація відштовхує користувачів і вони не використовують програму зовсім. Для авторизації необхідно ввести номер телефону (рисунок 6.1).

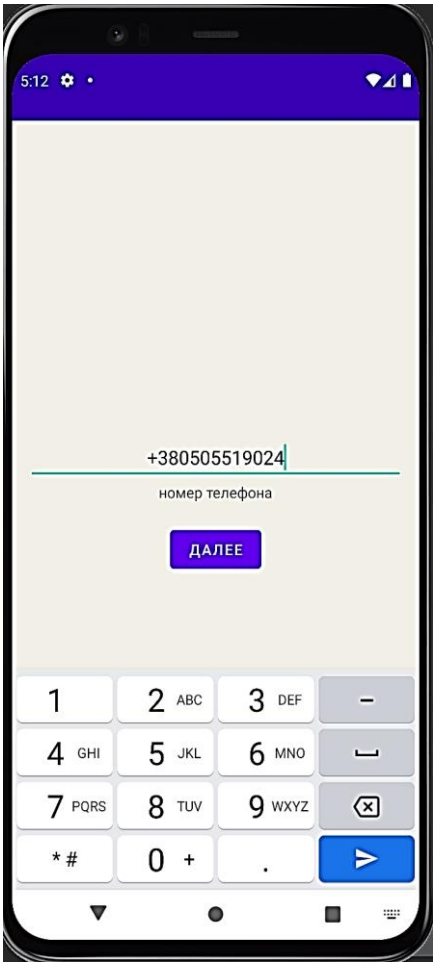

Рисунок 6.1 – Авторизація за номером телефону

На телефон прийде смс повідомлення з шестицифровим числом, це код авторизації, який потрібно ввести в поле, на екрані, на який перенаправило після введення номера телефону (рисунок 6.2).

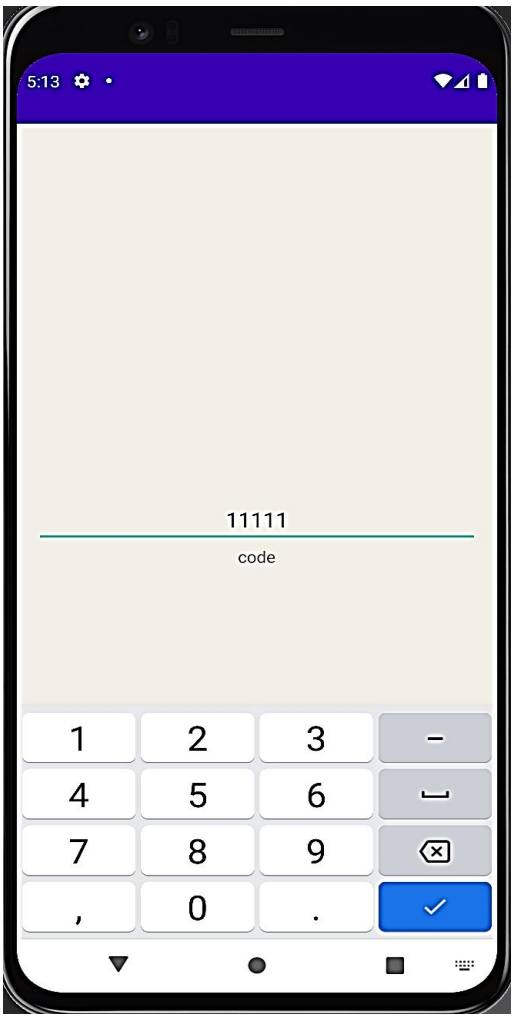

Рисунок 6.2 – Авторизація – введення коду

Якщо ввести неправильний номер телефону або невірний код авторизації, на екрані виявиться повідомлення, яке попереджає про помилку (рисунок 6.3).

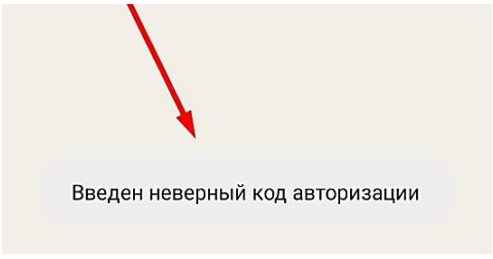

Рисунок 6.3 – Введення некоректних даних

Після успішної авторизації програма перенаправляє на екран профілю (рисунок 6.4).

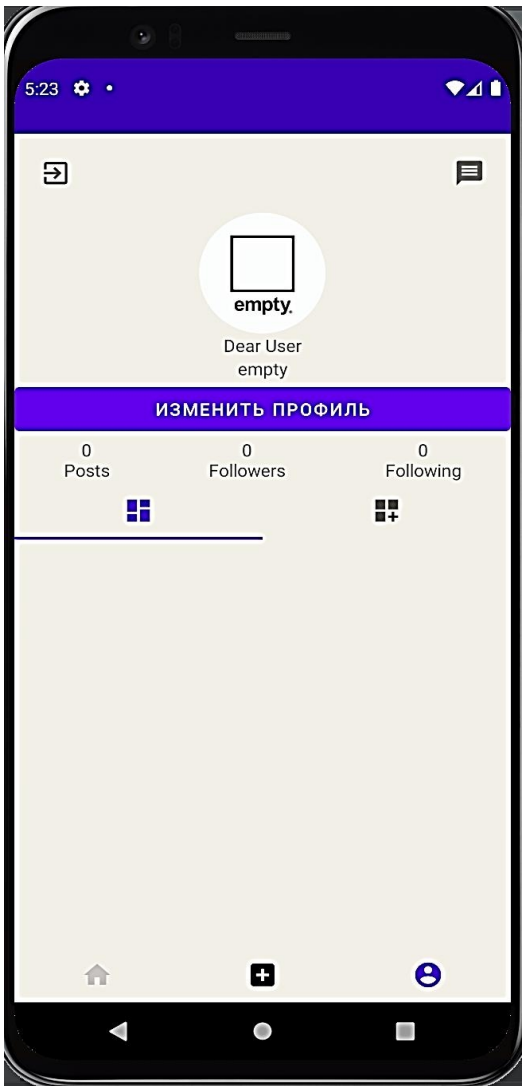

Рисунок 6.5 – Профіль

Вгорі екрана розташовані дві кнопки навігації. Кнопка зліва дозволяє вийти з поточного облікового запису. Кнопка права перенаправляє на екран надсилання повідомлень (рисунок 6.6).

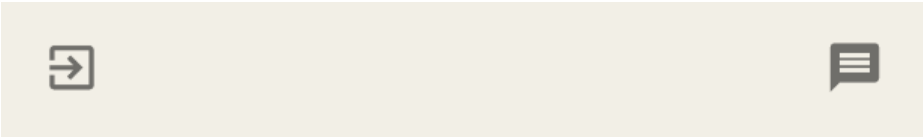

Рисунок 6.6 – Верхня навігація

Знизу екрана розташовані інші навігаційні кнопки. Перша кнопка дозволяє перейти на домашню сторінку. Друга кнопка дозволяє зробити завантаження фотографії у програму. Третя кнопка – це перехід на сторінку профілю. Фіолетовим

кольором підсвічується вкладка, яка активна в даний момент (рисунок 6.7).

Рисунок 6.7 – Нижня навігація

Для того щоб змінити аватар і додати інформацію про себе, необхідно натиснути кнопку "Змінити профіль" (рисунок 6.8).

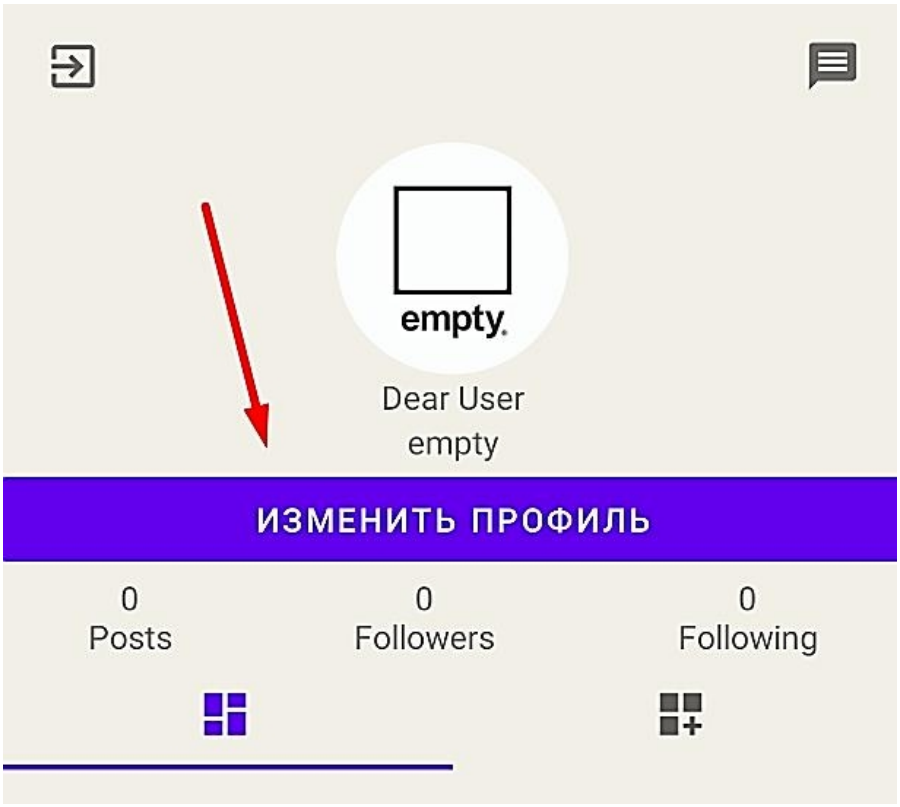

Рисунок 6.8 – Кнопка редагування профілю

Відкриється вікно редагування профілю, зображене на рисунку 6.9.

Для того щоб встановити аватар користувача необхідно натиснути на фотографію, або на напис "Змінити аватар" (рисунок 6.10).

Рисунок 6.9 – Редагування профілю

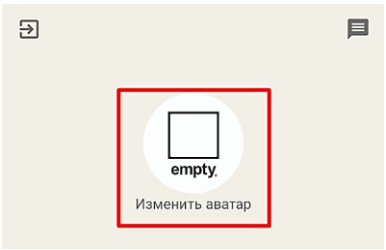

Рисунок 6.10 – Кнопка завантаження аватара

Знизу екрана вилізе діалогове вікно, там будуть всі фотографії, які є на телефоні. Для того, щоб встановити фотографії, потрібно просто натиснути на неї (рисунок 6.11).

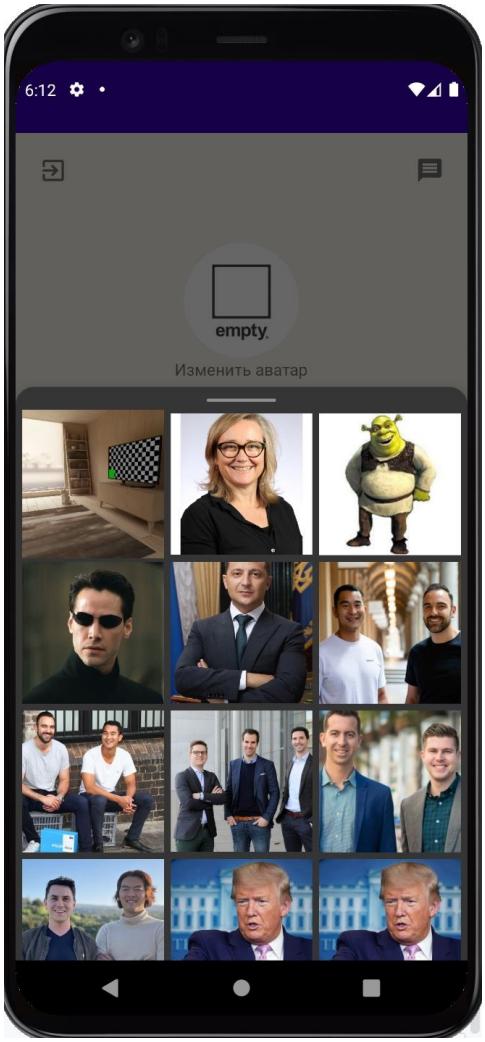

Рисунок 6.11 – Вибір фотографії

Якщо потрібно зробити нову фотографію для аватара прямо зараз, потрібно натиснути на перший квадрат, це камера (рисунок 6.12).

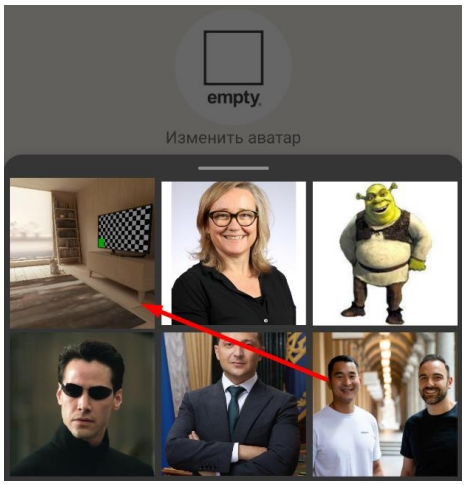

Рисунок 6.13 – Живий знімок

Після того як відкрився екран камери для того, щоб зробити знімок з фронтальної камери потрібно натиснути кнопку на якій намальована камера зі стрілками і після цього натиснути кнопку в центрі екрана (рисунок 6.14).

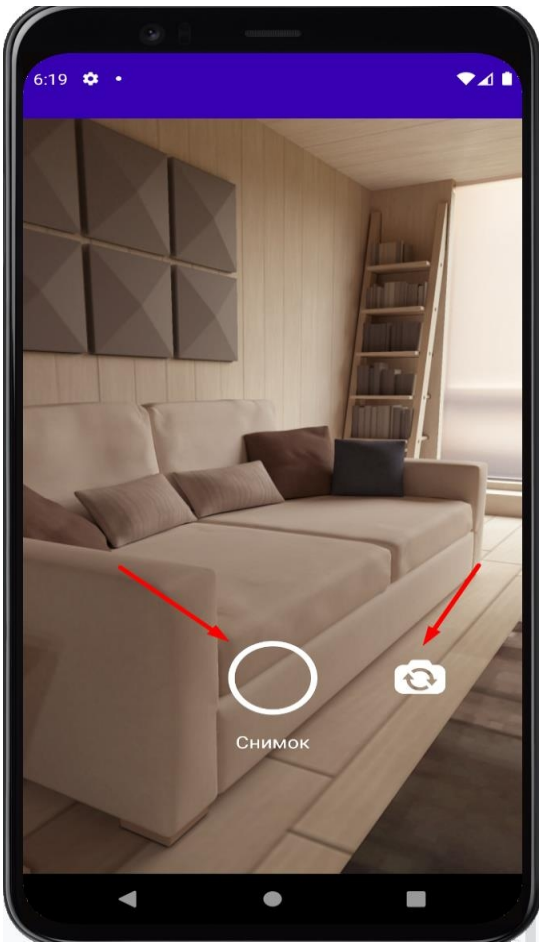

Рисунок 6.14 – Екран камери

Після того як вся необхідна інформація була заповнена для того, щоб вийти з екрана редагування профілю, необхідно натиснути кнопку з галочкою знизу екрана. На неї також можна натиснути для виходу, якщо інформація не була змінена. Вона як кнопка “назад” та кнопка “зберегти” (рисунок 6.15).

Щоб опублікувати пост, необхідно натиснути кнопку по центру, внизу екрана (рисунок 6.16).

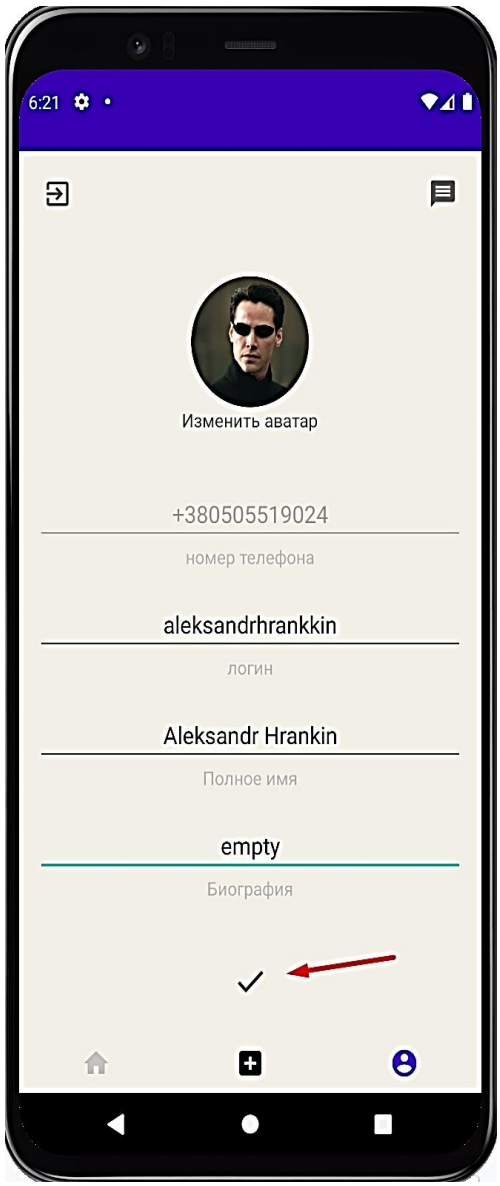

Рисунок 6.15 – Збереження інформації профілю

Рисунок 6.16 – Додавання посту

Відкриється вікно з фотографією, щоб вибрати фотографію, потрібно просто натиснути на неї. Можна вибирати кілька фотографій за один раз. Щоб продовжити потрібно натиснути кнопку з галочкою знизу екрана (рисунок 6.17).

Рисунок 6.17 – Вибір фотографій для посту

Після того як нейромережа проаналізувала фотографію, визначила на ній людей і зберегла дані, можна додати будь-який опис до посту, довжина опису не може перевищувати допустиму межу 255 символів. Далі натисніть кнопку “Зберегти” (рисунок 6.18).

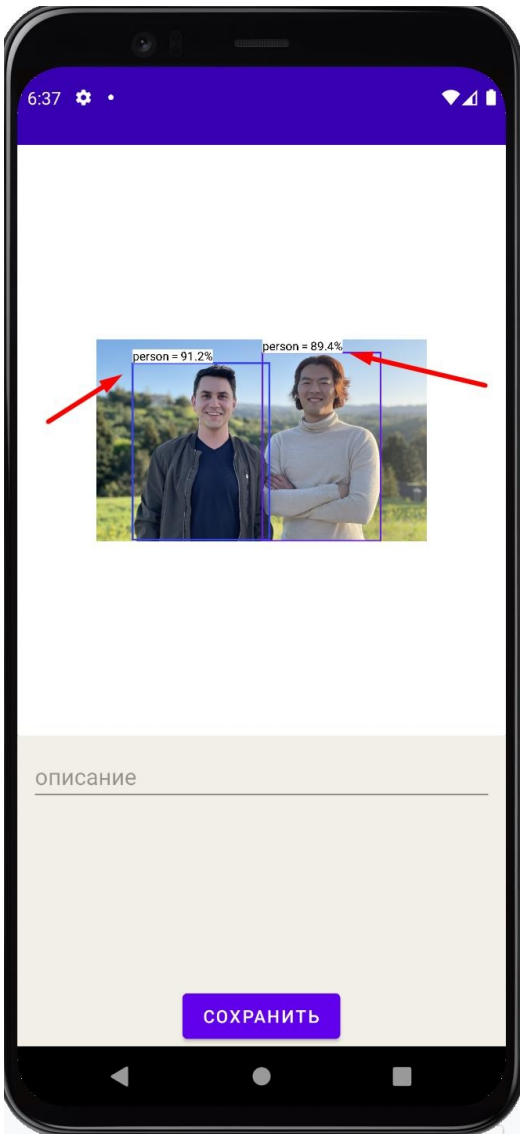

Рисунок 6.18 – Аналіз фотографії

На сторінці профілю можна побачити, що з'явився новий пост, а також лічильник постів збільшився. Усі пости будуть збережені тут у спеціальний контейнер постів. Їх можуть переглядати інші користувачі соціальної мережі, які відвідують цю сторінку. Щоб подивитися пост повністю, необхідно натиснути нього (рисунок 6.19).

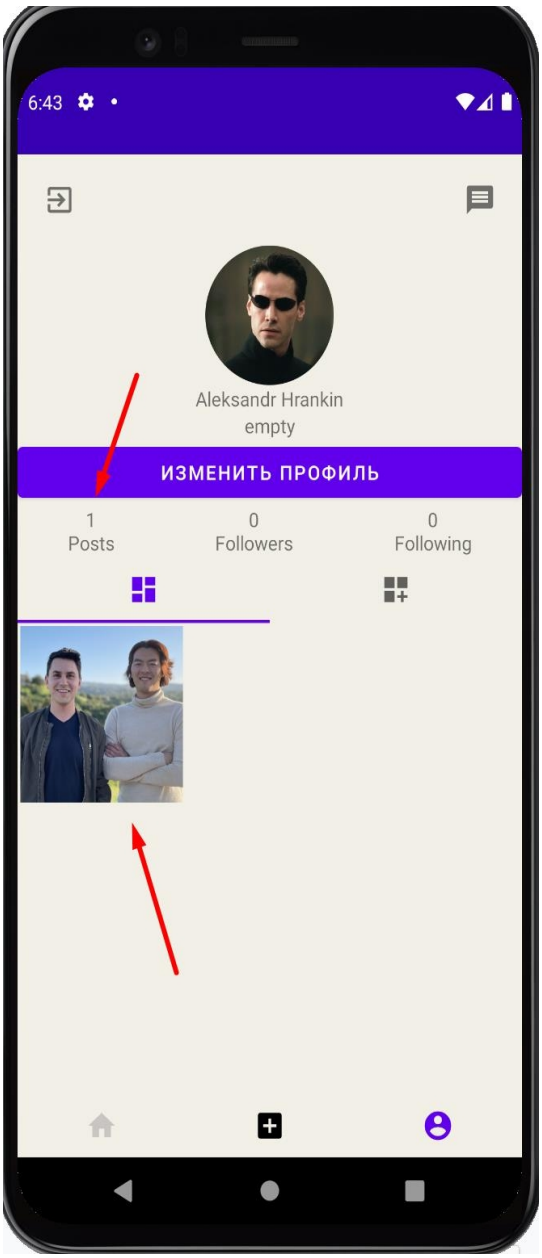

Рисунок 6.19 – Контейнер постів

Після того як був натиснути пост, відкривається вікно, на якому зображена фотографія. На ній можна бачити, що біля кожної людини є кнопка лайка. Будь-який користувач може поставити 1 лайк біля кожної людини на фото (рисунок 6.20).

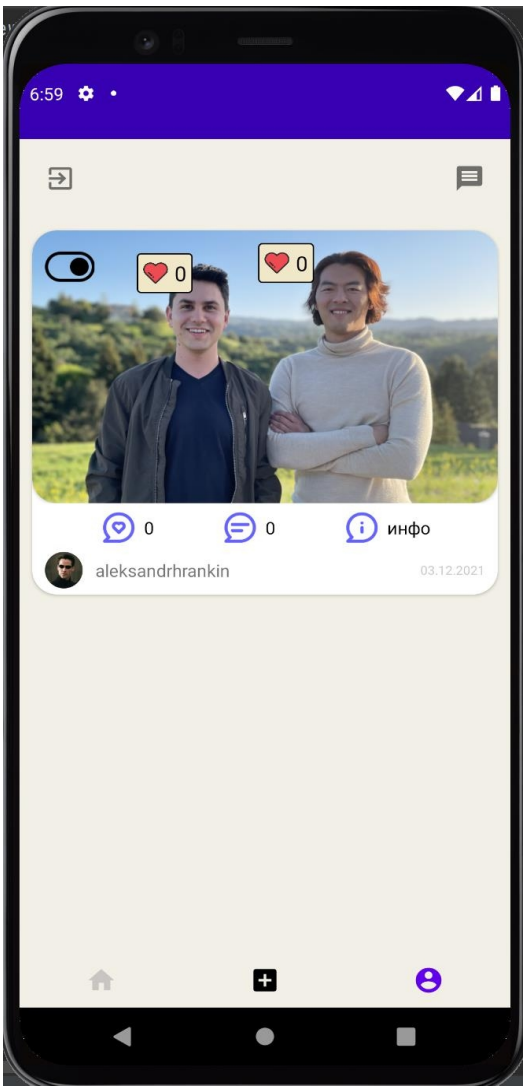

Рисунок 6.20 – Екран з постом

Під фотографією можна бачити панель із кнопка. Перша кнопка дозволяє поставити лайк для посту в цілому. Друга кнопка відкриває екран із коментарями. Третя кнопка показує загальну інформацію про фотографію. Нижче кнопок відображається аватар автора поста, логін автора та дата публікації поста (рисунок 6.21).

Рисунок 6.22 – Панель кнопок посту

Для того щоб кнопки лайків не заважали переглянути фотографію, можна натиснути кнопку вимкнення відображення кнопок, вона розташована у верхньому лівому кутку (рисунок 6.23).

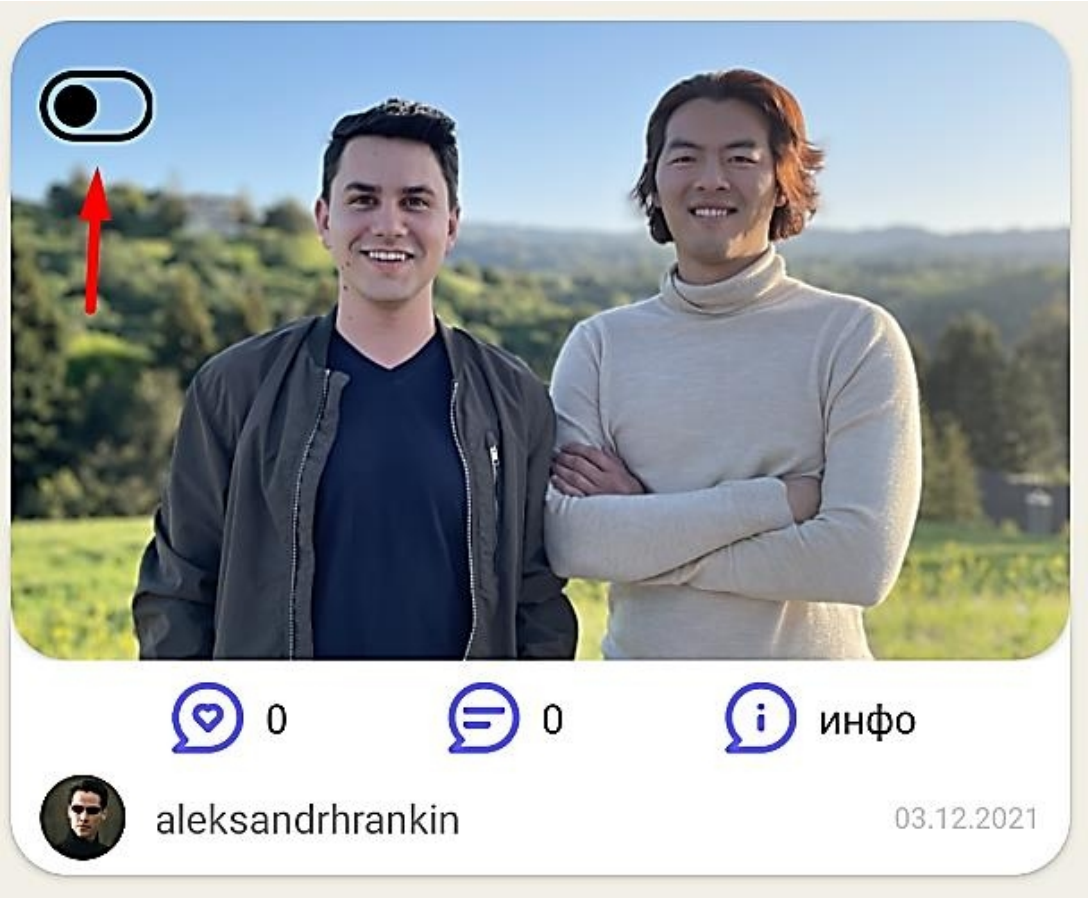

Рисунок 6.24 – Вимикач відображення кнопок лайків

Щоб переглянути публікації інших користувачів, необхідно натиснути на вкладку HomePage. Це перша кнопка внизу екрана на панелі навігації. Якщо зараз користувач ні на кого не підписано, то тут будуть посади всіх людей, які були зареєстровані у соціальній мережі. Якщо користувач підписано на когось, то в першу чергу будуть показані нові пости тих людей, на кого він підписаний (рисунок 6.25).

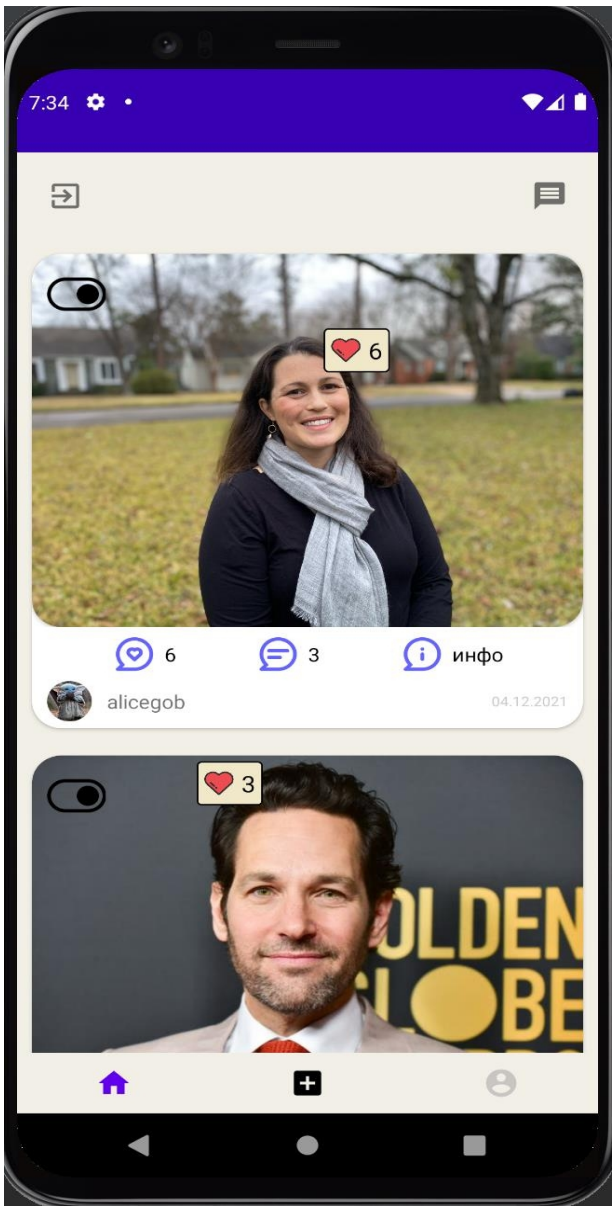

Рисунок 6.25 – Стрічка новин на сторінці Home

Якщо потрібно залишити коментар потрібно натиснути кнопку з коментарями, програма перенаправить на екран з коментарями. Тут можна почитати які коментарі залишили інші користувачі і залишити свій (рисунок 6.26).

Рисунок 6.26 – Екран з коментарями

Також на цьому екрані можна почитати опис посту, щоб детальніше розібрати і зрозуміти що саме хотів донести автор.

ВИСНОВКИ

Мета кваліфікаційної роботи магістра була у створенні соціальної мережі з використанням нейронної мережі для аналізу контенту та переробки поточної системи оцінювання контенту.

У першому розділі роботи розглядалися існуючі соціальні мережі, їх робота та дизайн. Були проаналізовані та враховані їхні недоліки, а саме:

– вузькоспрямованість;

– оцінювання контенту проводиться застарілим способом, коли оцінка ставиться для посту, а не конкретної його складової.

У дипломній роботі було розглянуто нейронні мережі. принцип їхньої роботи, навчені моделі даних, а також сімейство алгоритмів класифікації об'єктів YOLO.

Провівши аналіз технологій для реалізації соціальної мережі було вибрано такі технології:

– технологія Firebase Realtime Database для хмарного зберігання даних та швидкого доступу до них;

– алгоритм класифікації об'єктів YOLOv5;

– Android Studio – як середовище розробки мобільного продукту.

У процесі роботи над дипломним проектом було розроблено мобільний додаток який включає використання нейронної мережі для аналізу зображень з метою класифікації об'єктів. Програма пройшла багато тестувань на емуляторі Android Studio, і на реальних пристроях. Тестування показали, що нейронна мережа працює коректно.

Розроблений програмний продукт відповідає вимогам, сформульованим у технічному завданні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Штучні нейронні мережі [Електронний ресурс] – Режим доступу: https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins (дата звернення: 02.09.2021).

2. Нейронна мережа (Neural network) [Електронний ресурс] – Режим доступу: https://wiki.loginom.ru/articles/neural-network.html (дата звернення: 02.09.2021).

3. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Instagram (дата звернення: 02.09.2021).

4. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/wiki/Kotlin (дата звернення: 18.09.2021).

5. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/wiki/Java (дата звернення: 30.09.2021).

6. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/wiki/Python (дата звернення: 02.10.2021).

7. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/ wiki/Gradle (дата звернення: 02.10.2021).

8. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/ wiki/XML (дата звернення: 15.10.2021).

9. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/ wiki/SQLite (дата звернення: 15.10.2021).

10. База даних Firebase у реальному часі [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/database (дата звернення: 16.10.2021).

11. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/wiki/Android_Studio (дата звернення: 16.10.2021).

12. Вікіпедія. [Електронний ресурс] – Режим доступу: https://ua.wikipedia.org/ wiki/PyCharm (дата звернення: 25.10.2021).

13. Використання Retrofit 2.x як REST клієнта [Електронний ресурс] – Режим доступу: https://habr.com/ua/post/428736/ (дата звернення: 26.10.2021).

14. Glide [Електронний ресурс] – Режим доступу: http://developer.alexanderklimov.ru/android/library/glide.php (дата звернення: 27.10.2021).

15. Clean Architecture [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/mobileup/blog/335382/ (дата звернення: 30.10.2021).

16. MVVM на Android с компонентами архитектуры [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/mvvm (дата звернення: 01.11.2021).

17. Масштабирование нагрузки [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/113992/ (дата звернення: 01.11.2021).