Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Ніколайчука Ігоря Олександровича*
	(ПІБ)
академічної групи	*121М-20-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка web-застосунку автоматичного тегування та категоризації зображень на основі згорткових нейронних мереж*

_____*І.О. Ніколайчук*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Якунін А.О.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент	*Проф. Байбуз О.Г.*			
Нормоконтролер	*Доц. Приходченко С.Д.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 21 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-20-1* *Ніколайчуку Ігорю Олександровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка web-застосунку автоматичного тегування та категоризації зображень на основі згорткових нейронних мереж*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від ___.___.2021 р. № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процеси автоматичного тегування та кластеризації зображень.

Предмет досліджень – використання існуючих програмних пакетів для автоматичного тегування та кластеризації зображень.

Методи дослідження: нейромережеві методи категорізації на основі згорткових нейронних мереж, використання існуючих програмних пакетів для обробки мовного сигналу, розробка програмного забезпечення модифікації мовного сигналу.

Мета роботи – розробити та реалізувати алгоритми тегування та категоризації на основі згорткових нейронних мереж та визначення візуальної схожості зображень дослідження методів.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що запропоновано новий алгоритм для автоматичного тегування та кластеризації зображень на основі згорткових нейронних мереж.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм для автоматичного тегування та кластеризації зображень на основі згорткових нейронних мереж.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Згорткові нейронні мережі зазвичай мають велику кількість нейронів та шарів. Навчання таких мереж потребує значних обчислювальних потужностей та затрат часу. За останні роки з'явилася велика кількість моделей створених і навчених професіоналами з використанням великої кількості даних і великих обчислювальних потужностей. Багато з цих моделей знаходяться у відкритому доступі і будь-хто може використовувати їх для вирішення своїх завдань абсолютно безкоштовно.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз джерел та постановка задачі	12.09.2021 - 30.09.2021
Побудова математичних моделей та розробка алгоритму	01.10.2021 - 31.10.2021
Розробка та тестування програмного забезпечення для автоматичного тегування та кластеризації зображень на основі згорткових нейронних мереж	01.11.2021 - 30.12.2021

Завдання видав _____ *Якунін А.О.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Ніколайчук І.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *20.01.2021*

РЕФЕРАТ

Пояснювальна записка: ___ стор., ___ рис., ___ таблиці, ___ додатка, ___ джерел.

Об’єкт досліджень – процеси автоматичного тегування та кластеризації зображень.

Предмет досліджень – використання існуючих програмних пакетів для автоматичного тегування та кластеризації зображень.

Методи дослідження: нейромережеві методи категорізації на основі згорткових нейронних мереж, використання існуючих програмних пакетів для обробки мовного сигналу, розробка програмного забезпечення модифікації мовного сигналу.

Мета роботи – розробити та реалізувати алгоритми тегування та категоризації на основі згорткових нейронних мереж та визначення візуальної схожості зображень дослідження методів.

Новизна запропонованих рішень визначається тим, що запропоновано новий алгоритм для автоматичного тегування та кластеризації зображень на основі згорткових нейронних мереж.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм для автоматичного тегування та кластеризації зображень на основі згорткових нейронних мереж.

Згорткові нейронні мережі зазвичай мають велику кількість нейронів та шарів. Навчання таких мереж потребує значних обчислювальних потужностей та затрат часу. За останні роки з'явилася велика кількість моделей створених і навчених професіоналами з використанням великої кількості даних і великих обчислювальних потужностей. Багато з цих моделей знаходяться у відкритому доступі і будь-хто може використовувати їх для вирішення своїх завдань абсолютно безкоштовно.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, АВТОМАТИЧНЕ ТЕГУВАННЯ, КЛАСТЕРИЗАЦІЯ, АСАМБЛЕВИЙ НЕЙРОМЕРЕЖЕВИЙ ПІДХІД

ABSTRACT

Explanatory note: ___ pages, ___ figures, ___ tables, ___ appendices, ___ sources.

The object of research is the processes of automatic tagging and clustering of images.

The subject of research is the use of existing software packages for automatic tagging and clustering of images.

Research methods: neural network categorization methods based on convolutional neural networks, use of existing software packages for speech signal processing, development of speech signal modification software.

The aim of the work is to develop and implement tagging and categorization algorithms based on convolutional neural networks and to determine the visual similarity of image research methods.

The novelty of the proposed solutions is determined by the fact that a new algorithm for automatic tagging and clustering of images based on convolutional neural networks.

The practical value of the results is that as a result of the study, an algorithm was designed for automatic tagging and clustering of images based on convolutional neural networks.

Convolutional neural networks usually have a large number of neurons and layers. Learning such networks requires significant computing power and time. In recent years, there have been a large number of models created and trained by professionals using large amounts of data and large computing power. Many of these models are publicly available and anyone can use them to solve their problems for free.

In the section "Economics" calculations of the complexity of software development, the cost of creating software and the duration of its development, as well as marketing research of the market for the software product.

List of key words: CONVOLVED NEURAL NETWORKS, AUTOMATIC TAGING, CLUSTERIZATION, ASSEMBLY NEURAL NETWORK APPROACH

ЗМІСТ

ВСТУП

Останні роки відбувається стрімкий рост кількості мобільних пристроїв (наприклад, смартфонів, цифрових камер тощо) та послуг хмарного зберігання інформації, що призвело до безпрецедентного зростання кількості персональних медіа-ресурсів, таких як зображення. Наприклад, люди знімають фотографії на смартфони кожен день і скрізь. Повідомляється, що Flickr має 1,7 мільйона фотографій, завантажених щодня, а Instagram претендує на 140 мільйонів фотографій щодня в 2021 році. Така велика кількість зображень вимагає ефективного доступу до них. Один з найбільш ефективних способів пошуку зображень – через текстові мітки.

Особливо актуальна дана тема власникам великих об’ємів зображень. Одними з них є сайти-фотостоки, на які люди завантажують безліч зображень. Для кожного з них необхідно власноруч проставити текстові мітки, які його характеризують. Дана операція знижує бажання долі клієнтів завантажувати ще більше фото, адже для кожного потрібно буде ставити теги. Дану операцію можна і потрібно автоматизувати. Одним з рішень є тегування зображень з допомогою згорткових нейронних мереж.

Кваліфікаційна робота складається з трьох розділів. Перший розділ присвячено загальному опису потреби автоматичного тегування зображень. У другому розділі описано математичний апарат та алгоритмізація методів тегування зображень на основі згорткових нейронних мереж. У третьому розділі представлено обґрунтування вибору використаних технологій, характерні особливості, архітектура розробленого програмного забезпечення, аналіз якості роботи систем та аналіз отриманих результатів.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ АВТОМАТИЧНОГО ТЕГУВАННЯ ЗОБРАЖЕНЬ

1.1. Актуальність теми дослідження.

Фотосток (photostock) або фотобанк – це компанія, яка надає на продаж графічні матеріали (контент), наприклад: фотографії, малюнки, ілюстрації [1]. Тобто за аналогією банк, stock (ринок) – місце зберігання або реалізації, в даному випадку фотоматеріалів. Тобто фотосток (фотобанк) – це посередник, між власником контенту (фотографом, ілюстратором) і покупцем.

Власник контенту (фотограф, ілюстратор) поставляє свої роботи в фотосток (фотобанк). Фотосток забезпечує залучення покупців, рекламу і механізми зберігання, пошуку, продажу та прийому платежів від покупців і відповідно утримує комісію за свої послуги. А з кожного продажу виплачує авторські відрахування (royalty) автору.

Дизайнер або інша заінтересована особа можуть звернутися в фотобанк, якщо їм необхідні якісь матеріали. Найчастіше – це елементи для підготовки дизайну, візуальний образ для оформлення статті в друкованому виданні або інтернет сайті, образ для оформлення презентації, різноманітна рекламна продукція.

Основними постачальниками контенту фотостоках є фотографи і ілюстратори. Відповідно основне наповнення складається з фотографій і векторних ілюстрацій.

Фотобанки діляться на традиційні (макростоки) і мікропейментові (мікростоки).

Традиційні фотобанки (макростоки) – великі регіональні або світові фотобанки з історією. З появою і розвитком інтернету макростоки, так само отримали свої представництва і там. Відрізняються великою базою фотографій,

зокрема, набагато ширше представлена репортажна фотографія. І в порівнянні з мікростоках – типом ліцензування. Найбільш відомі представники: Getty Images, Corbis, Alamy.

Традиційні фотобанки в основному продають фотографії за ліцензією Rights Managed (RM). З цієї ліцензії фотографія може продаватися тільки в одному фотобанку, а так само можливе послідовне використання фотографії різними покупцями. При покупці по RM ліцензії покупець повинен надавати інформацію про деталі використання фотографії. Купуючи фотографію за такою ліцензією, покупець може бути впевнений, що вона не використовувалася його конкурентами і не буде використовуватися кимось ще в цей же час. Вартість ліцензії розраховується виходячи з розміру зображення, виду і тривалості використання, а також його географії, і становить зазвичай від десятків до сотень доларів.

Зображення за ліцензією RM купуються для одноразового використання і на певний термін. Існує також можливість ексклюзивної покупки прав на зображення за ліцензією Rights Managed.

Мікропейментові фотобанки (мікростоки) – інтернет фотобанки з не настільки великим портфелем контенту, як у макростоків, з'явилися в 2000-х роках, у відповідь на попит на зображення за доступною ціною, коли вартість традиційних ліцензій RM категорично не підходила покупцям. Та й можливості RM ліцензії так само були зайві. Зокрема, для використання в оформленні в дизайні і наповненні сайтів в інтернет, коли немає як таких термінів використання і при цьому не важлива ексклюзивність зображення і історія його використання. Власне відповіддю був новий тип ліцензування – Royalty Free (RF). Найбільш відомі представники: iStockphoto, Shutterstock, Fotolia, Dreamstime.

Royalty Free (RF) – не накладає на автора обмежень на продаж фотографії тільки в одному фотобанку, також відсутні обмеження на кількість продажів, але

всі права на зображення (авторські та майнові залишаються у автора). При цьому автору виплачується фіксований гонорар (royalty), який не може бути змінений в подальшому. Покупець при цьому не несе ніяких додаткових витрат. Тобто один і той же матеріал може бути проданий в різних місцях необмежену кількість разів. Вартість ліцензії від одного до десятків доларів і часто залежить від розміру зображення. Однак цей тип ліцензії накладає певні обмеження на використання матеріалу.

Pixabay – це безкоштовний фотобанк, який об'єднує понад 2 682 447 користувачів з усього світу. З них 45 294 фотографів, які щодня поповнюють базу зображень [2].

Переваги Pixabay:

- більше 700 000 зображень;
- тегування зображень;
- висока якість наданого контенту;
- редакторська перевірка всіх зображень;
- дозвіл авторів на безкоштовне використання;
- більше 100 нових зображень щодня.

На Pixabay користувачі можуть не тільки завантажувати зображення і відео безкоштовно, але і оцінювати і контролювати їх якість. Для того щоб завантажувати файли на сайт або завантажувати зображення в оригінальній якості, необхідна реєстрація. При завантаженні файлів автори відмовляються від авторського і суміжних прав на зображення згідно з ліцензією Creative Commons CC0. Відповідно до цього документа будь-яка людина має право використовувати, модифікувати і поширювати все зображення з сайту Pixabay вільно як в особистих, так і в комерційних цілях. Для цього не потрібно запитувати у авторів додаткові дозволи.

Але слід мати на увазі, що представлені зображення і відео можуть бути захищені правами на використання товарних знаків, релізом моделі або релізом

власності. Щоб уникнути виникнення правових проблем і дотримання високого стандарту якості всі файли зображення перевіряються вручну співробітниками Pixabay.

Підбір ключових слів – це досить нудне, але абсолютно необхідне заняття. Від підібраних слів залежить рівень продаж фото Рекомендується використовувати не менше 20 ключових слів для кожного зображення, а краще ще більше.

Існують спеціалізовані сервіси для підбору ключових слів, наприклад у фотостоку Shutterstock. А деякі фотобанки, наприклад Dreamstime, надають платні послуги з ручного підбору ключових слів замість вас.

1.2. Тегування зображень за допомогою нейронних мереж

Задача тегування (класифікації) зображень є досить популярною. Розроблено досить багато математичних моделей для її рішення.

KNN: підхід, що ґрунтуються на K-Nearest-Neighbor. Створюють список тегів, розраховуючи релевантність тега для даного зображення. Процедура проводиться шляхом спочатку пошуку найближчих сусідніх зображень top-K, які були анотовані тегом, а також подальшого ранжирування списку тегів за частотою тегів.

TagProp: підхід на основі розповсюдження тегів використовують голосування сусідів із схемою навчання метричних показників відстані до тегів зображень [3].

CNN: Convolutional Neural Network сучасна архітектура мережі з декількома згортковими та повнозв’язними шарами. [4]

RPCA + CNN: Robust Principle Component Analysis та CNN спочатку видаляє зразки з великими помилками реконструкції за допомогою RPCA, а потом проводить навчання CNN на очищених зразках [5].

CAE + CNN: Convolutional Auto-Encoder + CNN пропонує знизити шумовий ефект при навчанні CNN шляхом попередньої підготовки даних та тонкої настройки покрокової стратегії [6].

Незважаючи на те, що вже розроблено достатньо моделей для вирішення даної задачі, більшість сайтів-фотостоків не надають можливості автоматичного тегування зображень.

1.3. Постановка задачі

Задача тегування представляє собою підбір ключових слів (тегів), що найкраще характеризують зміст зображення [7]. Існують різні підходи до її вирішення, які умовно поділяються на дві категорії: класифікація на множині тегів та пошук найбільш схожих зображень з подальшим ранжуванням та відбором їх тегів. Попереднім етапом обох підходів є пошук характерних рис зображення. Найбільш поширеним є спосіб, при якому вихідний графічний файл подається у вигляді вектора, компонентами якого є різні характеристики, що впливають на прийняття рішення про те, до якого класу можна віднести даний зразок та схожість зображень.

Створити систему автоматичного опису зображень для фотостоків, для чого необхідно:

1) провести аналіз предметної області, здійснити огляд існуючих підходів до задачі автоматичного тегування та категоризації зображень;

2) розробити програмне забезпечення збору даних та формування навчальної вибірки;

3) розробити та реалізувати алгоритми тегування та категоризації на основі згорткових нейронних мереж та визначення візуальної схожості зображень;

4) застосувати технологію Transfer Learning для підвищення швидкості навчання нейронних мереж;

5) запропонувати ансамблевий нейромережевий підхід для навчання за категоріями зображень;

6) провести аналіз якості роботи системи з використанням згорткових нейронних мереж VGG16, Inception v3, ResNet50;

7) розробити архітектуру програми, що надасть можливість зручного її використання сторонніми користувачами за допомогою технології дистанційного виклику процедур (RPC);

8) розробити web-додаток для зручної взаємодії користувача з системою.

РОЗДІЛ 2
РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ ТЕГУВАННЯ НЕЙРОННИХ МЕРЕЖ

2.1. Тегування зображень на основі згорткових нейронних мереж

У випадку згорткових нейронних мереж вектором, що характеризує зображення, є дескриптор, який генерується першою (згортковою) частиною мережі на основі вектора значень кольорів пікселів [8]. Класифікатором являється друга (повнозв’язна) частина мережі.

Згорткова нейронна мережа – це особливий вид штучних нейронних мереж. Згорткові нейронні мережі показують відмінні результати при обробці даних з просторовою структурою з кількох причин:

– стійкість до зрушень і поворотів об'єкта на зображенні, а так само стійкість до шумів;

– врахування просторової структури вхідних ознак;

– менша кількість параметрів, які необхідно оптимізувати, відносно класичних повнозв'язних мереж;

– більш швидке і якісне навчання відносно навчання повнозв'язних мереж.

Згорткова нейронна мережа зазвичай являє собою чергування згорткових (convolution layers), підвибіркових шарів (subsampling layers) і при наявності повнозв'язних шарів (fully-connected layer) на виході [9].

Шар згортки – це основний блок згорткової нейронної мережі. Він включає в себе для кожного каналу свій набір фільтрів, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента). Вагові коефіцієнти ядра згортки (невеликої матриці) невідомі і встановлюються в процесі навчання.

Скалярний результат кожної згортки потрапляє на функцію активації, яка представляє собою якусь нелінійну функцію. Шар активації зазвичай логічно пов'язують з шаром згортки (вважають, що функція активації вбудована в шар згортки). Функція нелінійності може бути будь-яка з вибору дослідника, традиційно для цього використовували функції типу гіперболічного тангенса або сигмоїдна. Однак в 2000х роках була запропонована і досліджена нова функція активації – ReLU (Rectified linear unit), яка дозволила суттєво прискорити процес навчання і одночасно спростити обчислення (за рахунок простоти самої функції). ReLU виконує операцію відсікання негативної частини скалярної величини. Станом на 2019 рік ця функція і її модифікації (Noisy ReLU, Leaky ReLU і інші) є найбільш часто використовуваними функціями активації в глибоких нейромережах, зокрема, в згорткових.

Пулінговий шар (інакше підвибірки, субдискретизації) являє собою нелінійне ущільнення карти ознак, при цьому група пікселів ущільнюється до одного пікселя, проходячи нелінійне перетворення [10]. Найбільш часто використовується при цьому функція максимуму. Перетворення зачіпають непересічні прямокутники або квадрати, кожен з яких скорочується в один піксель, при цьому вибирається піксель, що має максимальне значення. Операція пулінгу дозволяє істотно зменшити просторовий обсяг зображення. Підвибірка інтерпретується так: якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібно, і воно ущільнюється до менш докладного. До того ж фільтрація вже непотрібних деталей допомагає не перенавчатися мережі. Шар пулінгу, як правило, розташовується після шару згортки перед шаром наступної згортки. Можна використовувати і інші функції – наприклад, середнього значення або L2-нормування. Однак практика показала переваги саме пулінгу з функцією максимуму, який включається в типові системи.

Dropout шар (dropout регуляризація) – спосіб боротьби з перенавчанням в нейронних мережах. Dropout регуляризація полягає в зміні структури мережі: кожен нейрон викидається з певною ймовірністю p. За такою прорідженою мережею проводиться навчання, після чого всі викинуті нейрони повертаються в мережу. Таким чином, на кожному кроці навчання ми налаштовуємо одну з можливих 2^N архітектур мережі, де під архітектурою ми розуміємо структуру зв'язків між нейронами, а через N позначаємо сумарне число нейронів. При тестуванні нейромережі нейрони вже не викидаються, але вихід кожного нейрона множиться на (1-p) – завдяки цьому на виході нейрона ми будемо отримувати математичне очікування його відповіді по всім 2^N архітектурам. Таким чином, навчену за допомогою dropout-регуляризації нейромережу можна розглядати як результат усереднення 2^N мереж.

Після кількох проходжень згортки зображення і ущільнення за допомогою підвибірок система перебудовується від конкретної сітки пікселів з високою роздільною здатністю до більш абстрактних карток ознак, як правило на кожному наступному шарі збільшується число каналів і зменшується розмірність зображення в кожному каналі. Зрештою залишається великий набір каналів, що зберігають невелику кількість даних (навіть один параметр), які інтерпретуються як абстрактні поняття, виявлені з вхідного зображення.

Ці дані об'єднуються і передаються на звичайну повнозв'язну нейронну мережу, яка теж може складатися з декількох шарів. При цьому повнозв'язні шари вже втрачають просторову структуру пікселів і мають порівняно невелику розмірність (по відношенню до кількості пікселів вхідного зображення).

2.2. Адаптивний метод навчання нейронних мереж

Навчання нейронної мережі – це процес, в якому параметри нейронної мережі налаштовуються за допомогою моделювання середовища, в яке ця мережа вбудована.

Нейронні мережі часто навчають стохастично, тобто на різних ітераціях використовуються частини даних. Це мотивовано, як мінімум, двома причинами: по-перше, набори даних, що використовуються для навчання, часто дуже великі, щоб зберігати їх повністю в оперативній пам'яті і/або робити обчислення ефективно; по-друге, функція, яка оптимізується, зазвичай неопукла. Таким чином, використання різних частин даних на кожній ітерації може допомогти від застрявання моделі в локальному мінімумі. Крім того, навчання нейронних мереж зазвичай проводиться за допомогою градієнтних методів першого порядку, так як через велику кількість параметрів в нейронній мережі неможливо ефективно застосовувати методи більш високих порядків [11].

Стандартним методом навчання нейронних мереж є метод стохастичного градієнтного спуску (SGD). Однак він може розходитися або сходитися дуже повільно, якщо крок навчання налаштований недостатньо акуратно. Тому існує багато альтернативних методів з метою прискорити збіжність навчання і позбавити користувача від необхідності ретельної настройки гіперпараметрів [12].

Модифікації методу SGD:

- стохастичний градієнтний спуск з інерцією (SGDm);
- метод адаптивного градієнта (Adagrad);
- метод адаптивного змінного середнього градієнтів (RMSprop);
- метод адаптивного кроку навчання (Adadelta);
- метод адаптивної інерції (Adam).

Ці методи більш ефективно обчислюють градієнти і адаптивно змінюють крок навчання. Розглянемо докладніше метод SGD і одну з його найбільш популярних модифікацій – метод Adam.

Стохастичний градієнтний спуск (SGD) оновлює кожен параметр, віднімаючи градієнт функції, яка оптимізується, по відповідному параметру і масштабуючи його на крок навчання, який є гіперпараметром. Якщо крок навчання занадто великий, то метод буде розходитися; якщо занадто маленький – буде сходитися повільно. Через складність підбору гіперпараметрів було вирішено у роботі використовувати метод Adam

Метод Adam схожий на інші модифікації методу SGD (Adagrad, Adadelta, RMSprop). Відрізняється він від них двома ідеями: по-перше, оцінка перших кроків обчислюється як ковзне середнє; по-друге, через те, що оцінки першого і другого моментів встановлюються нулями, використовується невелика корекція, щоб результуючі оцінки не були зміщені до нуля. Метод також інваріантний до масштабування градієнтів [13].

2.3. Підхід Transfer Learning

Згорткові нейронні мережі зазвичай мають велику кількість нейронів та шарів. Навчання таких мереж потребує значних обчислювальних потужностей та затрат часу. За останні роки з'явилася велика кількість моделей створених і навчених професіоналами з використанням великої кількості даних і великих обчислювальних потужностей. Багато з цих моделей знаходяться у відкритому доступі і будь-хто може використовувати їх для вирішення своїх завдань абсолютно безкоштовно.

Одними з найбільш поширених є:

– VGG16 – мережа Visual Geometry Group з університету Оксфорда для розпізнавання об'єктів на зображеннях, складається з 16 шарів (Рис. 2.1);

– Inception v3 – нейронна мережа компанії Google для розпізнавання об'єктів на зображеннях (Рис. 2.2);

– ResNet50 – нейронна мережа компанії Microsoft, яка використовує залишкове навчання (Рис. 2.3).

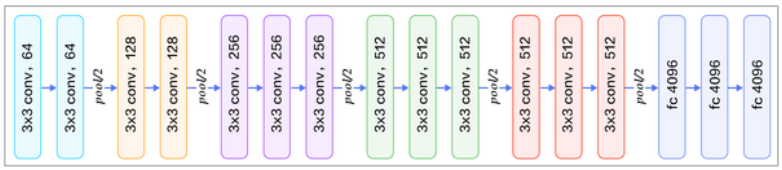

Рис. 2.1. Архітектура мережі VGG16

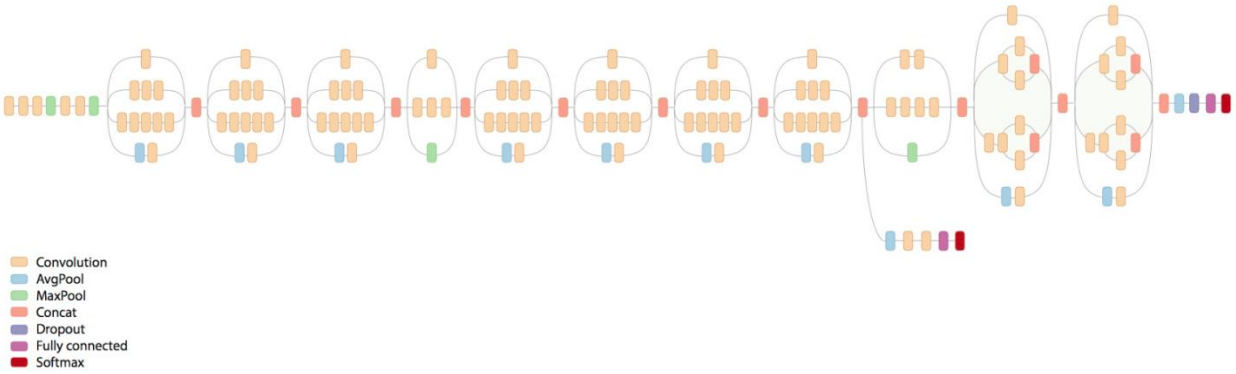

Рис. 2.2. Архітектура мережі InceptionV3

Ці мережі натреновані на зображеннях з бази даних ImageNet, яка містить 14 мільйонів зображень, що відносяться до 21 тисячі класів. В основному, навчання проводять на об'єктах з класів, з якими ми часто зустрічаємося в повсякденному житті.

Мережі складаються з двох частин.

Перша частина виділяє характерні ознаки в зображенні. Складається з каскадів згортки і підвибірки, які чергуються між собою.

Друга частина відповідає за класифікацію об'єкта на зображенні по виділеним на попередньому етапі ознаками.

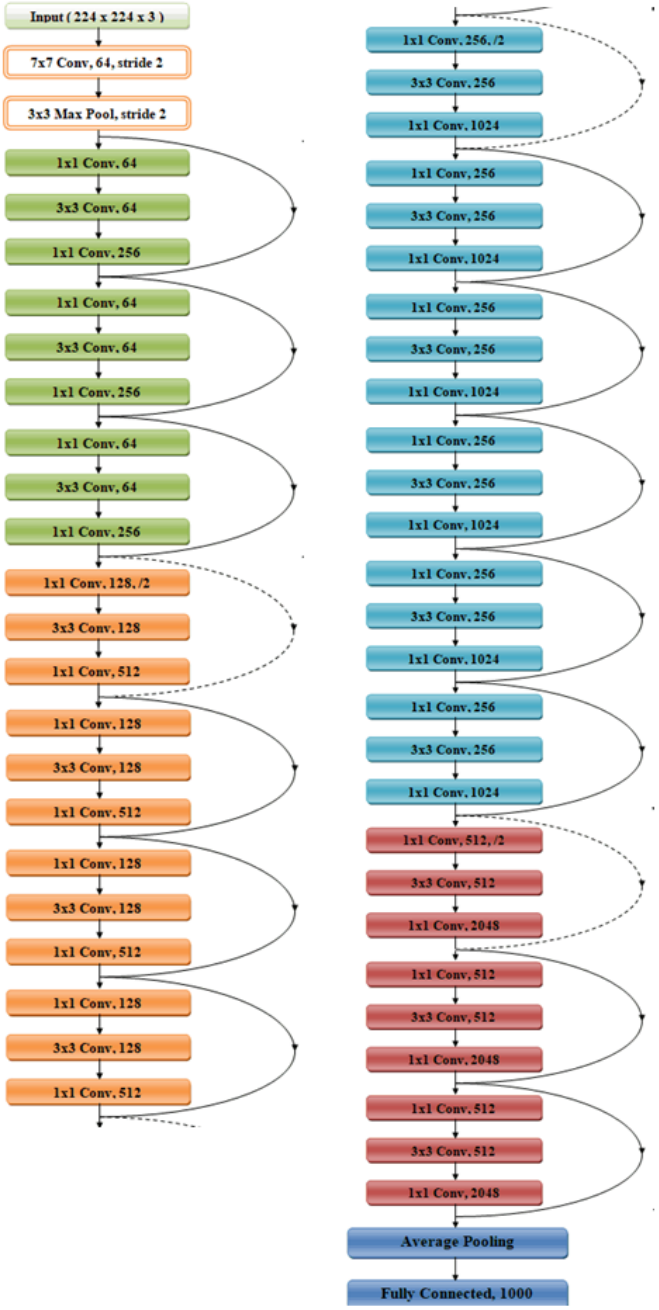

Рис. 2.3. Архітектура мережі ResNet50

На вхід мережі отримують зображення з трьома каналами кольору (червоний, зелений і синій). На виході мережі видають ймовірності, що на зображенню відповідаю той чи інший тег [14].

Для використання попередньо навчених мереж для вирішення нашої нової задачі необхідно змінити їх архітектуру за допомогою технології перенесення навчання, а саме:

– замінити повнозв'язну частину мережі своєю, яка розроблена безпосередньо для вирішення необхідної нам задачі;

– залишити незмінними вагові коефіцієнти згорткових шарів мережі, які вже є налаштованими для виділення різноманітних ознак зображень;

– навчити мережу на нових даних, що відповідають поставленій задачі.

Таким чином навчання потребує лише повнозв'язана частина, що значно зменшує необхідні обчислювальні потужності та час. А завдяки попередньому навчанню згорткової частини на великому обсязі даних досягається гарний результат.

2.4. Побудова ансамблю нейромереж для навчання за категоріями зображень

Використання нейронних мереж для тегування зображень є досить гарною практикою. Згорткові шари якісно виконують пошук характерних рис зображення, а повнозв’язні, на основі отриманих дескрипторів, класифікують зображення [15].

Було реалізовано нейронну мережу, яка має наступну архітектуру (Рис. 2.4):

– попередньо навчена згорткова мережа без повнозв’язних шарів;

– повнозв’язна модель, у якої кількість вихідних нейронів дорівнює кількості тегів (у випадку тестування – 100).

Але для вирішення такої задачі на більш-менш великому наборі даних необхідні потужні обчислювальні ресурси.

```
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                1605696
_________________________________________________________________
activation_1 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               12900
_________________________________________________________________
activation_3 (Activation)    (None, 100)               0
=================================================================
Total params: 16,341,604
Trainable params: 1,626,916
Non-trainable params: 14,714,688
```

Рис. 2.4. Архітектура мережі тегування зображень

Для навчання було зібрано 12000 зображень та 18000 відповідних їм міток. Отже, для реалізації такої модель необхідно було б навчати мережу, у якої 18000 вихідних нейронів. Для тестування даної моделі було вирішено скоротити набір тегів до 100 найбільш популярних. Реалізувавши дану мережу, були отримані невтішні результати. А саме, час навчання такої моделі складав 24 години лише для однієї епохи. Для отримання прийнятного результату необхідно щонайменше 25 епох. Тому було вирішено модернізувати даний підхід та запропоновано наступний алгоритм:

1) розділити навчальну вибірку зображень на категорії;
2) для кожної категорії створити свою нейронну мережу;
3) навчити мережу розпізнавати теги зображень своєї категорії;
4) створити та навчити окрему нейронну мережу, яка буде розпізнавати категорію зображення.

Оскільки на фотостоках, в тому числі на сайті https://pixabay.com, з якого збиралися дані для роботи, зображення вже розбиті на категорії, тому було вирішено використовувати саме їх.

Для обраних категорій відібрано популярні теги, які відносяться щонайменше до 15 зображень з навчальної вибірки.

В результаті модернізації алгоритму час навчання мережі значно скоротився. Також збільшилася гнучкість навчання, адже для кожної категорії можна окремо і по-різному навчати моделі.

2.5. Визначення тегів шляхом пошуку найбільш схожих зображень

Одним з популярних методів класифікації зображень є метод найближчих сусідів. Це досить простий метричний класифікатор, заснований на оцінюванні подібності об'єктів. Об'єкт, що класифікується, відноситься до того класу, якому належать найближчі до нього об'єкти навчальної вибірки.

Порівнювати зображення попіксельно – це старий метод, який дає дуже поганий результат. Тому необхідно описати кожне зображення у деякому кращому форматі для подальшого їх порівняння [16].

Вихідною інформацією зі згорткових шарів нейронної мережі є дескриптор зображення, тобто інформація, що описує його. Дескриптор являє собою багатомірний вектор, що є досить зручним форматом для порівняння.

В якості метрики схожості зображень було обрано манхетенську відстань (2.1).

$$D = \frac{1}{n}\sum_{i=1}^{n}\left|x_{1,i} - x_{2,i}\right| \qquad (2.1)$$

Отже, даний метод має наступний алгоритм:

1) за допомогою згорткової частини нейронної мережі отримуємо дескриптор зображення;

2) порівнюємо дескриптор зображення, що тегуємо, з дескрипторами з навчальної вибірки;

3) знаходимо найбільш схожі зображення та отримуємо їх теги;

4) сортуємо теги за популярністю та отримуємо набір тегів для вихідного зображення.

2.6. Критерії аналізу якості опису зображення

Провести аналіз якості опису зображення можна за допомогою методу F-score (2.2), який обчислюється середнім гармонічним рівнем точності та відкликання.

$$F = \frac{2 \times P \times R}{P + R} \tag{2.2}$$

де P – точність,

R – відкликання.

Точність (precision) і відкликання (recall) є метриками які використовуються при оцінці багатьох алгоритмів вилучення інформації. Іноді вони використовуються самі по собі, іноді в якості базису для похідних метрик, таких як F-score [17].

Точність системи в межах тегу – це частка зображень, що дійсно мають даний тег, відносно всіх зображень, які система помітила цим тегом.

Відкликання системи – це частка знайдених зображень, що мають даний тег, відносно всіх зображень з цим тегом в тестовій вибірці.

Точність (2.3) розраховується на основі кількості правильно позначених зображень кожного тегу та кількості тегів, що дійсно були назначені зображенням:

$$\frac{1}{c} \sum_{i=1}^{c} \frac{N_i^c}{N_i^p} \tag{2.3}$$

де c – кількість тегів,

N_i^c – кількість правильно позначених зображень для тегу i,

N_i^p – кількість прогнозів тегу i.

Відкликання (2.4) розраховується на основі кількості правильно позначених зображень кожного тегу та кількості тегів, що дійсно були попередньо назначені зображенням:

$$\frac{1}{c}\sum_{i=1}^{c}\frac{N_i^c}{N_i^g} \tag{2.4}$$

де c – кількість тегів,

N_i^c – кількість правильно позначених зображень для тегу i,

N_i^g – кількість тегів, що було попередньо назначено.

F-score є гарним методом формальної оцінки якості роботи системи тегування зображень. Він зводить до одного числа дві інші основоположні метрики: точність і відкликання.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Технології розробки програмного забезпечення

TensorFlow – це бібліотека програмного забезпечення з відкритим кодом для чисельних обчислень з високою ефективністю. Його гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (процесори, графічні процесори, TPU), а також від настільних комп'ютерів до кластерів серверів для мобільних і віддалених пристроїв. Розроблена дослідниками та інженерами команди Google Brain в організації AI Google, вона має розвинену підтримку для машинного та глибинного навчання, а гнучкі числові обчислення використовуються в багатьох інших наукових областях.

Особливості бібліотеки TensorFlow:

– основна бібліотека підходить для широкого сімейства технік машинного навчання, а не тільки для глибинного навчання;

– лінійна алгебра та інші нутрощі добре видно зовні;

– на додаток до основної функціональності машинного навчання, TensorFlow також включає власну систему логування, власний інтерактивний візуалізатор логів і навіть потужну архітектуру з поставки даних;

– модель виконання TensorFlow відрізняється від scikit-learn мови Python і від більшості інструментів в R.

Keras – відкрита нейромережева бібліотека, написана на мові Python. Вона являє собою надбудову над фреймворками Deeplearning4j, TensorFlow і Theano. Націлена на оперативну роботу з мережами глибинного навчання, при цьому спроектована так, щоб бути компактною, модульної та мати можливість розширюватися.

Зручність у використанні. Keras – API, призначений для людей, а не машин. Keras дотримується найкращих практик зменшення когнітивного навантаження: вона пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для звичайних випадків використання, і забезпечує чіткий та дієвий відгук щодо помилок користувача.

Модульність. Модель розуміється як послідовність або графік автономних, повністю налаштовуваних модулів, які можуть бути підключені разом із якомога меншими обмеженнями. Зокрема, нейронні шари, функції помилки обчислення, оптимізатори, схеми ініціалізації, функції активації, схеми регуляризації – це автономні модулі, які можна поєднати для створення нових моделей.

Легко розширювана. Нові модулі легко додавати як нові класи та функції, а існуючі модулі наводять достатньо прикладів. Можливість легко створювати нові модулі надає виразність, що робить Keras придатною для передових досліджень.

Робота з Python. Немає окремих файлів конфігурації моделей у декларативному форматі. Моделі описані в коді Python, який компактний, простий у налагодженні та дозволяє легко розширювати.

Python – мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Мова програмування Python – це потужний інструмент для створення програм найрізноманітнішого призначення, доступний навіть для новачків. З його допомогою можна вирішувати завдання різних типів.

Python має деякі примітні особливості, які обумовлюють її широке поширення.

Python – інтерпретована мова програмування. З одного боку, це дозволяє значно спростити налагодження програм, з іншого – зумовлює порівняно низьку швидкість виконання.

Динамічна типізація. У python не треба заздалегідь оголошувати тип змінної, що дуже зручно при розробці.

Гарна підтримка модульності. Ви можете легко написати свій модуль і використовувати його в інших програмах.

Вбудована підтримка Unicode в рядках. В Python необов'язково писати все англійською мовою, в програмах цілком може використовуватися ваш рідну мову.

Підтримка об'єктно-орієнтованого програмування. При цьому його реалізація в python є однією з найбільш зрозумілих.

Автоматичне прибирання сміття, відсутність витоків пам'яті.

Зрозумілий та лаконічний синтаксис, що сприяє ясному відображенню коду. Зручна система функцій дозволяє при грамотному підході створювати код, в якому буде легко розібратися іншій людині в разі необхідності. Також ви зможете навчитися читати програми і модулі, написані іншими людьми.

Величезна кількість модулів, як входять в стандартну поставку Python, так і сторонніх. У деяких випадках для написання програми досить лише знайти підходящі модулі і правильно їх скомбінувати. Таким чином, ви можете думати про складання програми на більш високому рівні, працюючи з уже готовими елементами, які виконують різні дії.

Remote Procedure Call (RPC) – віддалений виклик процедур за допомогою XML. Сама методика віддаленого виклику процедури відома давно і використовується в таких технологіях, як DCOM, SOAP, CORBA. RPC призначений для побудови розподілених клієнт-серверних додатків. Це дає можливість будувати додатки, які працюють в гетерогенних мережах, наприклад

на комп'ютерах різних систем, робити віддалену обробку даних і управляти віддаленими програмами.

Як працює RPC? Додаток, виконуючи обробку деяких даних на локальній машині, звертається до деякої процедурою. Якщо її реалізація присутня в програмі, то процедура (функція) приймає параметри, виконує дію і повертає деякі дані. Якщо це віддалений виклик, ми повинні знати, де буде виконуватися наша процедура. Запит на виконання процедури разом з параметрами записується у вигляді XML-документа і за допомогою HTTP передається по мережі на інший комп'ютер, де з XML-документа витягується ім'я процедури, параметри і інша потрібна інформація. Після завершення роботи процедури формується відповідь (наприклад, повертаються дані) – і вона передається комп'ютеру, який послав запит.

XML-RPC – стандарт/протокол виклику віддалених процедур, що використовує XML для кодування своїх повідомлень і HTTP в якості транспортного механізму. Відрізняється винятковою простотою в застосуванні. XML-RPC, як і будь-який інший RPC інтерфейс, визначає набір стандартних типів даних і команд, які програміст може використовувати для доступу до функціональності іншої програми, що знаходиться на іншому комп'ютері в мережі. Застосування XML для опису даних дозволило спростити програмні засоби створення розподілених додатків, знизилися вимоги до клієнта і сервера.

MySQL – це популярний сервер баз даних, який використовується в різних додатках. SQL означає мову структурованих запитів – Structured Query Language, який MySQL використовує для комунікації з іншими програмами. Понад те, MySQL має свої власні розширені функції SQL для того щоб забезпечити користувачам додатковий функціонал.

База даних являє собою структуровану сукупність даних. Ці дані можуть бути будь-якими – від простого списку майбутніх покупок до переліку експонатів картинної галереї або величезної кількості інформації в корпоративній мережі.

Для запису, вибірки і обробки даних, що зберігаються в комп'ютерній базі даних, необхідна система управління базою даних, якою і є ПО MySQL. Оскільки комп'ютери чудово справляються з обробкою великих обсягів даних, управління базами даних відіграє центральну роль в обчисленнях. Реалізовано таке управління може бути по-різному – як у вигляді окремих утиліт, так і у вигляді коду, що входить до складу інших додатків.

MySQL є дуже швидким, надійним і легким у використанні.MySQL володіє також рядом зручних можливостей, розроблених в тісному контакті з користувачами. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами. І ось уже протягом кількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Незважаючи на те що MySQL постійно вдосконалюється, він уже сьогодні забезпечує широкий спектр корисних функцій. Завдяки своїй доступності, швидкості і безпеці MySQL добре підходить для доступу до баз даних по Internet.

PHP – це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Це мова програмування, спеціально розроблена для написання web-додатків (сценаріїв), що виконуються на Web-сервері. Важливою перевагою мови PHP перед такими мовами, як мов Perl і C полягає в можливості створення HTML документів з впровадженими командами PHP. Значним відмінністю PHP від будь-якого коду, що виконується на стороні клієнта, наприклад, JavaScript, є те, що PHP-скрипти виконуються на стороні сервера.

PHP дозволяє створювати якісні Web-додатки за дуже короткі терміни, отримуючи продукти, які легко модифікуються і підтримувані в майбутньому. PHP простий для освоєння, і в той же час здатний задовольнити запити професійних програмістів.

Головним фактором мови PHP є практичність. PHP надає програмісту засоби для швидкого і ефективного вирішення поставлених завдань. Практичний характер PHP обумовлений п'ятьма важливими характеристиками:

- традиційністю;
- простотою;
- ефективністю;
- безпекою;
- гнучкістю.

Zend Framework 3 – це безкоштовний PHP фреймворк з відкритим вихідним кодом. Його розробка спрямовується і спонсорується компанією Zend Technologies, яка розробила і саму мову PHP.

3.2. Збір та підготовка даних для навчання нейронних мереж

Навчання згорткових нейронних мереж має відбуватися на великому об’ємі вхідних даних. Чим якісніша вибірка буде брати участь у навчанні, тим краще мережа буде реалізовувати поставлену задачу.

Вхідними даними для згорткових мереж є зображення. Їх підготовка є достатньо складною задачею, так як необхідно вирішити наступні проблеми.

Маркування зображень.

В інтернеті знаходиться неймовірно велика кількість зображень. Але вхідні дані для навчання повинні мати відповідні їм помітки (теги). Даних, що відповідають поставленій вимозі, у відкритому доступі знайти не вдалось. Тому було вирішено сформувати свій набір.

Авторські права.

Більшість великих сховищ захищають свої сховища авторськими правами.

Об’єм даних.

Якість роботи навченої нейронної мережі залежить не лише від методів навчання та архітектури, а й від даних на яких вона навчається. Для отримання коректних результатів, мережу необхідно навчати на великих об’ємах даних.

Гарним джерелом промаркованих зображень є сайти-фотостоки. Вони містять велику кількість зображень розбитих на категорії та з присвоєними характерними мітками. Але майже всі фотостоки захищають свої зображення авторськими правами. Переглянувши велику кількість фотостоків було знайдено сайт https://pixabay.com/. Він надає вільний доступ до своїх зображень та має відкрите API для роботи з ним. Нажаль тестування його API дало невтішний результат: можливість скачати фото лише за його ідентифікатором та одне фото за один запит, мала кількість запитів в секунду. Тому було вирішено написати програмне забезпечення для автоматичного збору зображень з подібних сайтів.

Основні вимоги до програмного забезпечення збору та підготовки даних для навчання нейронних мереж:

– пошук та завантаження зображень з вказаного сайту;
– збереження зображень до локальної файлової системи;
– ведення обліку зображень в базі даних;
– підготовка зображень для подальшого використання в навчанні нейронної мережі;
– можливість модифікації програми для роботи з іншими сайтами;

В результаті було реалізовано програмне забезпечення на мові програмування Python. Алгоритм роботи даного ПЗ наступний:

– програма переходить на вказану користувачем сторінку сайту;
– знаходить перелік зображень на сторінці;
– переходить на сторінку зображення;
– завантажує зображення та його текстові мітки (теги);
– зберігає їх у файлову систему та робить запис з тегами у базу даних;
– переходить до іншого зображення;

– коли на сторінці всі зображення оброблено, переходить на наступну сторінку.

Зображення зберігаються у локальній файловій системі. Пошук по файлам виконується дуже повільно та незручно. Для рішення цієї проблеми дані зберігаються в базі даних MySQL (Рис. 3.1).

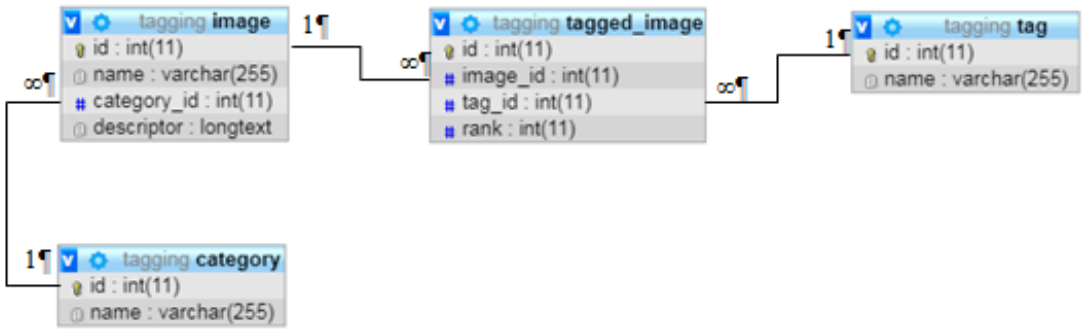

Рис. 3.1. Структура бази даних

При роботі з мережею інтернет трапляються збої, в результаті зображення зберігається не повністю або пошкодженим. Для рішення цієї проблеми реалізовано механізм валідації збережених зображень.

Архітектура програми спроектована таким чином, що модулі пошуку зображення на сайті, збереження у файловій системі та обробки відокремлені один від одного. Це дозволяє легко змінити сайт або місце збереження даних

3.3. Організація клієнт-серверної архітектури

Використання даного сервісу тегування зображень можливе через розроблений веб-сайт. Але це не основний спосіб його використання. Передбачається, що основна маса клієнтів буде його використовувати як додатковий функціонал на своїх ресурсах (веб-сайтах, мобільних додатках чи іншому ПЗ з виходом у мережу інтернет). Можливість використання сторонніми

ресурсами надається завдяки клієнт-серверній архітектурі програмного забезпечення.

Клієнт-сервер – мережева архітектура, в якій завдання розподілені між постачальниками послуг, званими серверами, і замовниками послуг, званими клієнтами (Рис. 3.2).

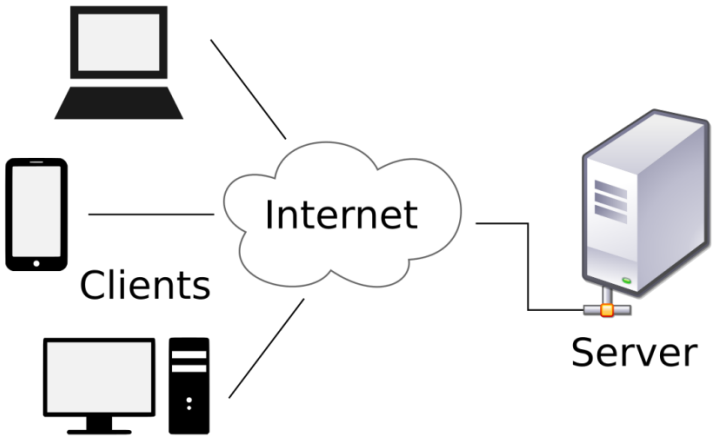

Рис. 3.2. Клієнт-серверна архітектура

В даному програмному продукті в ролі сервера виступає ПЗ розроблене на мові Python, яке приймає на вхід зображення, тегує його та видає на вихід текстові мітки зображення. В ролі клієнта – розроблений веб-сайт на мові PHP, який отримує від користувача зображення, відсилає серверові, отримує від нього текстові мітки та відображає їх користувачу.

```
$request = xmlrpc_encode_request( method: 'predict', ['image_base_64' => $data[1]]);
$response = $this->do_call( host: 'localhost',  port: '8000', $request);
```

Рис. 3.3. Запит на тегування на мові PHP

Реалізація клієнт-серверної архітектури виконана за допомогою технології RPC. Вона дозволяє легко та зручно зв’язувати клієнта та сервер. Наприклад, для виконання запиту на мові PHP необхідно виконати команду «xmlrpc_encode_request», вказавши в параметрах назву методу, та зображення в

форматі Base64, та відправити результат команди на сервер (Рис. 3.3). У відповідь прийде список тегів та їх вірогідність.

3.4. Серверна частина програмного забезпечення

Сервер являється основною частиною сервісу тегування зображень. Він виконую наступні функції:

– збір та обробка навчальних даних;

– навчання нейронних мереж;

– надання постійного доступу до засобів тегування та підбору категорій зображень.

Даний функціонал винесений в окремий модуль для надання можливості його використання стороннім сервісам.

Структура серверу:

– обробник вхідних запитів, який постійно прослуховує та приймає запити від клієнтів, обробляє їх та відсилає відповідь;

– функціонал тегування та категоризації зображень;

– засоби збору навчальних даних;

– засоби автоматизованого навчання згорткових нейронних мереж.

Для клієнта доступні три функції:

– тегування зображення за допомогою ансамблю нейронних мереж;

– підбір категорій зображенню;

– звужене тегування зображення за вказаною категорією.

Нижче описані алгоритми роботи сервера для кожної клієнтської функції.

Алгоритм категоризації зображення за допомогою згорткової нейронної мережі:

1) підготовка зображення для подальшої роботи (обрізка, масштабування, перенесення у масив numpy);

2) визначення вірогідності належності зображення кожній з категорій за допомогою згорткової нейронної мережі, яка відповідає за категоризування;

3) відбір категорій з найбільшою вірогідністю.

Алгоритм тегування зображення за допомогою ансамблю нейронних мереж:

1) підготовка зображення для подальшої роботи (обрізка, масштабування, перенесення у масив numpy);

2) визначення вірогідності належності зображення кожній з категорій за допомогою згорткової нейронної мережі, яка відповідає за категоризування;

3) відбір категорій з найбільшою вірогідністю;

4) для кожної відібраної категорії визначення вірогідності належності міток зображенню за допомогою згорткової нейронної мережі, яка відповідає за тегування саме даної категорії зображень;

5) відбір міток з найбільшою вірогідністю;

Алгоритм звуженого тегування зображення за вказаною категорією:

1) підготовка зображення для подальшої роботи (обрізка, масштабування, перенесення у масив numpy);

2) визначення вірогідності належності зображення кожній з міток за допомогою згорткової нейронної мережі, яка відповідає за тегування вказаної категорії зображень;

3) відбір міток з найбільшою вірогідністю.

3.5. Клієнтська частина програмного забезпечення

Для пересічного користувача було розроблено односторінковий веб-сайт. Він надає зручний та простий інтерфейс для використання сервісу опису зображень.

Сайт надає три основні функції:

– тегування зображення ансамблем нейронних мереж;

– підбір категорій до зображення;

– звужене тегування (в рамках обраної категорії).

Функціонал сайту досить простий. Він виступає в якості проксі-слою між клієнтом та сервером. Алгоритм роботи сайту наступний:

– клієнт завантажує зображення та обирає, що з ним зробити;

– за допомогою технології AJAX на backend частину відправляється запит з вхідними параметрами;

– backend його обробляє, підготовлює та відправляє запит до сервера;

– отримує відповідь від сервера та відправляє її на frontend частину;

– frontend відображає користувачу результат роботи.

Для реалізації було використано мову програмування PHP та Zend Framework 3.

Сайт розроблено за технологією Single page application (SPA). Тобто сайт використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і організовує взаємодію з користувачем через HTML, CSS, JavaScript, які завантажуються динамічно, за допомогою AJAX (Рис. 3.4). Завдяки цьому користувач працює з сайтом наче з настільним додатком, без перезавантаження сторінки.

```
$.ajax({
    type: "POST",
    url: "http://tagging/sendImage",
    data: "image=" + b64,
    success: function (data) {
        showGeneratedTags(JSON.parse(data));
    }
});
```

Рис. 3.4. Приклад AJAX запиту

3.6. Аналіз якості тегування та категоризації зображень

Аналіз якості тегування та категоризації зображень було проведено двома способами:

– метод F-score;

– вбудований метод бібліотеки Keras.

Нижче наведено результати аналізу якості роботи систем тегування зображень різними методами, а саме:

– згорткова + повнозв’язна нейронні мережі (табл. 3.1 та табл. 3.4);

– ансамбль згорткових нейронних мереж + повнозв’язна нейромережа (табл. 3.2 та табл. 3.5);

– згорткова нейронна мережа + метод найближчих сусідів (табл. 3.3 та табл. 3.6).

Таблиця 3.1

Тегування згортковою нейронною мережею

Згорткова нейронна мережа	F-score	Keras evaluate
VGG16	0.24	0.34
Inception v3	0.40	0.44
ResNet50	0.39	0.41

Таблиця 3.2

Тегування ансамблем згорткових нейронних мереж

Згорткова нейронна мережа	F-score	Keras evaluate
VGG16	0.68	0.71
Inception v3	0.74	0.76
ResNet50	0.71	0.74

Таблиця 3.3

Тегування методом найближчих сусідів

Згорткова нейронна мережа	F-score
VGG16	0.11
Inception v3	0.12
ResNet50	0.10

Таблиця 3.4

Категоризація згортковою нейронною мережею

Згорткова нейронна мережа	F-score	Keras evaluate
VGG16	0.35	0.42
Inception v3	0.42	0.48
ResNet50	0.43	0.47

Таблиця 3.5

Категоризація ансамблем згорткових нейронних мереж

Згорткова нейронна мережа	F-score	Keras evaluate
VGG16	0.71	0.74
Inception v3	0.76	0.81
ResNet50	0.74	0.80

Таблиця 3.6

Категоризація методом найближчих сусідів

Згорткова нейронна мережа	F-score
VGG16	0.14
Inception v3	0.17
ResNet50	0.15

3.7. Тестування програмного забезпечення та аналіз отриманих результатів

Результатом роботи є функціонуючий сервіс тегування зображень за допомогою згорткових нейронних мереж. Надано два варіанти його використання, а саме через сайт або RPC інтерфейс.

Сайт надає можливість звичайному пересічному користувачу тегувати зображення в простій та зручній формі. Він має лаконічний та інтуїтивно зрозумілий інтерфейс (Рис. 3.5 та 3.6). Реалізований з дотриманням підходу адаптивної верстки, що дозволяє використання як на комп'ютері з великим монітором, так і на смартфонах – з малим.

Рис. 3.5. Початкова сторінка сайту

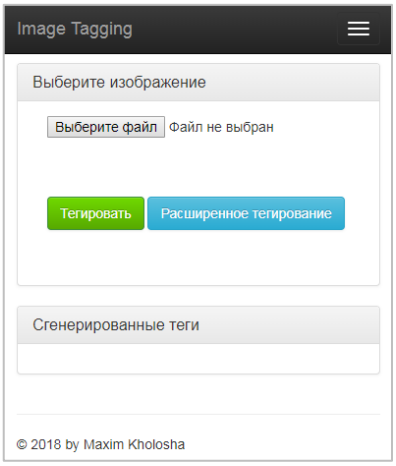

Рис. 3.6. Початкова сторінка сайту на смартфоні

Для роботи з сайтом необхідно завантажити зображення. Після завантаження воно відобразиться на сторінці (Рис. 3.7).

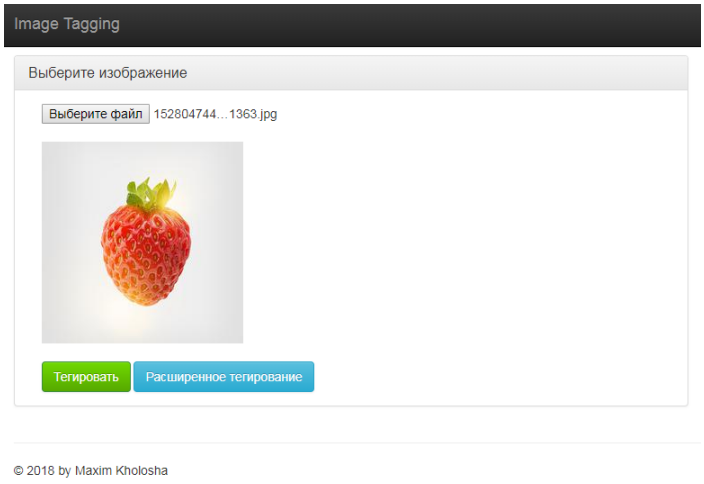

Рис. 3.7. Завантажене зображення

Для тегування зображення ансамблем нейронних мереж необхідно натиснути кнопку «Тегировать», після чого в правій частині сторінки з’явиться список тегів (Рис. 3.8).

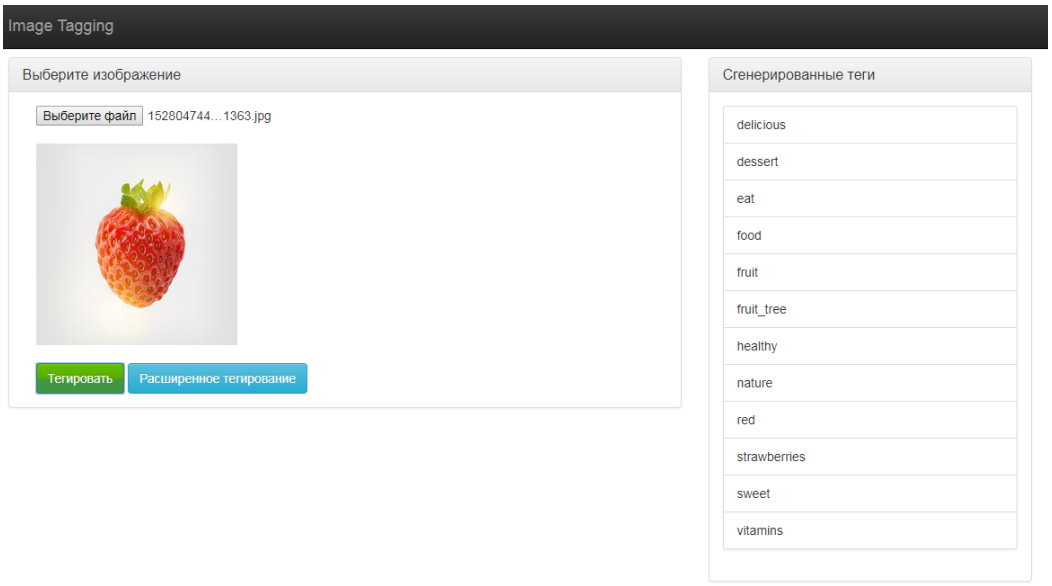

Рис. 3.8. Результат тегування

Для отримання категорій зображення необхідно натиснути на кнопку «Расширенное тегирование». З'явиться список категорій, до яких воно відноситься (Рис. 3.9).

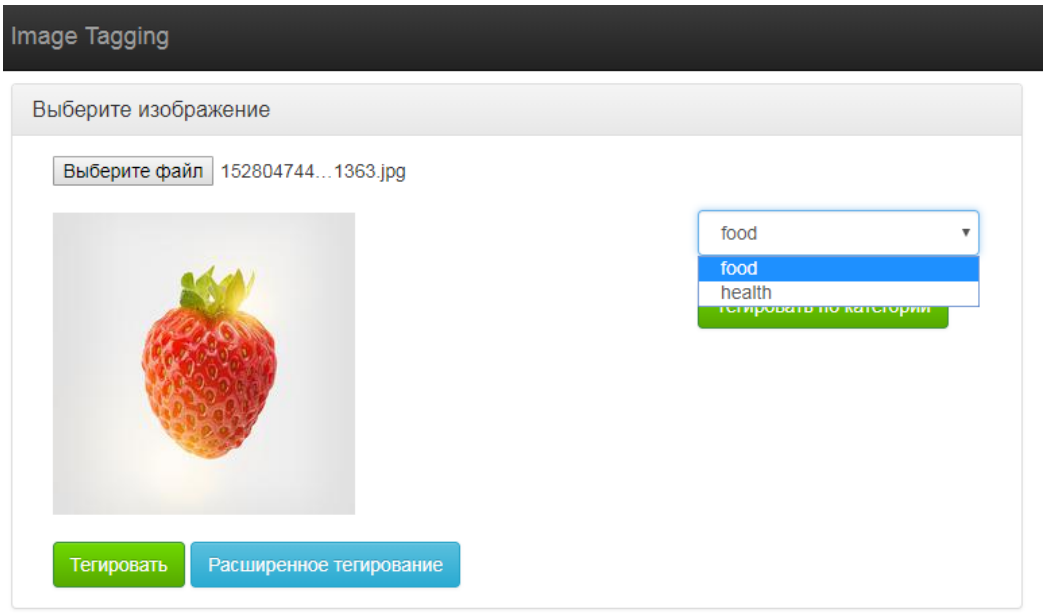

Рис. 3.9. Категорії зображення

Для звуженого тегування зображення необхідно обрати категорію та натиснути кнопку «Тегировать по категории». Після чого в правій частині екрану будуть відображені мітки (Рис. 3.10), які характеризують дане зображення відносно обраної категорії.

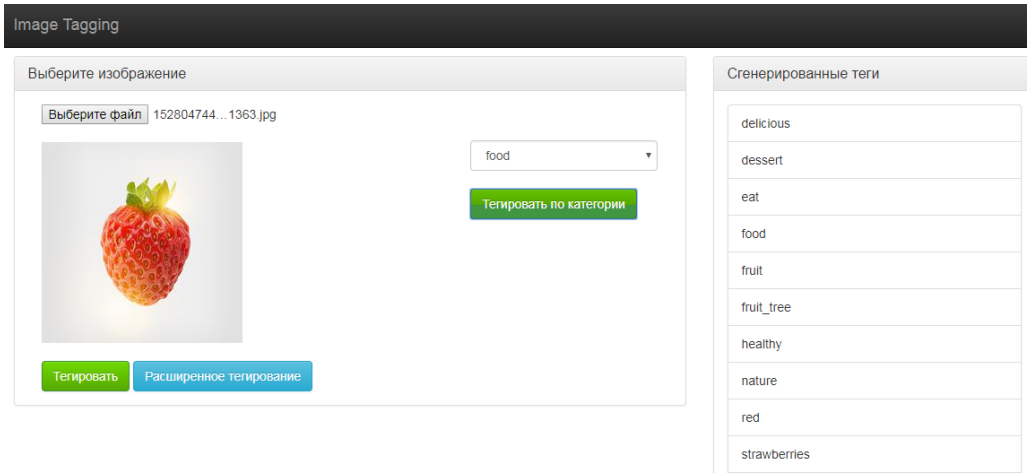

Рис. 3.10. Текстові мітки зображення відносно обраної категорії

Для виконання поставленої задачі було розроблено та реалізовано три методи тегування зображень:

1) нейронною мережею за всіма текстовими мітками;
2) ансамблем нейронних мереж;

3) шляхом пошуку найбільш схожих зображень.

Тегування зображень згортковою нейронною мережею за всіма текстовими мітками. Даний метод виявився занадто витратним в навчанні, тому було проведено його випробування на невеликій вибірці даних. Всі спроби навчити дану модель виявилися невдалими. Модель дуже повільно сходилася або взагалі розходилася. На рисунках 3.11 та 3.12 наведено результати навчання мережі.

Рис. 3.11. Точність мережі в процесі навчання

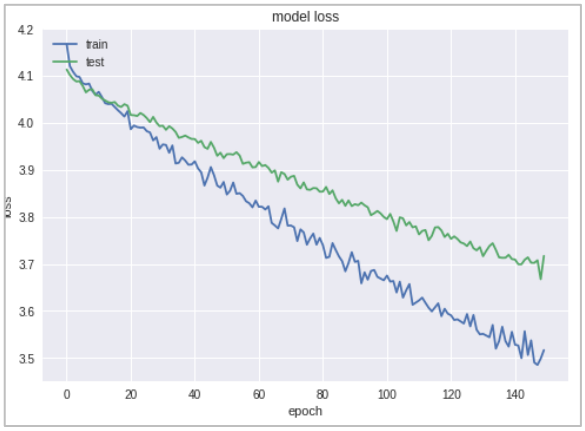

Рис. 3.12. Помилка мережі в процесі навчання

Тегування зображень ансамблем згорткових нейронних мереж. Даний метод є модифікацією методу тегування за всіма текстовими мітками. Його основними перевагами є значне збільшення швидкості та гнучкості навчання мережі. Навчання мереж проходить досить швидко та якісно.

Тестування методу з ансамблем нейронних мереж показало досить гарні результати як під час навчання, так і на практиці (Рис. 3.13 та 3.14).

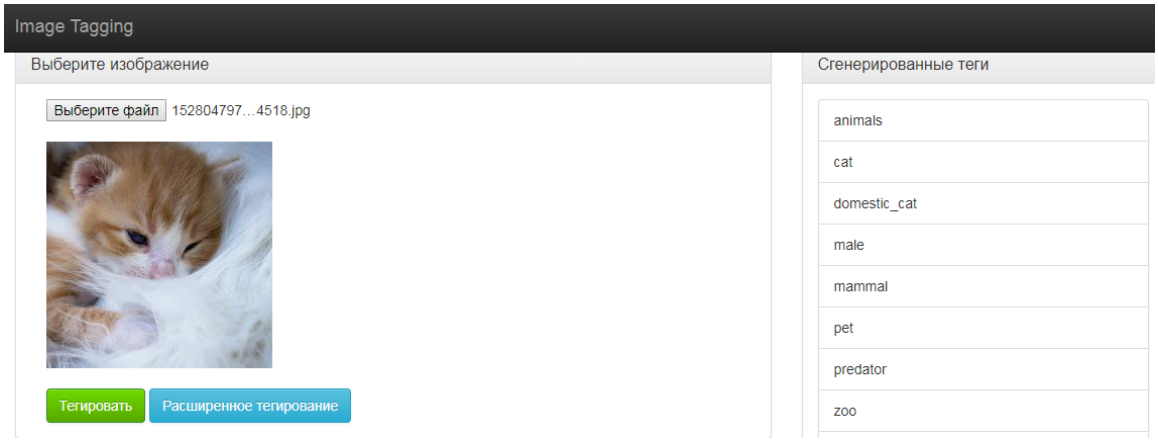

Рис. 3.13. Тегування зображення з кішкою

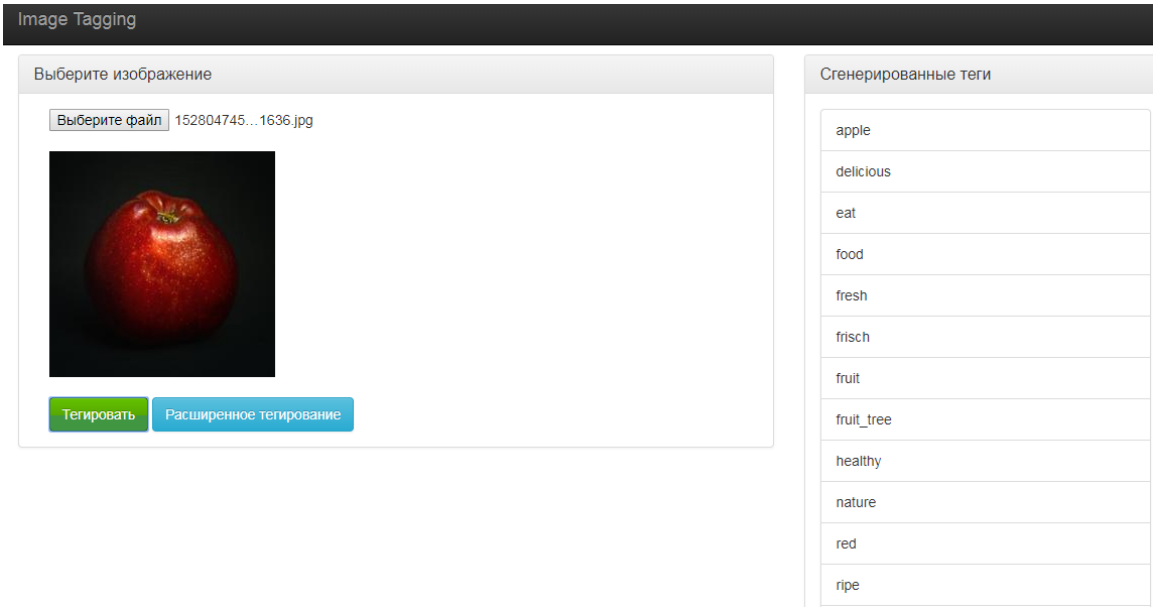

Рис. 3.14. Тегування зображення з яблуком

Тегування шляхом пошуку найбільш схожих зображень. В результаті тестування було виявлено, що методика пошуку схожих зображень за дескрипторами, отриманими на виході згорткових шарів нейронної мережі, не дає очікуваних результатів. Була виявлена закономірність надання переваги зображенням, які не мають характерних виділених об’єктів – монотонні зображення (Рис. 3.15 та 3.16). Виходячи з отриманих результатів можна зробити

висновок, що даний метод потребує додаткової модифікації для вирішення поставленої задачі.

Рис. 3.15. Приклад монотонного зображення

Рис. 3.16. Приклад монотонного зображення

Тестування розроблених методів опису зображень показало явну перевагу якості результатів методу з використанням ансамблю нейронних мереж над іншими. Результати довели доцільність використання даного методу для рішення поставленої задачі. А його гнучкість навчання та використання надають впевненості, що він знайде місце в рішенні реальних практичних задач.

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Розрахунок трудомісткості і вартості розробки програмного продукту

Вхідні дані:

- ✓ передбачуване число операторів – 3000
- ✓ коефіцієнт складності програми – 1,6
- ✓ коефіцієнт корекції програми в ході її розробки – 0,07
- ✓ часовая заработная плата программиста, грн/г – 80,0

У процесі створення ПЗ нормування праці ускладнено в силу творчого характеру праці програміста. Тому, трудомісткість розробки ПЗ розраховується на основі системи моделей з різною точністю оцінки.

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ чол.-г} \qquad (4.1)$$

де t_u – затрати праці на дослідження алгоритму розв'язання задачі, чол.-г;

t_a – затрати праці на розробку блок-схеми алгоритму, чол.-г;

t_n – затрати праці на програмування по готовій блок–схемі, чол.-г;

$t_{отл}$ – затрати праці на налагодження програми на ЕОМ, чол.-г;

$t_д$ – затрати праці на підготовку документації по завданню, чол.-г.

Ці затрати праці визначаються через умовне число операторів при розробці ПЗ, в число яких входять ті оператори , які необхідно написати в процесі роботи над програмою з урахуванням можливих уточнень у постановці завдання і вдосконалення алгоритму .

Умовне число операторів в програмі обчислюється за формулою:

$$Q = qC(1 + p), \qquad (4.2)$$

де q - передбачуване число операторів q = 2100 ;

c – коефіцієнт складності програми c = 1,6;

p − коефіцієнт кореляції програми в ході ее розробки p = 0,07.

Q = 3000 * 1,6 (1+ 0,07) = 3595

Затрати праці на вивчення опису завдання $t_и$ визначається з урахуванням уточнення опису та кваліфікації програміста.

$$t_u = \frac{QB}{(75..85)K} = \frac{3595*1,3}{77*1,2} = 50,58 \quad \text{чол.-год.,} \qquad (4.3)$$

де В - коефіцієнт збільшення затрат праці внаслідок недостатнього опису завдання:

$$B = 1,2. . . . 1,5;$$

К – коефіцієнт кваліфікації програміста, який визначається залежно від стажу роботи за даною спеціальністю. Він становить при стажі роботи, роки:

до 2 – 0,8;

від 2 до 3 – 1,0;

від 3 до 5 - 1,1 . . . 1,2;

від 5 до 7 - 1,3 . . . 1,4;

вище 7 – 1,5 . . . 1,6.

Затрати праці на розробку алгоритма для рішення задачі:

$$t_a = \frac{Q}{(20\ldots25)K} = \frac{3595}{22*1,2} = 136,17 \quad \text{чол.-год.} \qquad (4.4)$$

Витрати на складання програми по готовій блок -схемі:

$$t_n = \frac{Q}{(20..25)K} = \frac{3595}{22*1,2} = 136,17 \quad \text{чол.-год.} \qquad (4.5)$$

Затрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4..5)K} = \frac{3595}{4*1,2} = 748,96 \text{ чол.-год.} \qquad (4.6)$$

$$t_{отл}^{K} = 1,5t_{отл} = 1,5*748,96 = 1123,44 \text{ чол.-год.} \qquad (4.7)$$

Витрати на підготовку документації:

$$t_д = t_{др} + t_{до} \quad \text{чол.-год.,} \qquad (4.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів:

$$t_{др} = \frac{Q}{(15 \dots 20)K} = \frac{3595}{17 * 1,2} = 206,61 \text{ чол.-год.} \quad (4.9)$$

$t_{до}$ – трудомісткість редагування, друку та оформлення документації:

$$t_{до} = 0,75 t_{др} = 0,75 * 206,61 = 154,96 \text{ чол.-год.} \quad (4.10)$$

$t_{д} = t_{др} + t_{до} = 206,61 + 154,96 = 361,57$ чол.-год.

У підсумку отримуємо, що трудомісткість розробки ПЗ становить:

t = 50,58+136,17+136,17+1123,44+361,57 = 1807,93 чол.-год.

4.2 Затрати на створення програмного забезпечення

Витрати на створення ПО (Кпо) включають витрати на заробітну плату виконавців програми (Зз/п), визначену множенням сумарної трудомісткості розробки ПО (t) на середню зарплату з нарахуваннями програміста і вартості машинного часу, необхідного для відладки програми на ЕОМ (Змв), визначеною виходячи з вартості 1-го машинного години конкретного типу ЕОМ, і витрат машинного часу на налагодження.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{ , грн.} \quad (4.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * C_{ПР} = 1807,93 * 80,0 = 144634,4 \text{ грн.,} \quad (4.12)$$

де t - загальна трудомісткість, чол.-г.;

$C_{пр}$ - середня годинна заробітна плата програміста, грн./год;

$C_{пр}$ =50,0 грн./год.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * C_{мч} = 1123,44 * 1,2 = 1348,13 \text{ грн.,} \quad (4.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.;

$C_{мч}$ – вартість машинного часу ЕОМ, грн./год.

Кпо = 144634,4 + 1348,13 = 145982,53 грн. (4.14)

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУТП . Очікуваний період розробки ПЗ:

$$T = \frac{t}{B_k \bullet F_p} \text{ міс.,} \qquad (4.15)$$

де B_k – кількість виконавців;

F_p – місячний фонд робочого часу (при 40-ка годинному робочому тиждні F_p=176 годин).

$$T = \frac{1807,93}{1 * 176} = 11 \text{ місяців.}$$

Таким чином, період розробки програми складе приблизно 10 місяців.

методу.

4.3 Маркетингові дослідження ринку

Задача тегування представляє собою підбір ключових слів (тегів), що найкраще характеризують зміст зображення. Існують різні підходи до її вирішення, які умовно поділяються на дві категорії: класифікація на множині тегів та пошук найбільш схожих зображень з подальшим ранжуванням та відбором їх тегів. Попереднім етапом обох підходів є пошук характерних рис зображення. Найбільш поширеним є спосіб, при якому вихідний графічний файл подається у вигляді вектора, компонентами якого є різні характеристики, що впливають на прийняття рішення про те, до якого класу можна віднести даний зразок та схожість зображень.

Останні роки ми є свідками стрімкого росту кількості мобільних пристроїв (наприклад, смартфонів, цифрових камер тощо) та послуг хмарного зберігання інформації, що призвело до безпрецедентного зростання кількості персональних медіа-ресурсів, таких як зображення. Наприклад, люди знімають фотографії на

смартфони кожен день і скрізь. Повідомляється, що Flickr має 1,7 мільйона фотографій, завантажених щодня, а Instagram претендує на 140 мільйонів фотографій щодня в 2021 році. Така велика кількість зображень вимагає ефективного доступу до них. Один з найбільш ефективних способів пошуку зображень – через текстові мітки.

Особливо актуальна дана тема власникам великих об’ємів зображень. Одними з них є сайти-фотостоки, на які люди завантажують безліч зображень. Для кожного з них необхідно власноруч проставити текстові мітки, які його характеризують. Дана операція знижує бажання долі клієнтів завантажувати ще більше фото, адже для кожного потрібно буде ставити теги. Дану операцію можна і потрібно автоматизувати. Одним з рішень є тегування зображень з допомогою згорткових нейронних мереж.

ВИСНОВКИ

За результатами проведеної роботи можна зробити такі висновки:

1) створено систему автоматичного опису зображень для фотостоків;

2) проведено аналіз предметної області, здійснено огляд існуючих підходів до задачі автоматичного тегування та категоризації зображень;

3) розроблено програмне забезпечення збору даних та формування навчаючої вибірки;

4) розроблено та реалізовано алгоритми тегування та категоризації на основі згорткових нейронних мереж та визначення візуальної схожості зображень;

5) застосовано технологію Transfer Learning для підвищення швидкості навчання нейронних мереж;

6) запропоновано ансамблевий нейромережевий підхід для навчання за категоріями зображень;

7) проведено аналіз якості роботи системи з використанням згорткових нейронних мереж VGG16, Inception v3, ResNet50;

8) розроблено архітектуру програми, що надала можливість зручного її використання сторонніми користувачами за допомогою технології дистанційного виклику процедур (RPC);

9) розроблено web-додаток для зручної взаємодії користувача з системою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation, Kyoto, Japan, 2009. – 6с.
2. W. Zhou, H. Li, Q. Tian, Recent Advance in Content-based Image Retrieval: A Literature Survey, Fellow, 2017. – 31с.
3. A. Makadia, V. Pavlovic, S. Kumar, New Baseline for Image Annotation, Piscataway, 2015. – 10с.
4. БОДЯНСЬКИЙ Є.В, ДЕЙНЕКО А.О., ДЕЙНЕКО Ж.В, ШАЛАМОВ М.О., Адаптивне навчання нейронної мережі опорних векторів найменших квадратів, 2015, ІКСЗТ. –52с.
5. P. Kingma, J. Lei Ba, Adam: a method for stochastic optimization, Toronto, 2015. – 3с.
6. J. Fu, Y. Rui, Advances in deep learning approaches for image tagging, Cambridge, 2017. – 1с.
7. Y. Gong, Y. Jia, T. K. Leung, Deep Convolutional Ranking for Multilabel Image Annotation, Fellow , 2014. – 31с.
8. Frahim J. Securing the Internet of Things: A Proposed Framework / J. Frahim // Cisco White Paper.- 2015.
9. Aujla GS Data Offloading in 5G-Enabled Software-Defined Vehicular Networks: A Stackelberg Game-Based Approach / GS Aujla // IEEE Commun. Mag.- vol. 55, no. 7.- July 2017.
10. Peng M. Energy-Efficient Resource Assignment and Power Allocation in Heterogeneous Cloud Radio Access Networks / M. Peng // IEEE Transactions on Vehicular Technology.- vol. 64, no. 11.- Nov. 2015.- P. 5275-5287.
11. Gonzales D. Cloud-trust - A Security Assessment Model for Infrastructure as a Service (Iaas) Clouds / D. Gonzales // IEEE Transactions on Cloud Compu-ting.- vol. 5, no. 3.- July-Sept. 1, 2017.- P. 523-536.

12. John W. Research Directions in Network Service Chaining / W. John // 2013 IEEE SDN for Future Networks and Services.- Nov. 2013.- p. 1-7.
13. Automatic speech recognition and speech variability: A review / M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, C. Wellekens // Speech Communication, Elsevier. – 2007. – №49. – P. 763-786.
14. V.V.R. Vegesna. Prosody modification for speech recognition in emotionally mismatched conditions / V.V.R. Vegesna, K. Gurugubelli, A.K. Vuppala // International Journal of Speech Technology, 3. – 2018. – P. 521-532.
15. Harpreet Kaur. Prosody Modification of its Output Speech Signal / Harpreet Kaur, Parminder Singh // International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE). – 2014. – №5. – P. 1056–1059.
16. Krothapalli Sreenivasa Rao. Real Time Prosody Modification / Krothapalli Sreenivasa Rao // Journal of Signal and Information Processing. – 2010. – №1. – P. 50-62.
17. Fast Prosody Modification using Instants of Significant Excitation / S. R. M. Prasanna, D. Govind, K. S. Rao, B. Yegnanarayana // Speech Prosody. – 2010.
18. Synthesis of Emotional Speech by Prosody Modification of Vowel Segments of Neutral Speech / Md Shah Fahad, Shreya Singh, Shruti Gupta, Akshay Deepak and Abhinav // Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE). – 2019. – P. 49-54
19. D. Joshi. Speech Emotion Recognition: A Review / D. Joshi, M. B. Zalte // IOSR Journal of Electronics and Communication Engineering (IOSR – JECE). – 2013. – №4. – P. 34–37.

Додаток А

Лістинг програми

```
/*css file for graphs.html only*/
/*default classes joined from style.css*/

header{
  position: fixed;
  top: 0px;
  left: 0px;
  padding: 15px 22px;
  margin: 10px;
  background-color: #151618;
  border-radius: 10px;
  color: #f1f1f1;
  font-family: Montserrat_medium;
  font-size: 16px;
  float: left;
}

header:hover{
  text-decoration: underline;
}

.sidebar{
  height: calc(100vh - 200px) !important;
  padding: 100px 0px !important;
}

.sidebar_button{
  margin: 21px 21px !important;
}

.calc_content{
  width: calc(100% - 60px);
  height: calc(100vh - 60px);
  padding: 30px;
  margin: none;
  background-color: transparent;
  float: left;
}

.canvas_wrap{
  width: calc(50% - 25px - 50px);
  margin-right: 25px;
  padding: 25px;
  background-color: #00000044;
  overflow: hidden;
  border-radius: 25px;
  float: left;
}
```

```
canvas{
  width: 100%;
  height: 100%;
  float: left;
}

.canvas_navigation{
  float: left;
  width: 120px;
  height: 120px;
  overflow: hidden;
  border-radius: 100%;
  transform: rotate(45deg);
  position: relative;
}

.canvas_navigation div.button{
  float: left;
  width: 20px;
  height: 20px;
  padding: 20px;
  border: none;
  background-color: #151618;
  font-family: Montserrat_medium;
  text-align: center;
  line-height: 20px;
  color: #f1f1f1;
  font-size: 20px;
  cursor: pointer;
}

.canvas_navigation div.button:hover{
  background-color: #00000088;
}

.canvas_navigation div.button[onclick="changeCenter('top')"] span{
  transform: translateX(-4px);
  transform: translateY(-2px);
}

.canvas_navigation div.button[onclick="changeCenter('right')"] span{
  transform: translateX(-4px);
  transform: translateY(-2px);
}

.canvas_navigation div.button[onclick="changeCenter('center')"]{
  position: absolute;
  width: 20px;
  height: 20px;
```

```
  top: 30px;
  left: 30px;
  right: 30px;
  bottom: 30px;
  padding: 17px;
  background-color: #151618;
  border: 3px solid #1d1e21;
  border-radius: 100%;
}

.canvas_navigation div.button[onclick="changeCenter('center')"]:hover{
  background-color: #00000088;
}

.canvas_navigation div.button span{
  width: 20px;
  height: 20px;
  line-height: 20px;
  color: #f1f1f1;
  font-weight: 400;
  font-size: 26px;
  font-family: Montserrat_medium;
  cursor: pointer;
  float: left;
}

.canvas_navigation div.button span::selection, .canvas_navigation div.button::selection{
  background-color: transparent !important;
}

.scale_wrap{
  width: 50px;
  margin: 0px 0px 0px 30px;
  float: left;
}

.scale_wrap div.scale{
  width: 50px;
  height: 14px;
  padding: 10px 0px;
  margin: 3px 0px;
  background-color: #00000044;
  border-radius: 6px;
  font-size: 16px;
  color: #f1f1f1;
  text-align: center;
  float: left;
}
```

```
.scale_wrap div.scale:hover{
  background-color: #00000088;
  cursor: pointer;
}

hr{
  width: 2px;
  height: 120px;
  background-color: #f1f1f133;
  margin: 0px 20px;
  border: none;
  float: left;
}

.labelbar{
  float: none;
  margin: 8px 0px;
  width: auto;
  height: auto;
}

.labelbar p{
  display: inline-block;
  color: #f1f1f1;
  font-family: Montserrat_medium;
  font-size: 13px;
  margin: 0px 5px;
}

.labelbar .switch {
  position: relative;
  width: 34px;
  height: 20px;
  float: left;
}

.labelbar .switch input {
  opacity: 0;
  width: 0;
  height: 0;
}

.labelbar .slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
```

```
  background-color: #00000088;
  border-radius: 50px;
  transition: 0.4s;
}

.labelbar .slider:before {
  position: absolute;
  content: "";
  height: 14px;
  width: 14px;
  top: 0px;
  left: 0px;
  bottom: 0px;
  margin: 3px;
  background-color: white;
  transform: translateX(0px) scale(0.6);
  border-radius: 50px;
  transition: 0.4s;
}

.labelbar input:checked + .slider:before {
  transform: translateX(14px) scale(1);
  background-color: #ffca00;
}

.func_content{
  width: 50%;
  height: 100%;
  float: left;
}

.func_set{
  width: calc(100% - 30px);
  height: 128px;
  margin-bottom: 10px;
  padding: 15px;
  background-color: #00000044;
  border-radius: 25px;
  float: left;
}

.func_set input{
  height: 16px;
  padding: 13px;
  margin: 9px;
  background-color: #00000044;
  border: 2px solid transparent;
  font-family: monospace;
  border-radius: 8px;
```

```
  font-size: 16px;
  color: #f1f1f1;
  float: left;
}

.func_set input:hover, .func_set input:focus{
  border: 2px solid #ffca00;
}

.func_set input.func_field{
  width: calc(100% - 20px - 26px - 2px);
}

.func_set input.graph_int_min, .func_set input.graph_int_max{
  width: 100px;
}

.func_set p{
  font-family: Montserrat_medium;
  font-size: 16px;
  color: #f1f1f1;
  padding: 22px 15px;
  float: left;
}

.func_set .button{
  position: relative;
  width: 110px;
  height: 14px;
  margin: 10px;
  padding: 12px 0px;
  background-color: #ffca00;
  text-align: center;
  font-size: 14px;
  border: 2px solid #ffca00;
  border-radius: 12px;
  overflow: hidden;
  cursor: pointer;
  float: right;
}

.func_set .button:before{
  position: absolute;
  margin: auto;
  z-index: 2;
  left: 0px;
  top: 0px;
  right: 0px;
  bottom: 0px;
```

```
  content: attr(content);
  width: 110px;
  height: 110px;
  transform: scale(0);
  border-radius: 100%;
  background-color: #1d1e21;
  transition: 0.3s;
}

.func_set .button:hover:before{
  transform: scale(1.1);
}

.func_set .button span{
  display: inline-block;
  position: absolute;
  z-index: 4;
  left: 0px;
  top: 0px;
  right: 0px;
  bottom: 0px;
  padding: 12px 0px;
  color: #27292d;
  font-family: Montserrat_medium;
  font-weight: 300;
  transition: 0.3s;
}

.func_set .button:hover span{
  color: #ffca00;
}

.analysis_field{
  width: calc(100% - 30px);
  margin-bottom: 10px;
  padding: 15px;
  background-color: #00000044;
  border-radius: 25px;
  position: relative;
  float: left;
}

.analysis_field.keyboard button{
  width: calc((100% / 7) - 10px);
  height: 40px;
  margin: 5px;
  color: #f1f1f1;
  font-family: monospace;
  font-size: 14px;
```

```
  background-color: #00000044;
  border: none;
  border-radius: 5px;
  overflow: hidden;
  cursor: pointer;
  transition: 0.4s;
  float: left;
}

.analysis_field.keyboard button:hover{
  background-color: #00000088;
}

.analysis_field.keyboard button.clear{
  background-color: #ffca00;
  color: #1e1e1e;
  font-family: Montserrat_medium;
}

.analysis_field.keyboard button.clear:hover{
  background-color: #aa8700;
}

.analysis_field.keyboard button sub, .analysis_field button sup{
  font-family: monospace;
  margin: 1px 2px;
  opacity: 0.55;
}

.analysis_field#analysis_field{
  width: calc(100% - 50px);
  padding: 25px;
}

.analysis_field#analysis_field span{
  width: 100%;
  color: #f1f1f1;
  font-family: monospace;
  font-size: 16px;
  text-align: left;
  float: left;
}

.analysis_field#analysis_field span#fz_title{
  width: auto;
  float: none;
}

.analysis_field#analysis_field span#function_zeroes{
```

```
  margin-top: 10px;
}

.analysis_field#analysis_field nav{
  width: 6px;
  height: 6px;
  border-radius: 10px;
  margin: 6px 7px 6px 0px;
  float: left;
}

/*---medias---*/

@media (max-width: 1425px){ /* for middle width monitors */
  .func_values input{
    width: 6vw !important;
  }

  .func_values p{
    display: none;
  }
}

@media (max-width: 1300px){ /* <1300px */
  .analysis_field button{
    width: calc((100% / 5) - 10px);
  }
}

@media (max-width: 1000px){ /* Mobile version (<1000px) */
  .func_content{
    width: calc(100% - 20px);
    margin: 10px;
  }

  .analysis_field, .func_set{
    width: calc(100% - 60px - 50px);
    margin: 10px 30px;
  }

  .analysis_field#analysis_field{
    width: calc(100% - 80px - 50px);
    margin: 10px 30px;
    margin-bottom: 30px;
  }

  .canvas_wrap{
    width: calc(100% - 80px - 50px);
    margin: 10px 30px;
```

```
  }

  .analysis_field button{
    width: calc((100% / 7) - 10px);
  }

  header {
    font-size: 30px;
    padding: 20px;
  }
}

@media (max-width: 800px){ /* for small mobiles */
  .func_content{
    width: 100%;
    margin: 0px;
  }

  .analysis_field button{
    width: calc((100% / 3) - 10px);
  }

  .canvas_wrap{
    width: calc(100% - 60px - 50px);
    margin: 30px;
  }

  .calc_content{
    margin: 0px;
    padding: 0px;
    width: 100%;
    float: left;
  }

  .analysis_field, .func_set{
    width: calc(100% - 90px);
    margin: 10px 30px;
    float: left;
  }

  .analysis_field#analysis_field{
    width: calc(100% - 60px - 50px);
  }
}

/*presets for main page (index.html)*/

header{
  width: 100%;
```

```
  height: 80px;
  background-color: #27292d33;
  position: absolute;
  top: 0px;
  left: 0px;
  right: 0px;
  z-index: 1001;
  transition: 0.3s;
  float: left;
}

header img{
  height: 50px;
  padding: 15px;
  margin: 0px 40px;
  float: left;
}

header div.button{
  width: 90px;
  height: 14px;
  padding: 11px 14px;
  margin: 20px 30px;
  text-align: center;
  border: 2px solid #ffca00;
  border-radius: 50px;
  overflow: hidden;
  position: relative;
  float: right;
}

header div.button:before{
  width: 100%;
  height: 100%;
  content: "";
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 1;
  background-color: #ffca00;
}

header div.button:hover:before{
  width: 0px;
}

header div.button span{
  width: 100%;
  position: absolute;
```

```
  top: 0px;
  left: 0px;
  z-index: 2;
  color: #1e1f23;
  padding: 11px 0px;
  font-size: 14px;
  font-family: Montserrat_medium;
  font-weight: 600;
  line-height: 16px;
  text-align: center;
}

header div.button:hover span{
  color: #ffca00;
}

header .dropdown{
  width: 130px;
  height: 80px;
  margin: 0px 40px;
  overflow: hidden;
  position: relative;
  transition: 0.3s;
  float: right;
}

header .dropdown .dropdown_btn{
  width: 100%;
  height: 14px;
  padding: 33px 0px;
  text-align: center;
  color: #f1f1f188;
  font-size: 14px;
  font-family: Montserrat_medium;
  font-weight: 600;
  line-height: 16px;
  float: left;
}

header .dropdown:hover .dropdown_btn{
  color: #f1f1f1;
}

header .dropdown .dropdown_content{
  width: 100%;
  height: 132px;
  position: absolute;
  z-index: 200;
  top: 80px;
```

```
  left: 0px;
  right: 0px;
  overflow: hidden;
  border-radius: 0px 0px 10px 10px;
  float: left;
}

header .dropdown:hover{
  height: 212px;
}

header .dropdown .dropdown_content a{
  width: 100%;
  height: 14px;
  padding: 14px 0px;
  margin: 1px 0px;
  background-color: #f1f1f122;
  text-align: center;
  color: #f1f1f188;
  font-size: 14px;
  font-family: Montserrat_medium;
  font-weight: 600;
  line-height: 16px;
  float: left;
}

header .dropdown .dropdown_content a:hover{
  color: #f1f1f1;
}

.block{
  position: relative;
  width: 100%;
  height: 100vh;
  background-size: cover;
  overflow: hidden;
}

.block .img{
  position: absolute;
  top: 0px;
  left: 0px;
  right: 0px;
  bottom: 0px;
  z-index: 2;
  width: 100vw;
  height: 100%;
  background-image: url(../img/pattern.png);
  background-repeat: repeat;
```

```
  background-size: 300px;
 }

 .block #light_round{
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 1;
  width: 400px;
  height: 400px;
  border-radius: 100%;
  background-image: radial-gradient(#ffca00 0%, #ffca00aa 25%, #ffca0055 50%, transparent
70%, transparent 100%);
  animation: 6s bit linear infinite;
  transition: none;
 }

 @keyframes bit{
  0%{opacity: 0;}
  45%{opacity: 1;}
  55%{opacity: 1;}
  90%{opacity: 0;}
  100%{opacity: 0;}
 }

 .block .about{
  position: absolute;
  top: calc(80% - 200px);
  left: calc((100% - 800px)/2);
  right: calc((100% - 800px)/2);
  z-index: 3;
  width: 800px;
  height: 264px;
  font-family: Montserrat_medium;
  font-size: 20px;
  float: left;
 }

 .block .about h2{
  width: 400px;
  color: #f1f1f1;
  margin-bottom: 15px;
  font-family: Montserrat_medium;
  font-size: 2.5em;
  float: left;
 }

 .block .about p{
  width: 500px;
```

```
  color: #f1f1f1;
  margin-bottom: 15px;
  font-family: Montserrat_light;
  font-weight: 300;
  font-size: 1em;
  float: left;
}

.block .about div.button{
  width: 120px;
  height: 14px;
  padding: 11px 14px;
  margin: 10px 0px;
  text-align: center;
  border: 2px solid #ffca00;
  border-radius: 50px;
  overflow: hidden;
  position: relative;
  float: right;
}

.block .about div.button:before{
  width: 100%;
  height: 100%;
  content: "";
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 1;
  background-color: #ffca00;
}

.block .about div.button:hover:before{
  width: 0px;
}

.block .about div.button span{
  width: 100%;
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 2;
  color: #1e1f23;
  padding: 11px 0px;
  font-size: 14px;
  font-family: Montserrat_medium;
  font-weight: 600;
  line-height: 16px;
  text-align: center;
```

```
}

.block .about div.button:hover span{
  color: #ffca00;
}

/*second button*/

.block .about div.button.secondr{
  border: 2px solid #f1f1f1;
}

.block .about div.button.secondr:before{
  background-color: #f1f1f1;
}

.block .about div.button.secondr:hover span{
  color: #f1f1f1;
}

#content{
  position: relative;
  width: 100%;
  height: auto;
  background-color: #1e2024;
  color: #f1f1f1;
  float: right;
}

footer{
  position: absolute;
  bottom: 0px;
  left: 0px;
  right: 0px;
  z-index: 100;
  width: 100%;
  padding: 15px 0px;
  text-align: center;
  background-color: #27292d33;
  font-family: Montserrat_medium;
  color: #f1f1f1bb;
  float: left;
}

footer a{
  font-family: Montserrat_medium;
  color: inherit;
}
```

```
footer a:hover{
  color: #ffca00;
}

/* --- medias --- */

@media (max-width: 1150px) {
  .sidebar.sidebar_full{
    width: 35%;
    position: fixed;
    border-right: 2px solid #ffca00;
  }

  #content.sidebar_full{
    width: 100%;
  }

  .sidebar{
    background-color: #000000fa;
    border-right: 0px solid #ffca00;
  }
}

@media (max-width: 900px) {
  header img{
    margin: 0px 20px;
  }

  header .dropdown{
    margin: 0px 15px;
  }

  header .dropdown_content{
    background-color: #00000088;
  }

  header .dropdown_content a{
    background-color: #f1f1f111;
  }

  .block .about{
    width: calc(100% - 100px);
    top: 30vh;
    left: 50px;
    right: 50px;
  }

  .block .about h2, .block .about p{
    width: 100%;
```

```
    text-align: center !important;
  }

  .block .about p{
    margin-bottom: 45px;
  }

  .block .about .button{
    margin: 10px calc((100% - 152px) / 2) !important;
  }

  .sidebar.sidebar_full{
    width: 50%;
    position: fixed;
  }

  #content.sidebar_full{
    width: 100%;
  }
}

@media (max-width: 500px) {
  header img{
    height: 36px;
    padding: 22px 0px;
    margin: 0px 15px;
  }

  header .dropdown{
    margin: 0px 10px;
    width: 100px;
  }

  header .dropdown_btn{
    font-size: 10px;
    height: 10px;
    padding: 35px;
  }

  header .dropdown_content a{
    font-size: 10px;
    height: 10px;
  }

  .block .about h2{
    font-size: 2.2em;
  }

  .block .about p{
```

```
    margin-bottom: 20px;
  }

  .block .about .button{
    margin: 5px calc((100% - 152px) / 2) !important;
  }

  .sidebar.sidebar_full{
    width: 100%;
    border: none;
  }

  .sidebar{
    height: calc(100vh - 100px) !important;
    background-color: #000000;
  }

  footer {
    font-size: 10px;
    bottom: 0px;
    height: auto;
  }
}

@media (max-height: 700px) {
  .block .about{
    top: 130px;
    font-size: 14px;
  }
}  @font-face {
 font-family: deleteMind;
 src: url(../fonts/deleteMind.woff);
}

@font-face {
 font-family: Montserrat_thin;
 src: url(../fonts/Montserrat/Montserrat-Thin.ttf);
}

@font-face {
 font-family: Montserrat_light;
 src: url(../fonts/Montserrat/Montserrat-Light.ttf);
}

@font-face {
 font-family: Montserrat_medium;
 src: url(../fonts/Montserrat/Montserrat-Medium.ttf);
}
```

```
@font-face {
 font-family: Montserrat_black;
 src: url(../fonts/Montserrat/Montserrat-Black.ttf);
}

html {
  height: 100%;
  width: 100%;
}

body {
  height: 100%;
  width: 100%;
  font-family: Montserrat;
  font-weight: 100;
  background-color: #27292d;
  cursor: default;
}

* {
  margin: 0;
  padding: 0;
  outline: none;
  font-family: Montserrat;
  transition: 0.5s;
}

*:before, *:after {
  transition: 0.3s;
}

*::selection{
  background: #a1a1a166;
}

p {
  font-family: Montserrat;
}

h1{
  font-size: 5em;
  }

h2 {
  font-size: 4em;
}

h3 {
```

```
  font-size: 3em;
}

h4{
  font-size: 2.5em;
}

h5{
  font-size: 2em;
}

h6 {
  font-size: 1.7em;
}

a, a:hover {
  text-decoration: none;
}

ul li, li {
  list-style-type: none;
}

::-webkit-scrollbar{
  width: 3px;
  height: 3px;
  background-color: #27292d;
}

::-webkit-scrollbar-thumb{
  width: 3px;
  height: 3px;
  background-color: #ffca00;
}

/*sidebar*/

.sidebar{
  position: fixed;
  z-index: 1000;
  left: 0px;
  bottom: 0px;
  width: 0px;
  height: calc(100vh - 100px);
  padding: 50px 0px;
  background-color: #151618;
  overflow: hidden;
}
```

```
.sidebar .sidebar_content{
  width: 300px;
  margin: 100px 0px;
  float: left;
}

.sidebar .sidebar_content .link{
  width: 100%;
  height: 46px;
  float: left;
  transition: 0.5s;
}

.sidebar .sidebar_content .link a{
  position: relative;
  height: 16px;
  margin: 18px 30px;
  color: #f1f1f188;
  font-family: Montserrat_light;
  font-weight: 300;
  float: left;
}

.sidebar .sidebar_content .link a:hover{
  color: #f1f1f1;
}

.sidebar .sidebar_content .link a:after{
  float: right;
  content: "\2192";
  font-size: 24px;
  font-family: Montserrat_light;
  padding-left: 50px;
  line-height: 16px;
  opacity: 0;
  transform: scaleX(3);
  color: #ffca00;
  transition: 0.3s;
}

.sidebar .sidebar_content .link:hover a:after{
  padding-left: 10px;
  opacity: 1;
  transform: scaleX(1.4);
}

.sidebar.sidebar_full{
  width: 300px;
}
```

```
#content.sidebar_full{
  width: calc(100% - 300px);
}

header{
  z-index: 1001 !important;
}

header .sidebar_button{
  width: 18px;
  height: 18px;
  margin: 30px 35px;
  position: relative;
  cursor: pointer;
  float: left;
}

header .sidebar_button nav{
  width: 18px;
  height: 2px;
  margin: 2px 0px;
  background-color: #f1f1f1;
  transition: 0.5s ease;
  float: left;
}

.loader_screen{
  width: 100%;
  height: 100%;
  background-color: #27292d;
  position: fixed;
  top: 0px;
  bottom: 0px;
  left: 0px;
  right: 0px;
  z-index: 9999;
}

.loader{
  width: 20px;
  height: 20px;
  margin: calc(50vh - 20px) auto;
}

.loader_round {
  width: 18px;
  height: 18px;
  border: 1px solid #f1f1f1;
```

```
  border-top: 1px solid transparent;
  border-radius: 50px;
  display: inline-table;
  -webkit-animation: spin 0.5s linear infinite;
  animation: spin 0.5s linear infinite;
}

@keyframes spin {
  0% {transform: rotate(0deg);}
  100% {transform: rotate(360deg);}
}

.loaded{
  display: none;
}
```

design.js

```
window.onload = function() {
    document.getElementById("loader_screen").style.display = "none";
}

const sidebar = document.getElementById("sidebar");
const content = document.getElementById("content");
const round = document.getElementById("light_round");

var check = true;

function openCloseSidebar() {
    if (check) {
        check = false;
        sidebar.classList.add("sidebar_full");
        content.classList.add("sidebar_full");
    } else {
        check = true;
        sidebar.classList.remove("sidebar_full");
        content.classList.remove("sidebar_full");
    }
}

function lighterMoving() {
    round.style.marginTop = event.clientY - 200 + "px";
    round.style.marginLeft = event.clientX - 200 + "px";
}
```

graphs.js

```
/* --- main sourse file --- */

var OPoint = [0, 0]; //start point of coordinates
```

```
var scale = 1; //scaling of coordinate plane
var step = 30; //step of web

var graphColors = ["#ffca00","#d67400","#005bff","#e20000","#70cc17","#12cca7","#7c23d6","#ba22d6"]; //color of graphs

var funcStringArray = []; //array of functions (strings)
var func = []; //array of functions, which going to compile
var funcZero = document.getElementById("function_zeroes"); //output of function's zeroes

var intMin = document.getElementById("graph_int_min").value; //input of interval (start)
var intMax = document.getElementById("graph_int_max").value; //input of interval (finish)

window.onload = function() { //if page load
   graphButtonClick();
}

function addSymbol(symbol) {
   switch (symbol) {
      case 'abs':
         document.getElementById("func_field").value += "abs(x)";
         break;
      case 'radical':
         document.getElementById("func_field").value += "sqrt(x)";
         break;
      case 'power':
         document.getElementById("func_field").value += "^";
         break;
      case 'eyler':
         document.getElementById("func_field").value += "e";
         break;
      case 'pi_num':
         document.getElementById("func_field").value += "pi";
         break;
      case 'sin':
         document.getElementById("func_field").value += "sin(x)";
         break;
      case 'cos':
         document.getElementById("func_field").value += "cos(x)";
         break;
      case 'tg':
         document.getElementById("func_field").value += "tg(x)";
         break;
      case 'ctg':
         document.getElementById("func_field").value += "ctg(x)";
         break;
      case 'arcsin':
         document.getElementById("func_field").value += "arcsin(x)";
```

```
            break;
        case 'arccos':
            document.getElementById("func_field").value += "arccos(x)";
            break;
        case 'arctg':
            document.getElementById("func_field").value += "arctg(x)";
            break;
        case 'arcctg':
            document.getElementById("func_field").value += "arcctg(x)";
            break;
        case 'ln':
            document.getElementById("func_field").value += "ln(x)";
            break;
        case 'lg':
            document.getElementById("func_field").value += "lg(x)";
            break;
        case 'log':
            document.getElementById("func_field").value += "log(x, b)";
            break;
        case 'sh':
            document.getElementById("func_field").value += "sh(x)";
            break;
        case 'ch':
            document.getElementById("func_field").value += "ch(x)";
            break;
        case 'th':
            document.getElementById("func_field").value += "th(x)";
            break;
        case 'cth':
            document.getElementById("func_field").value += "cth(x)";
            break;
        case 'clear':
            document.getElementById("func_field").value = "";
            graphButtonClick();
    }
}

function changeScale(coef) { //scale change
    if (coef == 0) {
        scale = 1;
    } else if (coef < 0) {
        if (scale <= 2) {
            scale *= 2;
            OPoint[0] = (scale * OPoint[0]).toFixed(3);
            OPoint[1] = (scale * OPoint[1]).toFixed(3);
            scale = scale.toFixed(3);
        } else {
            return;
        }
```

```
    } else {
        if (scale >= 0.25) {
            scale /= 2;
            OPoint[0] = (OPoint[0] / scale).toFixed(3);
            OPoint[1] = (OPoint[1] / scale).toFixed(3);
            scale = scale.toFixed(3);
        } else {
            return;
        }
    }
    graphDraw();
}

function changeCenter(code) { //center position change
    switch (code) {
        case "center":
            OPoint[0] = 0;
            OPoint[1] = 0;
            break;
        case "top":
            OPoint[1] = parseInt(OPoint[1]) - step;
            break;
        case "right":
            OPoint[0] = parseInt(OPoint[0]) + step;
            break;
        case "left":
            OPoint[0] = parseInt(OPoint[0]) - step;
            break;
        case "bottom":
            OPoint[1] = parseInt(OPoint[1]) + step;
    }
    graphDraw();
}

function graphButtonClick() { //if button "Build" was click
    func = [];
    funcZero.innerHTML = "";
    var fs = document.getElementById("func_field").value;
    intMin = document.getElementById("graph_int_min").value;
    intMax = document.getElementById("graph_int_max").value;
    for (var i = 0; fs.length != 0; i++) {
        var index = fs.indexOf(";");
        if (index == -1) {
            index = fs.length;
        }
        funcStringArray[i] = fs.substring(0, index);
        func[i] = math.compile(fs.substring(0, index));
        fs = fs.substring(index + 1, fs.length);
    }
```

```
    graphDraw();
}

function graphDraw() { //work with canvas
    document.getElementById("fz_title").innerHTML = " [" + intMin + "; " + intMax + "]:";
    funcZero.innerHTML = "";

    var my_canvas = document.getElementById("canvas");
    var c = my_canvas.getContext("2d");
    c.clearRect(0, 0, 635, 635);
    my_canvas.width = 635;
    my_canvas.height = 635;
    c.translate(300, 300);
    c.lineWidth = 0.5;

    //web drawing

    c.beginPath();
    for (var x = -300 + OPoint[0] % step; x <= 300; x += step) {
        c.moveTo(x, -300);
        c.lineTo(x, 300);
    }

    for (var y = -300 + OPoint[1] % step; y < 301; y += step) {
        c.moveTo(-300, y);
        c.lineTo(300, y);
    }
    c.strokeStyle = "#3f3f3f";
    c.closePath();
    c.stroke();

    //axises

    c.beginPath();
    if (Math.abs(OPoint[0]) <= 300) {
        c.moveTo(OPoint[0], -300);
        c.lineTo(OPoint[0], 300);
    }
    if (Math.abs(OPoint[1] <= 300)) {
        c.moveTo(300, OPoint[1]);
        c.lineTo(-300, OPoint[1]);
    }

    //scale values (for x)
    for (var x = OPoint[0] % step - 300; x < 290; x += step) {
        if (!(x < -290)) {
            c.font = "8pt monospace";
            c.fillStyle = "#aaaaaa";
            c.fillText(1 * ((x - OPoint[0]) * scale / step).toFixed(2), x - 3, 313);
```

```
    }
}

//scale values (for y)
for (var y = OPoint[1] % step - 300; y < 290; y += step) {
    if (!(y < -290)) {
        c.font = "8pt monospace";
        c.fillStyle = "#aaaaaa";
        c.fillText(1 * (-(y - OPoint[1]) * scale / step).toFixed(2), 307, y + 3);
    }
}
c.strokeStyle = "#dddddd";
c.closePath();
c.stroke();

//drawing graphs

for (var i = 0; i < func.length; i++) {
    c.beginPath();
    c.lineWidth = 2;
    c.lineJoin = 'round';
    var checkerMin = 0;
    var checkerMax = 0;
    var spacing = 0.1; //step of "increment"
    var y = 0, x = 0;
    var cx = [];
    var cy = [];
    var mathFns = { //special object for math.js library
        x: 0,
        tg: math.tan,
        ctg: math.cot,
        ln: math.log,
        arcsin: math.asin,
        arccos: math.acos,
        arctg: math.atan,
        arcctg: arcctg,
        lg: math.log10,
        sh: math.sinh,
        ch: math.cosh,
        th: math.tanh,
        cth: math.coth
    };

    if (document.getElementById("cut_graph").checked) {
        checkerMin = parseInt(intMin) / scale * step + parseInt(OPoint[0]);
        checkerMax = parseInt(intMax) / scale * step + parseInt(OPoint[0]);
    } else {
        checkerMin = -300;
        checkerMax = 300;
```

```
            }
            for (cx[0] = checkerMin; cx[0] <= checkerMax; cx[0] += spacing) { //draw graphs
               mathFns.x = (cx[0] - OPoint[0]) * scale / step;
               y = func[i].eval(mathFns);
               cy[0] = (-y / scale * step + parseInt(OPoint[1]));
               if (Math.abs(cy[0]) <= 300 && Math.abs(cx[1]) <= 300) {
                  c.moveTo(cx[1], cy[1]);
                  c.lineTo(cx[0], cy[0]);
               }
               cy[1] = cy[0];
               cx[1] = cx[0];
            }

            c.strokeStyle = graphColors[i % 8]; //set color
            c.stroke();
            spacing = step; //update spacing to step of web
            y = 0;
            x = 0;
            cx = [];
            cy = [];
            var w = 3; //check scale for change points size

            if (scale >= 4) {
               w = 2;
            }

            if (document.getElementById("show_points").checked) { //draw points
               for (cx[0] = intMin * step / scale + parseInt(OPoint[0]); cx[0] <= (parseInt(intMax) +
1) * step / scale + parseInt(OPoint[0]); cx[0] += spacing / scale) {
                  mathFns.x = (cx[0] - OPoint[0]) * scale / step;
                  y = func[i].eval(mathFns);
                  cy[0] = (-y * step / scale + parseInt(OPoint[1]));
                  if (Math.abs(cy[1]) <= 300 && Math.abs(cx[1]) <= 300) {
                     c.beginPath();
                     c.arc(cx[1], cy[1], w, 0, Math.PI*2, true);
                     c.moveTo(cx[0], cy[0]);
                     c.closePath();
                     c.fillStyle = "#aaaaaa";
                     c.fill();
                  }
                  cy[1] = cy[0];
                  cx[1] = cx[0];
               }
            }

            if (document.getElementById("show_analysis").checked) { //counting function zeroes
               var a = 0;

               document.getElementById("analysis_field").style.display = "block";
```

```
                funcZero.innerHTML += "<nav style='background: " + graphColors[i % 8] +
";'></nav>";
                funcZero.innerHTML += (i + 1) + ") " + funcStringArray[i] + ": ";
                for (cx[0] = intMin * step; cx[0] <= intMax * step; cx[0]++) {
                    mathFns.x = cx[0];
                    y = func[i].eval(mathFns);
                    cy[0] = -y;
                    if (cy[0] == 0) {
                        funcZero.innerHTML += cx[0] + ", ";
                        a++;
                    }
                    cy[1] = cy[0];
                    cx[1] = cx[0];
                }

                if (a == 0) {
                    funcZero.innerHTML += "none;" + "<br>"; //if there are not zeroes
                } else {
                    funcZero.innerHTML = funcZero.innerHTML.slice(0, -2); //slicing two last
symbols (, )
                    funcZero.innerHTML += "; " + "<br>";
                }

            } else {
                document.getElementById("analysis_field").style.display = "none";
            }

        }

        //drawing area beyond the interval

        if (document.getElementById("cut_graph").checked) {
            c.beginPath();
            c.lineWidth = 1;
            c.lineJoin = 'round';
            c.setLineDash([5]);
            if (intMin / scale * step + parseInt(OPoint[0]) <= 300) {
                c.moveTo(intMin / scale * step + parseInt(OPoint[0]), -300);
                c.lineTo(intMin / scale * step + parseInt(OPoint[0]), 300);
            }

            if (intMax / scale * step + parseInt(OPoint[0]) <= 300) {
                c.moveTo(intMax / scale * step + parseInt(OPoint[0]), -300);
                c.lineTo(intMax / scale * step + parseInt(OPoint[0]), 300);
            }
            c.strokeStyle = "#e20000";
            c.stroke();

            c.beginPath();
```

```
            c.fillStyle = "#e2000011";

            if (intMin / scale * step + parseInt(OPoint[0]) <= 300) {
               c.fillRect(-300, -300, 300 + intMin / scale * step + parseInt(OPoint[0]), 600);
            } else {
               c.fillRect(-300, -300, 600, 600);
            }

            if (intMax / scale * step + parseInt(OPoint[0]) <= 300) {
               c.fillRect(intMax / scale * step + parseInt(OPoint[0]), -300, 300 - intMax / scale * step
- parseInt(OPoint[0]), 600);
            }
         }

         function arcctg(x) {
            return Math.PI / 2 - math.atan(x);
         }
      }

      mathjs.js

      (function webpackUniversalModuleDefinition(root, factory) {
            if(typeof exports === 'object' && typeof module === 'object')
                  module.exports = factory();
            else if(typeof define === 'function' && define.amd)
                  define([], factory);
            else if(typeof exports === 'object')
                  exports["math"] = factory();
            else
                  root["math"] = factory();
      })(this, function() {
      return /******/ (function(modules) { // webpackBootstrap
      module.exports = function deepMap(array, callback, skipZeros) {
        if (array && typeof array.map === 'function') {
          // TODO: replace array.map with a for loop to improve performance
          return array.map(function (x) {
            return deepMap(x, callback, skipZeros);
          });
        } else {
          return callback(array);
        }
      };

      /***/ }),
      /* 1 */
      /***/ (function(module, exports, __webpack_require__) {

      "use strict";
```

```
function factory(type, config, load, typed) {
  var matrix = typed('matrix', {
    '': function _() {
      return _create([]);
    },
    'string': function string(format) {
      return _create([], format);
    },
    'string, string': function stringString(format, datatype) {
      return _create([], format, datatype);
    },
    'Array': function Array(data) {
      return _create(data);
    },
    'Matrix': function Matrix(data) {
      return _create(data, data.storage());
    },
    'Array | Matrix, string': _create,
    'Array | Matrix, string, string': _create
  });
  matrix.toTex = {
    0: '\\begin{bmatrix}\\end{bmatrix}',
    1: '\\left(${args[0]}\\right)',
    2: '\\left(${args[0]}\\right)'
  };
  return matrix;

  function _create(data, format, datatype) {
    // get storage format constructor
    var M = type.Matrix.storage(format || 'default'); // create instance

    return new M(data, datatype);
  }
}

exports.name = 'matrix';
exports.factory = factory.
```

Додаток Б

ВІДГУК

керівника економічного розділу

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника економічної частини Професора Вагонової О.Г.
(прізвище, ім'я, по батькові, вчене звання)

на магістерську роботу

Студента ІІ курсу групи 121м-20-1 Ніколайчука Ігоря Олександровича
(прізвище, ім'я, по батькові)

На тему: Розробка web-застосунку автоматичного тегування та категоризації зображень на основі згорткових нейронних мереж

.

«___»_____2022 р. _____
(підпис)

Додаток Д

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Ніколайчук.doc	Пояснювальна записка кваліфікаційної роботи. Документ Word.
Ніколайчук.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація роботи