Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ОНЛАЙН-МАГАЗИНОМ З ВПРОВАДЖЕННЯМ ТЕХНОЛОГІЙ РОЗПІЗНАВАНЬ ЗОБРАЖЕНЬ ТА ШТУЧНОГО ІНТЕЛЕКТУ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-3
Федорук М. Ю.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О. А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Федоруку Матвію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для управління онлайн-магазином з впровадженням технологій розпізнавань зображень та штучного інтелекту

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 26 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, інтернет джерела, методики створення згорткових нейронних мереж ,платформа для створення тестового набору даних Kaggle, мови програмування Python, JavaScript, фреймворк Node.js, система управління базами даних MongoDB, мовні моделі GPT, Gemini, архітектура нейронних мереж ResNet.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів для створення згорткових нейронних мереж для класифікації зображень.
2. Огляд існуючих застосунків для розпізнавання типів одягу і брендів.
3. Проведення тестування мовних моделей.
4. Вибір мовних моделей і архітектур нейронних моделей.
5. Проєктування компонентів застосунку.
6. Програмна реалізація.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність сфери використання технологій аналізу зображень, сучасні технології для аналізу зображень, постановка задачі, аналіз роботи нейронних мереж, формальний опис процесів виявлення одягу на зображенні і роботи текстових моделей, проблеми аналізу і класифікації, проєктування застосунку, програмна реалізація, демонстрація застосунку.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Кобилін О. А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 6 табл., 17 рис., 32 джерела.

АНАЛІЗ ЗОБРАЖЕНЬ, НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ОДЯГУ, КЛАСТЕРИЗАЦІЯ, ТЕКСТОВІ ПІДКАЗКИ, СТВОРЕННЯ ОПИСУ.

Об’єктом роботи є різні методи для класифікації зображень і логотипів на зображенні за допомогою створення спеціальної моделі.

Метою роботи є створення вебзастосунку, який покращує якість управління онлайн-магазинами за допомогою створення задач і отримання підказок з використанням штучного інтелекту, а також створення нейронної мережі, яка надає можливість аналізувати зображення і виявляти тип одягу, його бренд і генерувати опис на основі цих даних.

Використано методи аналізу зображень, включаючи створення згорткової нейронної мережі для класифікації. Підключено наявні моделі для створення підказок і описів.

У результаті було розроблено вебзастосунок для управлінням задачами онлайн-магазину з використанням штучного інтелекту і технологій розпізнавань зображень.

IMAGE ANALYSIS, NEURAL NETWORKS, CLOTHING CLASSIFICATION, CLUSTERIZATION, TEXT HINTS, DESCRIPTION CREATION.

The object of the work is various methods for classifying images and logos in an image by creating a special model.

The aim of the work is to create a web application that improves the quality of online store management by creating tasks and receiving hints using artificial intelligence, as well as creating a neural network that provides the ability to analyze images and detect the type of clothing, its brand and generate a description based on this data.

Image analysis methods were used, including the creation of a convolutional neural network for classification. Existing models were connected to create hints and descriptions.

As a result, a web application was developed to manage online store tasks using artificial intelligence and image recognition technologies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network (згорткові нейронні мережі)

GPU – Graphics Processing Unit (графічний процесор)

LLM – Large Language Model (велика мовна модель)

JS – JavaScript

БД – база даних

YOLO – You Only Look Once

ВСТУП

Стрімкий розвиток нейронних моделей і глибинного навчання відкриває нові горизонти у сфері аналізу зображень, зокрема в галузі автоматичної класифікації типу одягу та розпізнавання бренду за логотипом. Ця кваліфікаційна робота зосереджена на важливому аспекті цієї стрімкої технологічної індустрії – розробці вебзастосунку для управлінням задачами онлайн магазину з використанням штучного інтелекту, а також створення нейронної моделі для ідентифікації типів одягу та виявленні логотипів популярних брендів на зображеннях. Метою роботи є створення ефективної моделі глибокого навчання, здатної виконувати це завдання задля покращення можливості управляти онлайн магазинами.

Методологія дослідження поєднує в собі аналіз наукової літератури, побудову моделі на основі ResNet, використання моделі Gemini, навчання на готових датасетах, а також створення вибірки логотипів. Після реалізації, було проведено валідацію та тестування моделі в умовах реального використання. Основні методи включають глибинне навчання, оборобку зображень, аугментацію даних та багатокласову класифікацію.

Актуальність роботи полягає у збільшенні попиту на системи штучного інтелекту у сфері електронної комерції та застосунків моди. Автоматичне розпізнавання типів одягу, а також класифікації брендів зумовлює покращення взаємодії користувача з онлайн-магазинами, вдосконалює системи пошуку за зображенням, а також відкриває нові можливості для персоналізованої реклами. Таким чином, розробка нейронної моделі, спеціалізованої на класифікації одягу і логотипів є важливою та перспективною потребою в сучасному цифровому середовищі.

1 АКТУАЛЬНІСТЬ І СФЕРА ЗАСТОСУВАННЯ

1.1 Еволюція технологій аналізу зображень в реальному часі

З розвитком штучного інтелекту, комп'ютерного зору та моделей глибокого навчання за останнє десятиліття значно змінилось ставлення до аналізу зображень у реальному часі. Цей стрімкий розвиток показує не тільки технологічний прогрес, а і підкреслює збільшення попиту на обробку зображень в різних застосунках [1]. Метою цього розділу є підкреслення актуальності розвитку технологій у сфері аналізу зображень і засобів, що дозволяють його покращити.

Перші підходи в аналізі одягу базувались на класичних алгоритмах машинного навчання, таких як метод опорних векторів, кластеризації або rule-based систем. Але ці підходи обмежувались в точності і масштабованості, а також вимагали великої кількості роботи для побудови ознак.

Справжній прорив у галузі став можливим з появою згорткових нейронних мереж (CNN), які дали змогу моделі навчатись самостійно на основних ознаках без потреби їх ручного створення. Зокрема, такі архітектури як AlexNet та ResNet та їх модифікації стали стандартом у класифікації зображень, включаючи ідентифікацію одягу і логотипів на ньому. Ці архітектури продемонстрували високу якість і точність даних, навіть на великих і різновидних наборах даних, таких як Mnist чи Deep Fashion.

Логічним продовженням досліджень у даній галузі стало створення моделей для розпізнавання брендів одягу за логотипами. Адже логотип – це ключовий елемент в класифікації ідентичності товару [2]. У завданнях розпізнавання бренду дуже велику увагу приділяють локалізації логотипу, його розміру і орієнтації. З часом моделі почали розрізняти логотипи на все більш складних картинках, що зумовлено низькою якістю, частковим його перекриттям.

Сьогодні системи комп’ютерного зору використовуються в онлайн-магазинах в режимі реального часу, наприклад, для пошуку подібного одягу, або ж в автоматизованих системах для інвентаризації і безпеки в торгівлі. Отже, це свідчить про те, що технології для аналізу зображень вже набули свого значення в різних сферах, але з покращенням якості виникають і нові виклики, наприклад, потреба у нових і більших датасетах з брендованим одягом, великою кількістю різноманітних логотипів, а також адаптація до нових, або менш відомих брендів.

Крім того, дуже важливим етапом у еволюції моделей комп’ютерного зору стала поява потужних графічних процесорів (GPU), що дало можливість підтримувати і розгортати складні моделі, на кшталт ResNet або EfficientNet [3], оскільки вони потребують багато ресурсів під час навчання. Таке обладнання, в поєднанні з складними алгоритмами дало змогу стрімко розвивати сферу аналізу зображень

Еволюція технологій також охопила поступовий перехід від простих задач, до комплексних задач по розпізнаванню зображень. Мультизадачні моделі мають змогу одночасно аналізувати тип одягу, його колір, фактуру, стиль і навіть сезонність [4], що дає змогу використовувати ці технології в багатьох сферах діяльності, від ведення онлайн магазинів, до безпеки, де під час відеоспостереження є можливість розпізнавати одяг в реальному часі.

Отже, розвиток технологій аналізу зображень є продовженням загальної еволюції технологій комп’ютерного зору та відповідає нинішнім тенденціям цифровізації візуального контенту. Під час розвитку штучного інтелекту і апаратного забезпечення, технологія аналізу одягу і їх брендів стала результатом цієї еволюції, що заклало основу для розуміння нинішнього стану технологій і потенціального вектору розвитку. При цьому цей напрям залишається перспективним і динамічним, адже з кожним роком попит на інтелектуальну обробку візуальних даних стрімко зростає і набуває все більшої важливості в різних сферах діяльності. Попит на технології штучного інтелекту є зараз практично всюди, тож обробка і аналіз зображень також мають широкий спектр застосування в різноманітних галузях.

1.2 Важливість класифікації одягу і розпізнавання брендів у сучасному середовищі

Поява технологій штучного інтелекту кардинально змінила способи споживання та створення візуального контенту. У сферах медіа, маркетингу, онлайн-шопінгу, соціальних мережах, розвагах і навіть безпеці ці технології стали важливими інструментами, які забезпечують автоматичну обробку зображень і відео, зменшуючи кількість рутинних процесів, при цьому підвищуючи ефективність аналізу контенту та надаючи нові можливості для взаємодії з користувачем.

Однією з важливих переваг автоматичного розпізнавання одягу є можливість швидко ідентифікувати тип одягу по фото або відео – наприклад, визначити чи це куртка, худі або футболка. В поєднанні з алгоритмами розпізнавання логотипу, з’являється можливість визначити бренд, що дає можливість перевірити одяг на оригінальність, побудувати систему рекомендацій для користувача, налаштувати фільтри пошуку, або ж провести автоматичну інвентаризацію. Ці технології особливо важливі в сфері електронної комерції, адже вони підвищують якість взаємодії з користувачами, покращують ефективність і значно збільшують конверсії продажів [5, 6].

Не менш важливим є значення технологій аналізу зображень для захисту великих брендів і інтелектуальної власності. Під час глобалізації ринку та поширенню підробок різних великих брендів, важливо мати автоматизовані системи для розпізнавання оригінальності бренду, щоб забезпечити безпеку у великих компаніях. Для цього вони інвестують значні кошти у подібні системи, які здатні відслідкувати несанкціоноване використання їх логотипів в рекламах, або на різних торгівельних платформах.

Крім того, як і у випадку з іншими задачами комп’ютерного зору, важливе місце має етичність: зокрема, конфіденційність зображень користувачів, а також неупередженість моделей в сторону певних брендів або компаній. Тож під час створення застосунку для управління онлайн-магазином важливо враховувати

прозорість, контрольованість і безпечність використання цих систем має бути пріоритетом під час масового використання.

Також, дуже важливою складовою таких моделей є здатність адаптувати їх до роботи в реальному часі. Інтеграція цих технологій у відеопотоки або мобільні застосунки значно розширюють сфери використання для аналізу зображень. Але при цьому виникає потреба у поєднанні точності, швидкості і легкості моделей. Це сприяє розвитку створення різних стратегій для оптимізацій, таких як квантизація, або використання мобільних архітектур, таких як MobileNet.

Отже, класифікація типів одягу і розпізнавання є не лише цікавою технічною задачею, а й важливою стратегічною технологією для різноманітних сфер діяльності. Вона є не лише корисною для оптимізації процесів [7, 8], а й для покращення взаємодії користувачів з цифровим простором, зробивши її безпечною, зручною і легкою в розумінні.

1.3 Сфери використання

У сфері маркетингу розпізнавання логотипів стає потужним інструментом для оцінки популярності бренду серед користувачів у різних статтях, відео або ж на публічних виступах. Завдяки цьому, маркетологи можуть автоматично отримувати інформацію про активність навколо бренду, виявляти незаконне використання логотипів, або оцінювати ефективність проведених рекламних кампаній. Це дозволяє великим компаніям будувати нові стратегії, спираючись на основі даних, отриманих з наявних джерел.

Також слід розглянути технології розпізнавання одягу у сфері безпеки та відеоспостережень. Автоматичне розпізнавання одягу в реальному часі може використовуватись для ідентифікації підозрілих осіб, виявлення людей у натовпі а також знаходження специфічних видів одягу (наприклад, уніформ). Водночас логотипи певних брендів можуть стати своєрідними маркерами, що дозволяє зорієнтуватись в контексті ситуації.

Крім того, у сфері логістики аналіз зображень теж має важливе значення. Автоматизовані системи інвентаризації і сортування значно спрощують і пришвидшують багато процесів в цій галузі. Також при використанні цих технологій компанії економлять значні кошти, оскільки зменшується потреба в використанні великої кількості людей. Це особливо важливо для великих брендів, або маркетплейсів, які мають широкий асортимент продукції.

В галузі екології аналіз зображень теж набув широкого використання. Під час сортування різних відходів, автоматизовані системи для аналізу зображень надають змогу визначити тип і склад матеріалу, що прискорює процес сортування і дає можливість правильно відсортувати і переробити одяг, зменшивши вплив на оточуюче середовище.

Також різні інструменти для аналізу одягу можуть використовуватись в освіті і різних процесах навчання. Технології для розпізнавання одягу і логотипів дають можливість студентам краще зрозуміти принципи стилізації одягу. А також, на основі даних з’являється можливість створити різноманітні освітні програми і матеріали, які містять інформацію про дизайн і бренди одягу [9, 10].

Технології розпізнавання логитипів можуть використовуватись в спорті, аналізуючи різні банери, спортивну форму і різний контрактний одяг певних брендів. Після чого може створюватись аналітика і розрахунок ціни на рекламну кампанію для певного бренду і оптимізувати їх вартість, або можна провести аналіз ефективності певного спонсора [9, 10].

Не дивлячись на всі успіхи і переваги впровадження технологій розпізнавання одягу супроводжується появою складних задач і викликів – наприклад, складністю обробки різних зображень з нетиповими кутами зйомки, освітленням, варіацією стилів. У поєднанні з потребою розпізнавати логотипи, які можуть бути спотворені, закриті або стилізовані, виникає потреба у створенні стійких і адаптивних моделей. Для цього потрібно враховувати всі потреби і проблеми, які з’являються в процесі розробки моделей і шукати баланс у вирішенні їх. Для цього потрібно використовувати сучасні методи і алгоритми програмування.

1.4 Сучасні технології для аналізу зображень і їх обмеження

Сучасні технології для класифікації одягу та розпізнавання логотипів досягли значного прогресу завдяки розвитку глибинного навчання і комп'ютерного зору, але вони все ще зазнають нових обмежень і викликів. Але розуміння цих проблем, в поєднанні з сучасними технологіями дозволяє досягати успіху і вдосконалювати методи для класифікації одягу.

Нині основу таких технологій складають згорткові нейронні мережі (CNN), які демонструють високу точність розпізнавання. Вони навчаються на великих об’ємах даних, які містять різні види зображень, це дає їм змогу вдало знаходити одяг і класифікувати його. Але, ці моделі стикаються з рядом проблем при розпізнаванні, на що може впливати якість зображення, його розмір, освітлення, на зображенні, задній фон. При відеозйомці проблеми виникають через динамічність світла, або рухливі об’єкти. Крім того, моделі які використовуються в реальному часі мають потребу у великій кількості ресурсів, що ускладнює роботу [11, 12]. Також виникає проблема в розпізнаванні логотипів, оскільки навіть сучасні моделі не можуть розпізнати точно зім’яті або зіпсовані логотипи. Ще одною важливою проблемою цих моделей є потреба у великій кількості даних для навчання. Оскільки навчання проводиться на обмеженій кількості даних, під час аналізу виникають проблеми на нетипових видах одягу, або невідомого логотипу.

Різні архітектури типу ResNet, EfficientNet дуже добре оптимізовані для аналізу зображень і виявленні логотипів. Використання цих архітектур дозволяє набагато покращити аналіз зображень і на основі них створити власну модель, яка буде якісніше обробляти різні види зображень. Також ці архітектури є безкоштовними, що робить їх більш підходящими.

Значним проривом у розпізнаванні об’єктів є алгоритм YOLO (You Only Look Once), який забезпечив аналіз об’єктів в режимі реального часу. Але ця модель може викликати додаткові витрати, що створює певні проблеми в реалізації.

Основними проблемами при аналізі зображень і виявленні логотипів є:

– складність розпізнавання при різних деформаціях і перекриттях, що викликає неправильне розпізнавання логотипу, або навіть типу одягу, в залежності від зображення;

– обмежена здатність розпізнавання. При аналізі використовуються моделі, які були навчені на певних даних, але мода змінюється швидко, що може ускладнити розпізнавання певних елементів, які не використовувались при навчанні моделі;

– висока складність обчислювання обмежує їх використання на мобільних і слабких пристроях. Хоч вже і існують моделі типу MobileNet, але зазвичай доводиться шукати компроміс між точністю і швидкістю;

– необхідність у великих наборах даних для навчання. Якість аналізу залежить від кількості різноманітних зображень, на яких тренується модель, це вимагає великої кількості ресурсів;

– проблема у визначенні оригінальності одягу залишається важливою, оскільки іноді підробку майже неможливо відрізнити від оригіналу, не лише на зображеннях, а й на відео.

Для подолання цих складнощів, використовується мультимодальний підхід в створенні моделей, де поєднуються різні види даних, як візуальні, так і контекстні [13, 14]. Також використовується самонавчання, щоб зменшити обсяги даних і пришвидшити навчання на нерозмічених даних. Додатково залучаються новітні алгоритми і технології, які покращують процес навчання і допомагають у створенні моделей.

Незважаючи на наявні обмеження, технології розпізнавання одягу і логотипів стрімко розвиваються, долаючи існуючі проблеми і відкриваючи нові можливості в різних сферах. Щороку з’являються все більш складні і багатофункціональні моделі, які дають все більше нових можливостей в галузях, де використовується аналіз зображень. Це значно покращує користувацький досвід і також полегшує роботу багатьом професіям. Великі компанії все частіше залучають роботу нейронних моделей у своїх застосунках, оскільки це також змешнує витрати на компанію і значно збільшує обсяг виконаної роботи.

1.5 Нейронні мережі в комп’ютерному зорі

У сфері комп’ютерного зору, нейронні мережі зробили революцію в інтерпретації і візуалізації даних. Цей розділ присвячений огляду роботи і структури нейронних мереж, їх функціонування і застосування у таких задачах, як розпізнавання об’єктів, зокрема виявлення типів одягу і брендів на зображеннях.

Нейронні мережі за принципом роботи імітують людський мозок. Вони складаються з серії алгоритмів, що розпізнають основні взаємозв’язки у наборі даних. Цей процес схожий на операції, який виконує мозок для виконання різних типів задач. У контексті комп’ютерного зору ці мережі призначені для обробки візуальних даних шляхом розпізнавання різних характеристик на зображеннях

Зазвичай, нейронна мережа сладається з різних шарів пов’язаних нейронів або вузлів, кожен з яких виконує певні перетворення над вхіднимпи даними. Існує декілька видів шарів, такі як: вхідний, приховані і вихідний. В кожному шарі існують ваги, які пов’язують вузли і передають сигнали між ними. Сила цих ваг регулюється під час навчання моделі, що впливає на здатність моделі нейронної мережі класифікувати об’єкти або виконувати інші функції, в залежності від типу моделі.

Функція цих згорткових шарів полягає в перетворенні зображень в числові значення, після чого модель має змогу отримати певні шаблони з цих наборів значень. Додатково нейронні моделі мають фільтри, які створюють двовимірні масиви значень, які далі передаються на пізніші шари моделі, ті, які вивчають візерунки в зображеннях.

Для аналізу зображень в задачах комп’ютерного зору використовуються згорткові нейронні мережі (CNN). Принцип їх роботи полягає в проходженні зображення через багато шарів, кожні з яких виконують свою роль. Потім отримана інформація подається в одному, або декількох пов’язаних шарах, де виконується класифікація (рис. 1.1). Операція згортки проходить за допомогою спеціального набору фільтрів, що навчаються [15, 16].

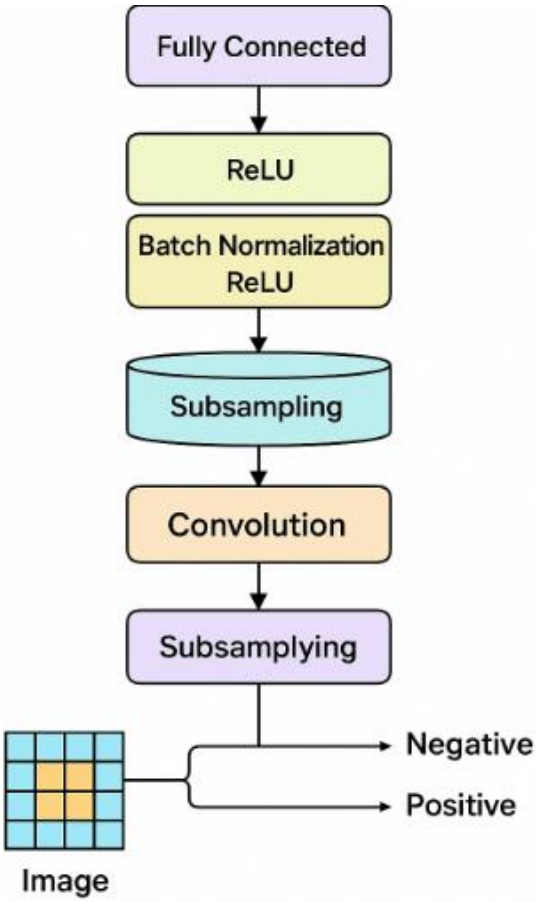

Рисунок 1.1 – Структура згорткових нейронних мереж

Такі моделі досить добре показують себе в роботі з піксельними даними, тому їх використовують в різних задачах, пов’язаних з аналізом зображень, класіфікації, сегментації і розпізнаванні об’єктів. CNN використовують декілька фільтрів для вивчення шаблонів, в залежності від їх кількості, модель матиме більше можливостей для перевірки вхідних даних і навчанню. Для розпізнавання меж об’єктів, CNN аналізує різницю в значенні пікселів. Якщо на вході модель отримує чорно-білу картинку, використовується один канал кольорів, але при різнокольорових зображеннях, використовується 3 канали кольорів – червоний, зелений і синів. Кількість каналів у фільтрі називають глибиною фільтру, і при аналізі важливо щоб кількість каналів у фільтрі збігалась з кількістю каналів у зображенні.

Загалом, архітектура CNN працює наступним чином: на початку кожної мережі знаходиться згортковий шар, що застосовує фільтри до зображення і перетворює дані зображення в числовий масив [17, 18]. Далі на ранніх рівнях проводиться обробка зображення для розпізнавання простих ліній, а пізніше

вони об’єднуються в складні фігури. Цей процес буде тривати поки модель не почне розпізнавати складі об’єкти, такі як наприклад одяг, тварини, автомобілі.

1.6 Постановка задачі

Кваліфікаційна робота присвячена розробці нейронної мережі для розпізнавання одягу і виявленню бренду за логотипом.

Об’єктом роботи є різні методи для класифікації зображень і логотипів на зображенні за допомогою створення спеціальної моделі.

Метою роботи є створення вебзастосунку, який покращує якість управління онлайн-магазинами за допомогою створення задач і отримання підказок з використанням штучного інтелекту, а також створення нейронної мережі, яка надає можливість аналізувати зображення і виявляти тип одягу, його бренд і генерувати опис на основі цих даних.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

– застосування методів обробки зображень, включаючи створення згорткової нейронної мережі (CNN), натренувати її на великих обсягах даних, які включають типи одягу і логотипи різних брендів;

– підключити наявну модель Gemini для отримання підказок в задачах і ChatGPT для створення описів зображень на основі аналізу;

– створити відповідне програмне забезпечення;

– провести тестування програми в реальних умовах і оцінити якість.

2 АНАЛІЗ РОБОТИ НЕЙРОННИХ МЕРЕЖ

2.1 Використання згорткових нейронних мереж в аналізі зображень

В задачах аналізу зображень згорткові мережі мають важливе значення. Їхнє завдання полягає у розпізнаванні об’єкта, в даному випадку одягу, його обробки та класифікації, а також в розпізнаванні логотипу на одязі, що робить цю задачу комплексною. Також важливим етапом є розпізнавання кольору одягу за допомогою принципу кластеризації, коли всі пікселі групуються в кластери RGB-простору [19, 20], де центрами класів виступають домінанті кольори, а розмір кластера відображає поширеність кольору в зображенні. Цей процес можна подати як:

$$C = KMeans(x_{pixels}, k = 3). \quad (2.1)$$

Ця формула відображає перетворення зображення x в множину пікселів, кожен з яких представлений як точка в RGB-просторі. Алгоритм KMeans групує ці точки в $k = 3$ кластери, знаходячи 3 центроїди, що мінімізують суму квадратів відстаней від точок до найближчих центроїдів.

Для розпізнавання об’єктів нейронну модель спершу тренують на відповідних наборах даних, що складаються із зображень з різними видами одягу, або брендів з різних ракурсів і позицій. Навчання передбачає подання вхідних даних і налаштування вагових коефіцієнтів для того, щоб мінімізувати різницю між результатом роботи мережі і фактичними мітками.

Під час тренування, зображення проходить всі шари, але в кінці додається два етапи, а саме: зворотне поширення помилки, і повторення епох. При зворотному поширенні обчислюються градієнти і поширюються назад через мережу, щоб перетворити ваги фільтрів за допомогою оптимізатора, наприклад Adam або SGD. Цей процес навчання повторюється багато разів, щоб модель навчилась розпізнавати закономірності. Під час кожної епохи, при правильному навчанні зростає точність, і зменшуються втрати або помилки. Таким чином,

навчання моделі базується на принципі градієнтного спуску і зворотному поширенню помилки, тобто модель поступово мінімізує помилку, кожного разу коригуючи свої параметри для точнішого передбачення. Для визначення локального градієнту використовується наступна формула:

$$\delta = e \cdot f(v_{out}), \tag{2.2}$$

де f – похідна функції активації. Вихідною функцією активації може бути або гіперболічний тангенс, або сигмоїда.

Після навчання нейронна модель може обробляти нові зображення, які містять одяг, а також бренд на ньому. Після чого, на основі цих даних, генерується опис до товару, за допомогою вже наявної моделі. При обробці зображення модель здатна розпізнати тип одягу, колір, матеріал, логотип, якщо він присутній, і класифікувати його.

При досліженні такої задачі варто врахувати основні проблеми, які з’являються при використанні нейронних моделей, це: безпека даних, точність, швидкість роботи, використання ресурсів. Варто звертати увагу на всі ці аспекти, і знайти баланс між ними, щоб модель могла швидко оборбляти зображення, при цьому не втрачати точність і не затрачати занадто багато ресурсів. Також, оскільки це реалізовано як вебзастосунок, важливо щоб користувач мав змогу безпечно завантажити зображення для аналізу, без ризику втрати конфіденційності.

Крім того, для створення підказок в задачах і створення опису товару, доцільно використати наявні текстові нейронні моделі, такі як ChatGPT і Gemini, при цьому врахувавши їхню ціну, а також швидкість і точність роботи.

Згорткові нейронні моделі стали дуже важливими в галузі комп’ютерного зору, допомагаючи проводити аналіз зображень. Їх застосування для класифікації одягу і брендів показує їх ефективність. З кожним роком нейронні моделі відіграють все більшу роль в різних галузях, а дослідження відкривають нові можливості в їх використанні.

2.2 Формальний опис процесів виявлення одягу на зображенні і роботи текстових нейронних моделей

2.2.1 Процес виявлення одягу на зображенні

Правильне виявлення одягу на зображенні і дуже важливим кроком для подальшої класифікації, цей етап передбачає виокремлення частини з одягом від фону. Для вирішення цієї задачі використані різні алгоритми і архітектури, що допомагають у вирішенні.

Процес виявлення одягу на зображенні поєднує в собі два основних завдання: локалізація об’єкта і його класифікація. У межах цієї задачі, нейронна модель матиме здатність розпізнати наявність одягу, його тип, колір, стиль і бренд.

На першому етапі основною задачею є знайти ділянки зображення, на яких найімовірніше знаходиться елемент одягу. Для цього було використано архітектуру, яка поєднує в собі створену CNN і модель ResNet-18 [21, 22], яка створює обмежувальні рамки для об’єкта. Вона допомагає моделі розпізнати важливі деталі зображення, такі як частини тіла, окремі деталі одягу, силуети. Архітектура ResNet трансформує вхідне зображення у вигляд компактного числового подання, яке можна описати формулою:

$$f = ResNet(I). \tag{2.3}$$

У цій формулі кожен блок архітектури робить наступне перетворення:

$$y = F(x_i, W_i) + x, \tag{2.4}$$

де x – вхід до блоку;

$F(x_i, W_i)$ – нелінійне перетворення (згортка, нормалізація, активація);

W_i – ваги згорткових шарів, y – вихід з блоку.

Завдяки додаванню x, модель вчиться лише між різницею вхідного і бажаного вихідного представлення, що значно прискорює і покращує навчання.

Другим етапом є – класифікація знайдених фрагментів. Під час якого для кожного знайденого об’єкта CNN аналізує отримані ознаки в роботі ResNet, які відображають форму, текстуру, колір. Це дозволяє моделі віднести об’єкт до певної категорії. Але для того щоб вона це робила правильно, попередньо її треба натренувати на великій кількості даних, в даному випадку гарним рішенням буде датасет від FashionMnist, який має понад 40000 тренувальних зображень. Класифікацію типу одягу можна описати наступною формулою:

$$p_{clothing} = softmax(W_{clothing} \cdot y_{fashion}(x) + b_{clothing}). \quad (2.5)$$

Оригінальне зображення x попередньо обробляється моделлю ResNet ($y_{fashion}$), після чого отриманий вектор ознак множиться на матрицю ваг $W_{clothing}$ і додається зміщення $b_{clothing}$, яке додає необроблені значення(логіти) для кожного класу. Функція softmax відповідає за перетворення логітів в розподіл ймовірностей $p_{clothing}$, де кожне значення відповідає ймовірності того, що зображення належить до певного класу. Після чого модель обирає клас з найбільшою точністю і обирає її для зображення (рис. 2.1). Функцію softmax можна представити у вигляді формули:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad (2.6)$$

де z_i – логіт для класу I;

K – загальна кількість класів, у випадку з ResNet – 10 класів одягу.

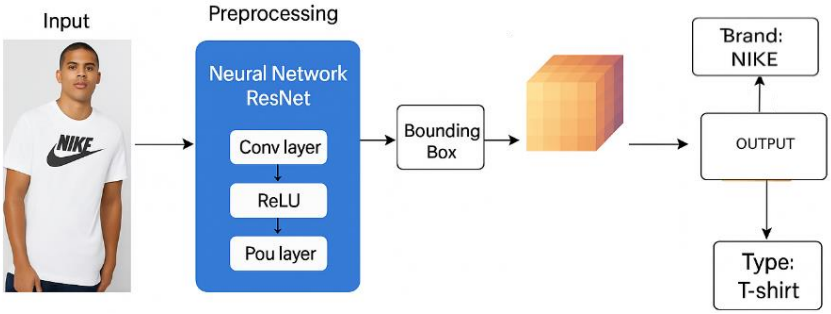

Рисунок 2.1 – Процес виявлення одягу і бренду

За допомогою глибинного навчання і згорткових нейронних моделей задачі розпізнавання одягу стали більш зрозумілими і доступними у вирішенні. З використанням архітектур типу ResNet, задачі класифікації і локалізації стають набагато простішими, оскільки вони сильно допомагають основній згортковій мережі вивчати складні закономірності і зв’язки даних в зображенні [23, 24].

Також важливим в роботі архітектури ResNet, є його архітектура кодера-декодера, яка допомагає вилучити ознаки зображення за допомогою кодера, і також відновити зображення за допомогою декодера. Це допомагає нейронній моделі не загубити частину даних при аналізі, оскільки вона їх стискає. Для цього з’єднуються відповідні рівні кодера і декодера, що дає змогу не втрачати інформацію.

Функції кодера полягають у застосуванні серії згорток, щоб отримати абстрактні ознаки, тим самим зменшивши розмірніть і виділити важливі закономірності або візерунки. Декодер в свою чергу проводить процес штучного підвищення дискретизації для того, щоб відновити розмірність і перевести всі ознаки в потрібний вигляд.

Таким чином, завдяки кодеру і декодеру створюється ієрархічна обробка ознак, від країв і текстур до контурів і об’єктів в абстрактному вигляді, після чого все повертається до піксельного вигляду для подальшого використання. Це потрібно для того, щоб сегментувати зображення, або повернути втрачений сигнал [25, 26].

Цей принцип значно знижує симетрію параметричного простору, тим самим зменшуючи кількість локальних мінімумів, зменшуючи ступінь перестановки симетрії шару з n! до 1! [27]. Якщо використати це по всій глибині мережі, це дозволить значно знизити рівень симетрії простору, що дозволить зробити її ландшафт більш підходящим для оптимізації (рис. 2.2). Також використання цього принципу допомагає вирішити проблему з деградацією точності, завдяки пропуску шарів точність завжди покращується, і не виникає проблем в роботі створеної моделі.

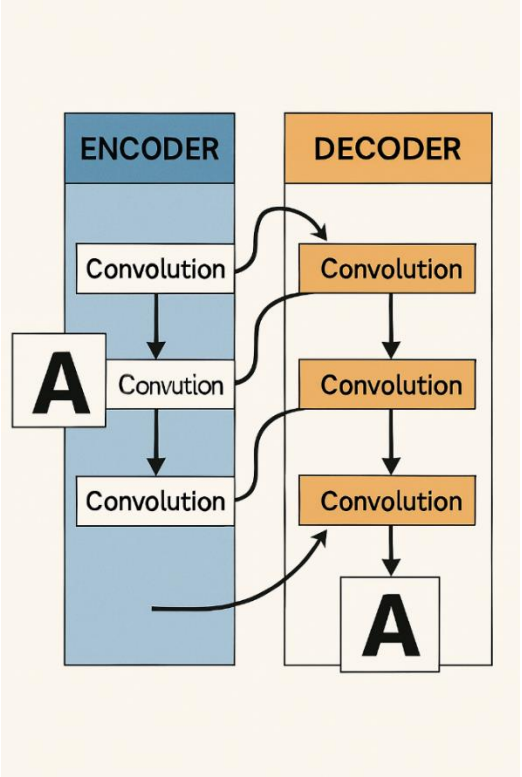

Рисунок 2.2 – Принцип skip connections зв’язку між кодером і декодером

Принцип пропусків є важливим і гарним рішенням, яке вперше було широко використане саме в архітектурах моделей ResNet, оскільки він допомагає вирішити проблему деградації точності. Суть проблеми полягає в тому, що при збільшенні глибини точність спочатку зростає, але потім стає сталою, або навіть знижується. Принцип skip connections допомагає вирішити саме цю проблему шляхом зниження симетрії параметричного простору, що оптимізує простір нейронної мережі, і усуває проблему деградації точності [28, 29].

Отже використання принципу пропуску зв’язків допоможе покращити точність моделі і підвищити якість навчання, усуваючи проблему деградації точності, це є важливим фактором при створенні великої моделі для класифікації.

При використанні принципу пропуску зв’язків можна також оптимізувати роботу за допомогою різних оптимізаторів, які допомагають пришвидшити як навчання, так і роботу моделі. В поєднанні з цим принципом, система отримує значну оптимізацію

2.2.2 Принцип роботи текстових нейронних моделей для створення підказок і опису товару

Текстові нейронні моделі, такі як ChatGPT і Gemini є складними системами машинного навчання, в основі яких лежать архітектури трансформерів, розроблених для обробки і генерації природної мови. Для навчання ці моделі використовують величезні масиви даних текстової інформації, включаючи книжки, статті, вебсторінки. Це дає їм змогу вивчити правила мови, закономірності і також вивчити інформацію з різних галузей знань.

Основою цих моделей є механізм передбачення слова, або фрагмента тексту на основі попереднього контексту. Під час навчання ці нейронні моделі оптимізовують мільярди параметрів, які визначають ймовірності появи слів у послідовності. Під час роботи, модель отримує запит від користувача, наприклад для створення підказок або описів товару, після чого вона конвертує цей запит в векторне представлення. Після аналізу цього запиту, модель LLM (Large Language Model) генерує відповідь, шляхом обрання найбільш доречних слів з урахуванням попереднього контексту, мети та задачі [30, 31].

Архітектура великих моделей, таких як GPT або Gemini, в основному базується на моделі Transformer. Вона складається з кодера і декодера, але у випадку великих моделей типу GPT-3 лишається лише декодер. Дуже важливим компонентом цієї архітектури є механізм самоуважності, тобто модель сама має змогу оцінювати важливість різних слів у вхідних даних під час прогнозування.

Параметри моделі, як і для зображень, тренуються за принципом стохастичного градієнтним спуском, який змінює параметри в залежності від епохи, щоб мінімізувати значення між прогнозом моделі і фактичним словом в фактичних тренувальних даних.

В основі багатьох текстових моделей лежить функція:

$$P(x_t | x_1, x_2, \dots, x_{t-1} = softmax(f(x_1, x_2, \dots, x_{t-1})). \quad (2.7)$$

Це ймовірність того, що наступний токен x_t буде певним словом, якщо модель вже бачила токени від x_1 до x_{t-1}. Ця функція реалізована трансформером, який приймає послідновність токенів, і генерує вектор логітів. Після цього, за допомогою softmax отримується ймовірність кожного токена з словника. На основі цієї функції моделі можуть передбачувати, доповнювати і створювати текст.

Для обрання найкращої текстової моделі для використання в задачі, було проведено аналіз моделей з використанням вебсервісу artificialanalysis.ai, на ньому зберігається інформація про наявні LLM моделі, які порівнюються між собою за такими параметрами як швидкість, ціна токенів, context window (кількість тексту, який модель може вивчити за один раз), і artificial analysis intelligence index. Для порівняння було взято найвідоміші моделі від OpenAI, Google, deepseek і antrophic. Результати порівняння на 22 квітня 2025 року наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння різних LLM моделей

Модель	Context Window	AI index	Ціна USD/1M Tokens	Швидкість s
Claude 3.7 Sonnet	200 тис.	48	6,00	0,96
GPT-4.1	1 млн.	53	3,50	0,39
Gemini 1.5 Flash	1 млн.	46	0,00	0,28
DeepSeek R1	128 тис.	60	0,96	3,30
o4-mini	200 тис.	70	1,93	38,3
GPT-4o Realtime	128 тис.	-	0,00	-

Проведений аналіз показав, що найбільш підходящою моделлю для виконання задачі є Gemini 1.5 Flash, оскільки вона є оптимальною по ціні, швидкості і кількості інформації, яку може запам’ятати. Ця модель чудово підійде для генерації підказок та описів товару за наданим промптом.

Сучасні текстові нейронні моделі також можуть покращувати свої відповіді і самоперевірятись. Вони добре оцінюють структуру і стиль тексту, що

є важливими у створенні таких комерційних відповідей, як описи, де точність і відповідність стандартам є дуже важливими факторами.

Такі моделі є дуже корисними для підприємств з великими каталогами продукції, або в платформах, які мають потребу в регулярному оновленні і створенні великих обсягів текстового контенту. LLM моделі можуть якісно і швидко оптимізувати такі процеси, оскільки вони є відносно недорогими і значно заощаджують час та використані ресурси для своєї роботи.

2.2.3 Огляд існуючих рішень для аналізу зображень

На сьогоднішній день існує багато застосунків для аналізу зображень, які варто порівняти їх і виявити недоліки, які стануть корисними для подальшої розробки нового рішення. Для порівняння буде використано основні критерії, які важливі при роботі з подібними застосунками, а саме: швидкість, ціна, точність. Порівняння буде проводитись на одному і тому самому зображені з однаковим запитом, який буде звучати наступним чином: «Проаналізуй зображення і визначи тип одягу на ньому, а також бренд, якщо він наявний». Для порівняння було взято найвідоміші сервіси, які є нейронними мережами з дуже великими обсягами даних і виконують різноманітні функції, але огляд буде зосереджений саме на аналізі зображень і виявленні типів одягу, а також класифікації брендів за логотипами.

Одним з найпотужніших сервісів для аналізу зображень є Google Cloud Vision API. Він може бути інтегрований в будь-яку систему і надавати функції аналізу зображень. Його проблемою є відсутність аналізу типів одягу, лише знаходження відомих логотипів. Для того щоб використовувати в ньому аналіз типів одягу, потрібно тренувати модель в окремому застосунку. Ще однією проблемою є те, що сервіс є платним, тобто для використання його у великих масштабах, доведеться постійно підтримувати баланс на акаунті для оплати. Ціна починається від 2 до 23 доларів США, в залежності від потрібного функціоналу, а також деякі функції є платними з самого початку, на деяких є

ліміт використання на місяць, в обсязі 1000 запитів на місяць, що буде недостатнім для вебзастосунку, який буде широко використовуватись кожного дня. Суттєвим недоліком також є те, що це лише сервіс, який можна інтегрувати до існуючого застосунку, тобто немає можливості його використати окремо. Якщо розглядати серед наявних застосунків, можна виділити відомий ChatGPT. Він має достатньо високу точність аналізу зображень, час аналізу в середньому складає до 10 секунд, що є досить гарним результатом. З недоліків є те, що безкоштовно можна аналізувати до 15 зображень на день, або ж можна придбати платну версію за 20 доларів за місяць, що також є не досить зручно. Наступним застосунком можна виділити Claude, який також є текстовою моделлю, з можливістю аналізувати логотип і бренд одягу. Він є швидшим і точнішим за GPT, але має більш обмежену безкоштовну версію, яка дає можливість аналізувати менше зображень на день. Платна версія коштує 20 і 100 доларів, в залежності від функцій, що також є достатньо затратно. Наступним сервісом є Gemini, який аналізує зображення повільніше, ніж попередні сервіси. Хоч він і є безкоштовним, при появі нових інструментів, платформа надає їх в підписці за 900 гривень на місяць. Недоліками є час аналізу і точність, порівняно з раніше описаними сервісами.

На основі переглянутих сервісів, можна створити таблицю, в якій відображено основні характеристики наявних моделей (табл. 2.2). Це допоможе в майбутній розробці, оскільки можна побачити, на що спираються великі і відомі моделі, але при цьому усунути недоліки, які в них наявні, і зробити кращу модель.

Таблиця 2.2 – Порівняння сервісів для аналізу зображень

Сервіс	Ціна	Точність	Швидкість (с)	Тип
Cloud Vision	від 2 до 20$	Висока	<5	API
ChatGPT	20$	Висока	від 5 до 10	застосунок
Claude	20$ або 100$	Дуже висока	до 10	застосунок
Gemini	20$	Середня	>10	застосунок

При цьому, варто так само брати до уваги ключові моменти, пов’язані з точністю, швидкістю і ціною, як це збалансовано у таких великих моделях. Це допоможе в майбутній розробці тим, що великі нейронні моделі демострують принцип їх розробки і на що спирались великі команди розробників. Визначення найважливіших аспектів допомагає будувати застосунок з нейронною мережею, оскільки розробка одразу починається з урахуванням всіх ключових моментів.

Таким чином, більшість сервісів, які дають можливість аналізувати і класифікувати зображення є платними, мають обмежену кількість запитів, або з самого початку мають обмежену функцію лише для підписки. Всі з переглянутих застосунків мають досить непоганий результат, але дослідження було проведене з обмеженою кількістю зображень, оскільки всюди були обмеження по кількості. Окремим фактором варто розглянути швидкість, оскільки всі застосунки витрачали різний час на аналіз, і лише сервіс для інтеграції в інші застосунки виконав завдання швидше ніж 5 секунд, середнім результатом роботи програми було від 5 до 10 секунд, що є досить швидко, але важливо притримуватись балансу між точністю і швидкістю, і головне – створити застосунок, який матиме ті самі можливості, але повністю безкоштовно.

2.3 Проблеми аналізу і класифікації

Однією з головних проблем при класифікації типів одягу і аналізу зображень є варіативність зовнішнього вигляду одягу. Один і той самий тип одягу може відрізнятись за кольором, фасоном, стилем. Це може створити певні проблеми, оскільки моделі важко розпізнати схожі між собою типи, або навпаки, знайти спільні риси між різними варіантами одного й того самого типу.

Також важливою проблемою є якість зображень. У реальних умовах зображення мають різне освітлення, ракурс фото, різне положення в кадрі. Або ж зображення можуть бути розмитими на деяких ділянках з перекриттями одягу, що значно впливає на точність і швидкість аналізу. Те саме відбувається і з

логотипом бренду, він може бути нечіткий, перекритий, або деформований, що погіршить якість програми.

Складність також викликає відсутність великих датасетів, особливо для класифікації брендів. Більшість датасетів сильно обмежені по кількості тренувальних даних, або брендів. А великі містять зайві логотипи, що не відносяться до одягу. Тому найкращим рішенням буде взяти готовий датасет, очистити його від зайвого, і доповнити підготованними даними.

Ще одним важливим моментом є те, що багато брендів схожі між собою, і на зображеннях де логотип зазвичай є маленьким, складно розпізнати до якого саме бренду він відноситься. При цьому існує багато різних брендів, які мають різні варіації логотипу, і вона може бути дуже великою, через що потрібно тренувати модель на великій кількості різних варіацій.

Динамічність моди може теж стати викликом, оскільки нові колекції і стилі в брендах з’являються постійно, тож треба постійно слідкувати за цим, оновлювати набори даних і тренувати модель. Також існує проблема в тому, що в більшості брендів є моделі, які значно більше представлені ніж інші, тому для деяких колекцій і логотипів важко знайти багато об’єму інформації, що може створити упередженість до окремих брендів.

Варто звернути увагу також на проблему використання ресурсів. Оскільки модель для аналізу зображень містить дуже багато коду і різних алгоритмів, дуже важливо слідкувати за тим, щоб використовувалась оптимальна кількість ресурсів навіть на слабких девайсах.

Через всі ці проблеми, нейронна модель може працювати недостатньо швидко і точно. Через що потрібно приділити дуже велику увагу цим аспектам. Для вирішення проблем з точністю допоможе поєднання декількох алгоритмів і архітектур для класифікації. Але при цьому також варто тренувати модель на величезному обсязі даних, щоб вона мала змогу чітко класифікувати різні типи одягу, не зважаючи на якість вхідного зображення. Це ще більше може вплинути на швидкість роботи моделі, тож варто приділити багато часу оптимізації коду, додатково використовуючи різні оптимізатори.

Також з цих проблем може випливати важлива задача, створити правильне конвертування зображень під час аналізу, оскільки нейронна модель перетворює зображення під час аналізу, деякі дані можуть втратитись, що є ключовими під час класифікації. Забезпечення правильного конвертування, при якому зображення не спотворюється і не втрачає таких деталей як наприклад логотип, або краї зображення, допоможе набагато краще впоратись з поставленною задачею, тож реалізації правильних методів конвертації потрібно приділяти багато уваги.

Для вирішення всіх цих проблем застосовуються новітні алгоритми і методи аналізу зображень. Ці алгоритми можуть змінювати свої параметри в залежності від змін умов на зображеннях, при цьому зберігаючи свою продуктивність. Також можна використовувати сторонні програми, або бібліотеки, які допомагають у створенні нейронних моделей, особливо в питаннях оптимізації і покращенні швидкості обробки зображень. Також варто приділити увагу зручності і зрозумілості, оскільки нейронна модель буде використана у вебзатосунку, важливо щоб будь-якому користувачу було зрозуміло, як користуватись застосунком, зокрема скористатись нейронною моделлю. Для цього її потрібно правильно інтегрувати в застосунок.

У підсумок, хоч технології для аналізу зображень і класифікації типів одягу та брендів на них, дають дуже багато переваг у зручності, конфіденційності і заощадженні коштів [7, 9], вони мають свої певні проблеми та виклики. Для вирішення цих проблем, варто звертати увагу на всі можливі аспекти і використовувати дуже багато різних алгоритмів, технологій і новітніх сервісів. Важливо шукати баланс між точністю, швидкістю і використанням ресурсів, для того щоб забезпечити максимально продуктивну, якісну і зручну програму для різних видів діяльності, і щоб користувачі могли використовувати її з різних пристроїв. Зростання попиту на такі технології дають розвиток подібним моделям, отже якість з часом буде лише покращуватись і це стане незамінним інструментом у різних сферах діяльності. Майбутнє дослідження має полягати у покращенні моделей з боку оптимізації, а також для створення нових алгоритмів, які будуть покращувати роботу моделей не лише для класифікації зображень, а

й в цілому для згорткових нейронних моделей. Моделі також повинні забезпечувати безпеку при використанні і полегшувати роботу користувачів. Використання програм з використанням нейронних моделей значно оптимізує різні процеси в сферах діяльності людей, тим самим оптимізуючи рівень витрат і збільшуючи можливі обсяги виконаної роботи.

2.4 Обгрунтування вибору середовища розробки і проєктування застосунку

2.4.1 Обгрунтування вибору середовища розробки

Під час реалізації застосунку для управлінням задачами онлайн магазину з технологіями аналізу зображень і штучним інтелектом було обрано середовище Visual Studio Code (VS Code), який є одним з найпопулярніших середовищ для розробки, оскільки він зручний, гнучкий і має потужний спектр функціональних можливостей, включаючи різні плагіни, що допомагають комфортно розробляти великі застосунки з різними мовами програмування. Основними плюсами цього середовища розробки є:

– підтримка різних мов програмування: VS Code підтримує багато різних мов програмування, зокрема HTML, CSS, JavaScript і Python, які потрібні в розробці застосунку, що розглядається;

– кросплатформність: VS Code підтримує різні операційні системи, включючи Windows, macOs та Linux, що забезпечує гнучкість в розробці;

– швидкість: не дивлячись на те, що VS Code має дуже багато різних можливостей, плагінів та інших інструменів, він залишається легким і швидким;

– постійні оновлення: це середовище постійно оновлюється і завжди підтримується розробниками, що постійно додає все більше нових можливостей та інструментів для розробки.

Таким чином, середовище розробки VS Code є чудовим варіантом для обрання, оскільки воно є гнучким, зручним та швидким, і включає в себе багато функціоналу і можливостей, які полегшують розробку, оптимізують різні

процеси і збільшують користувацькі можливості. Це робить його найкращим варіантом для обрання при реалізації такого складного вебзастосунку для управління задачами з нейронними мережами. Постійне оновлення цього середовища також дає змогу дотримуватись найновіших стандартів програмування і підтримувати існуючі програми оновленими.

2.4.2 Проєктування застосунку

Основним компонентом для будування вебзастосунку є сторінка, яка служить окремим екраном для взаємодії з користувачем. Вона містить свої певні частини контенту і функції, з якими користувач може взаємодіяти. Також основою сторінки є URL – адресу, що є унікальною, і за допомогою неї користувачі можуть переходити на цю сторінку за допомогою браузеру.

Кожна сторінка має своє призначення, контент, форматування і функціонал, який доступний користувачу. Для покращення користувацього досвіду важливо адаптувати сторінки під різні типи пристроїв, такі як комп’ютери, ноутбуки, телефони і планшети, щоб вебзастосунок був зручним і однаково працював незалежно від типу екрану.

З урахуванням основних функцій застосунку, потрібно створити сторінки «Task Manager» і «Image Uploader», на яких користувачі матимуть змогу управляти задачами і завантажувати зображення для аналізу відповідно.

Сторінка «Task Manager» відповідатиме за частину, пов’язану з управлінням задач в онлайн магазині, також вона буде головною сторінкою застосунку. Для цього важливо розробити функціонал, який дозволить керувати задачами, створювати їх, генерувати підказки, слідкувати за дедлайнами і зручно розподіляти задачі по категоріях. Для цього потрібно створити поля, які відповідатимуть за створення задач, а саме назву задачі, її опис і час, до якого вона має бути виконана. Також важливим елементом в таких застосунках є пріоритетність виконання задачі, щоб користувачі могли зручно слідкувати і розподіляти виконання задач протягом часу. Для перегляду створених задач

потрібно створити так звани дошки, або борди, на яких будуть відображатись створені задачі. Користувач зможе змінювати їхню назву, перетягувати задачі між ними в залежності від потреби і також помічати як готові задачі або видаляти їх. Оскільки цей застосунок буде використовуватись не для однієї задачі, важливо створити архів задач, це буде окремий блок, в якому будуть відображатись всі останні задачі, які були виконані і вони просто зберігаються на сервері, за потреби їх можна буде видалити.

Оскільки ця сторінка є основною, потрібно також буде помістити основну інформацію про застосунок в блоці About, а також створити блок для зворотного зв’язку, де буде міститись основна інформація про те, як зв’язатись з власником цього вебзастосунку.

Сторінка «Image Uploader» буде допоміжною і відповідатиме за завантаження картинок користувачів для аналізу. Для цього важливо розмістити відповіді компоненти для потрібного функціоналу. Основним елементом на сторінці буде поле, через яке зображення будуть завантажуватись, це може бути зроблене як за допомогою перетягування зображення на це поле, так і за допомогою вибору файлу. Далі ці зображення мають відображатись на сторінці, щоб користувач міг їх аналізувати, це буде зроблено у вигляді карток під полем для завантаження, вони міститимуть інформацію про зображення, а саме: id цього зображення на сервері, вага зображення, дата завантаження на сервер і опис. При завантаженні воно не міститиме опису, але після того, як користувач натисне кнопку «Get Tips», почнеться виконання програми на мові програмування Python, і через деякий час користувач отримає на екрані опис до зображення, яке він обрав, після чого цей опис збережеться, і можна буде його перегенерувати для потреби.

Застосунок буде складатись з трьох частин, це клієнтська, серверна та застосунок на Python. Важливо щоб всі три частини були правильно пов’язані між собою і чітко взаємодіяли одна з одною. Важливо дотримуватись безпеки і конфіденційності користувачів, щоб ніхто, крім адміністратора застосунку не міг отримати доступу до бази даних. Також важливо розуміти коли і яка частина застоунку має спрацьовувати на запити користувача. Важливо щоб сервер точно

пов’язував всі сервісі в застоунку і в тому числі клієнтський інтерфейс, для того щоб забезпечити чітку і швидку взаємодію. Для цього варто розробити діаграму взаємодії між цими трьома частинами, щоб розуміти, як це все буде відбуватись в застосунку (рис. 2.3). Дана діаграма відображає основні взаємодії між користувачем, фронтенд, бекенд частинами, базою даних і застосунком на Python. Вона описує послідовність головних ітерацій між елементами застосунку і клієнтом.

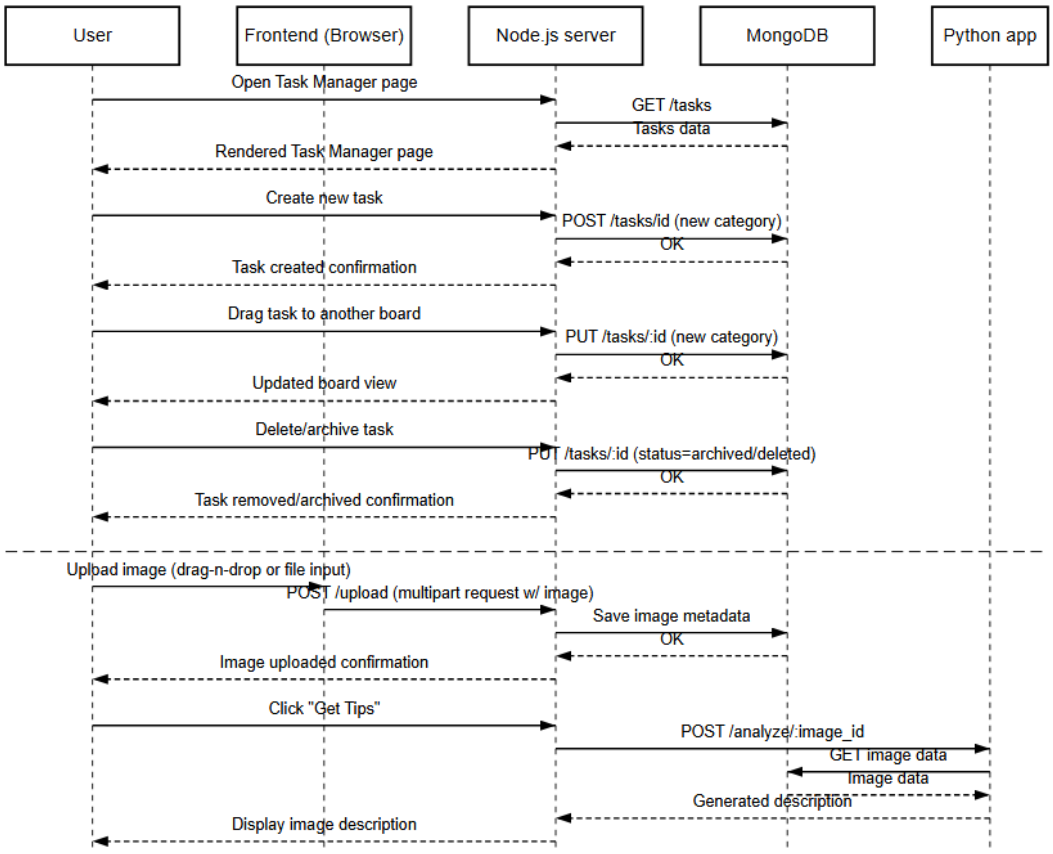

Рисунок 2.3 – Діаграма взаємодії застосунку

Основна взаємодія проходить через сервер, який буде підтримуватись через Node.js. Коли користувач взаємодіє з інтерфейсом на різних етапах, від відкриття сторінки, до створення задач, або натискання на кнопки, інформація передаватиметься на серверну частину застосунку. Далі сервер буде обробляти дані, в залежності від дії, яка була виконана і повертати результат. Забезпечення правильної взаємодії дає змогу дотримуватись безпеки і швидкості виконання різних функцій у застосунку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1 Основні компоненти програми

При розробці вебзастосунку для управлінням задачами в онлайн-магазині з використанням нейронної моделі для аналізу зображень і штучного інтелекту для підказок в задачах і описами товарів потрібно детально описати архітектуру, структуру коду і основні функції розробленої програми. Важливо також обрати найбільш підходящі середовища і мови програмування щоб реалізувати цей застосунок для комфортного використання.

Застосунок складається з декількох модулів, що включають в себе клієнтську частину, серверну і застосунок на Python. Для реалізації цих частин було обрано наступні мови програмування:

– для клієнтської частини було використано HTML, CSS та мову програмування JavaScript;

– для серверної частини було обрано фреймворк Node.js, який є середовищем JavaScript, що використовується для створення серверних застосунків. Додатково використовується фреймворк Express, що дозволяє створювати API;

– для реалізації нейронної моделі було використано мову програмування Python;

– додатково для зберігання інформації в базі даних використовується нереляційна база даних MongoDB, яка забезпечує гнучке управління великими обсягами інформації.

На початку релізації важливо слідкувати за підключенням до бази даних, щоб клієнтська частина могла приєднуватись до серверу з правильною логікою, щоб в подальшому обробляти і зберігати дані. Підключення до бази даних подано в лістингу 3.1.

Лістинг 3.1 Реалізація підключення до бази даних MongoDB за допомогою бібліотеки Mongoose:

```
async function connectDB() {
  try {
    const client = new MongoClient(mongoURL);
    await client.connect();
    db = client.db(dbName);
    console.log("Connected to MongoDB");
  } catch (error) {
    console.error("MongoDB connection error:", error);
    process.exit(1);
  }
}
connectDB();
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

Підключення виконано за допомогою особливості в Node.js, яка називається Mongo Client, це драйвер, який містить екземпляр MongoClient, який приймає URL, за яким надається доступ до MongoDB, і callback функцію, якій передається два параметри: помилка підключення (null, якщо підключення успішне), екземпляр об’єкта підключення до MongoDB, через API якого надається доступ до існуючих БД.

Далі важливою є реалізація клієнтського застосунку, щоб вона могла працювати з серверною частиною. Для прикладу наведено лістинг 3.2 в якому наведено приклад функції з фронтенд частини для завантаження зображень на сервер. Функція зв’язується з сервером, для того щоб передати статус і інформацію про зображення, після чого надсилає відповідне повідомлення.

Лістинг 3.2 Завантаження зображень з клієнтської сторони до серверу:

```
function uploadImageToServer(image) {
    const formData = new FormData();
    formData.append('image', image.file);
    fetch('http://localhost:5000/upload_image', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            console.log("Image uploaded successfully:", data);
        } else {
            console.error("Error uploading image:", data.error);
        }
    })
    .catch(error => console.error("Server error:", error));
}
```

Цей Node.js файл містить фреймворк Express, що надає змогу використовувати об'єкт під назвою app, за допомогою якого можна робити наступні операції:

– роутинг HTTP-запитів, наприклад app.get, app.post, app.delete;

– рендеринг відображень в HTML, app.render;

– реєстрація шаблонізаторів, app.engine.

В лістингу 3.3 наведено приклади використання роутингів HTTP-запитів, що дає змогу передавати дані між клієнтом і сервером. Всі запити передаються динамічно, що дає змогу постійно оновлювати інформацію. Це дає змогу сторінці постійно відображати актуальну інформацію, що надходить з серверу, або з’являється після взаємодії клієнта з інтерфейсом.

Лістинг 3.3 Приклади app.get, app.post і app.delete:

```
app.get("/tasks", async (req, res) => {
  try {
    const tasks = await db.collection("tasks").find({}).toArray();
    res.json(tasks);
  } catch (error) {
    console.error("Error fetching tasks:", error);
    res.status(500).json({ message: "Error fetching tasks" });
  }
});
app.post("/tasks", async (req, res) => {
  try {
    const { title, description, priority, deadline } = req.body;
    if (!title) {
      return res.status(400).json({ message: "Task title is required" });
    }
    const newTask = {
      title,
      description: description || "",
      priority,
      deadline: deadline ? new Date(deadline) : null,
      status: "todo",
      createdAt: new Date()
    };
    const result = await db.collection("tasks").insertOne(newTask);
    newTask._id = result.insertedId;
    res.status(201).json(newTask);
  } catch (error) {
    console.error("Error creating task:", error);
    res.status(500).json({ message: "Error creating task" });
```

```
    }
});
app.delete("/tasks/:id", async (req, res) => {
    try {
        const { id } = req.params;
        if (!ObjectId.isValid(id)) {
            return res.status(400).json({ message: "Invalid task ID" });
        }

        const result = await db.collection("tasks").deleteOne({
            _id: new ObjectId(id)
        });
        if (result.deletedCount === 0) {
            return res.status(404).json({ message: "Task not found" });
        }
        res.json({ message: "Task deleted successfully" });
    } catch (error) {
        console.error("Error deleting task:", error);
        res.status(500).json({ message: "Error deleting task" });
    }
});
```

За допомогою Express застосунок може швидко і досить просто створювати взаємодію між клієнтською і серверною частиною. Для того щоб фронтенд частина працювала з сервером, потрібно налагодити передачу даних, для цього буде використано асинхронні функції, які будуть використовувати серверні методи і витягувати дані з серверу за допомогою await fetch(http://localhost:5000/назва-колекції). Основні методи клієнтського файлу для управління задачами містять наступні методи: додавання, редагування, видалення і очищення задач, рендеринг сторінки для оновлення статусу при додаванні задачі, щоб вона залишалась при оновленні сторінки, звернення до

серверу щоб отримувати задачі, які вже є в базі даних, отримання підказок до задач, додавання і видалення бордів для задач, функція drag and drop для того щоб задачу можна було зручно перетягувати між бордами.

Функція getTaskTips() є першим використанням технологій штучного інтелекту. Вона звертається до серверу і надає запит на промпт текстової нейронної моделі, щоб отримати підказки до створених задач. В лістингу 3.4 наведено приклад цієї функції.

Лістинг 3.4 Функція для отримання і відображення підказок в задачах:

```
async function getTaskTips() {
    const tipsContainer = document.getElementById('tipsContainer');
    const tipsContent = document.getElementById('tipsContent');
    if (!currentTask) return;
    try {
      tipsContainer.style.display = 'block';
      tipsContent.innerHTML = '<div class="loading">Getting tips...</div>';
      const response = await fetch('http://localhost:5000/tasks/tips', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          title: currentTask.title,
          description: currentTask.description
        }),
      });
```

Після чого, викликається робота серверної частини, а саме HTTP-запит, який містить в собі підключення моделі Gemini-1.5-flash, яка надає підказку. Це робиться за допомогою створення змінної const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });.

Після підключення моделі створюється запит, який каже моделі, що саме вона потрібна відповісти. В застосунку запит заданий наступним чином «Give me 3-5 practical tips for completing this task:». Після чого модель отримує даний промпт, обробляє його. Сервер повертає цю відповідь до клієнта, і інформація відображається на екрані. Щоб все це працювало, потрібно отримати спеціальний токен, який надають Gemini до своїх моделей, в залежності від моделі, токени можуть коштувати по різному, але модель 1.5 flash є безкоштовною, тому отримавши унікальне значення, воно було підключене і збережене, щоб не втратити. Ці токени варто захищати, оскільки не можна допустити щоб його вкрали. Простіше всього створити .env файл, в якому він буде зберігатись і сервер буде звертатись до цього файлу щоб розшифрувати і отримати токен. Таким чином, на рисунку 3.1 відображено всі файли застосунку.

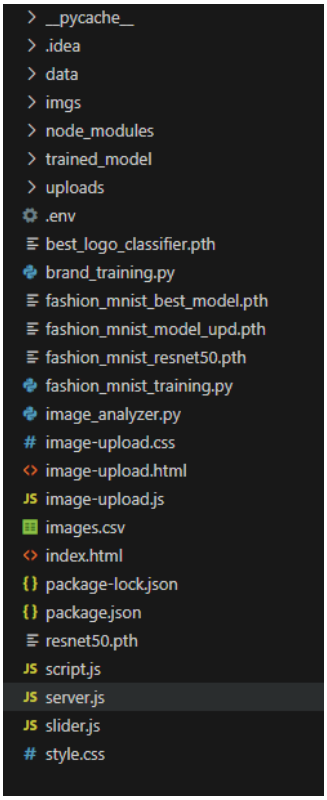

Рисунок 3.1 – Файли застосунку

Застосунок складається з папок, які зберігають файли для підтримання роботи мов програмування Python, Node.js, також папки де зберігаються зображення для сторінок і зображення серверу, додатково папки для зберігання

натренованих моделей і файли застосунку для клієнтської частини, серверу, а також основної нейронної моделі, і допоміжних для тренувння.

При розробці інтерфейсу вебзастосунку, який в подальшому буде використовуватись користувачами, важливо зберігати його зручність, зрозумілість і якість. Інтерфейс створюється з метою спрощення роботи з задачами, а також аналізом картинок, щоб кожен міг використати застосунок.

Ключовим файлом для головної сторінки є script.js, в якому реалізовані методи для зручного користування середовищем для управління задачами. Також важливо дотримуватись правильності зв’язку з серверною частиною, оскільки дані про задачу мають правильно створюватись, оновлюватись і видалятись. Для цього було використано наступні методи:

– метод addTask() реалізує процес створення нової задачі і додавання її до дошки з задачами. Користувач вводить назву і опис задачі, обирає дату, коли вона має бути виконана і пріоритет, після чого натискає кнопку «Add Task», і задача автоматично створюється в першій дошці;

– метод completeTask() відповідає за переміщення задачі до архіву задач шляхом зміни статусу задачі на «Completed». Коли користувач підтверджує що задача була виконана, цей метод надає їй відповідний статус і вона автоматично переноситься до архіву задач, а також змінює відповідні поля в базі даних, які пов’язані з статусом задачі і час створення змінюється на час виконання;

– метод deleteTask() дозволяє видаляти створені задачі. Якщо користувач створив задачу з неправильними даними, або під час роботи, задача втратила актуальність, вона може бути видалена. Користувач натискає на відповідну кнопку, після чого з’являється спливаюче вікно для підтвердження. Якщо вона все ж має бути видалена, то метод її видаляє з сторінки і дані в базі даних також зникають;

– метод deleteCompletedTask() відповідає за видалення задач, які знаходяться в архіві. Якщо задача з архіву вже не потрібна, користувач може її видалити, натиснувши кнопку «Delete task», після чого задача з архіву видаляється на сторінці і в базі даних;

– функція clearCompletedTasks() дозволяє видалити всі задачі з архіву. Якщо всі задачі в архіві потрібно видалити, можна натиснути кнопку «Delete all», після чого весь список задач видаляється з даними в базі;

– метод fetchTasks() потрібен для зв’язку даних про задачі між сервером та клієнтською частиною. Він оновлює дані в інтерфейсі, відповідно до бази даних. Тобто якщо в БД містяться дані про задачі, вони постійно будуть відображені на сторінці. Також при створенні, редагуванні і видаленні задач, цей метод оновлює інтерфейс і зберігає дані про поточні статуси задач, динамічно відображаючи зміни;

– метод fetchCompletedTasks() є допоміжним до методу fetchTasks(). Він так само пов’язує дані з задач і оновлює статуси, але у задач, що знаходяться в архіві;

– метод getTaskTips() є останнім і він виконує важливу роль в застосунку. Він відповідає за роботу підключенної нейронної мережі, а точніше за виклик її виконання. При натисненні на кнопку «Get Tips» функція відображає на інтерфейсі блок з опрацюванням роботи з надписом «Getting tips…», під час якого виконується запит до нейронної мережі і її робота. По завершенню роботи моделі, метод отримує створені підказки як результат, і відображає їх в блоці. Для роботи цього методу важлива наявність назви і опису задачі, оскільки вони передаються до серверної частини, де знаходиться код нейронної мережі, на основі цих даних виконується аналіз.

Таким чином виконується реалізація головної сторінки «Task Manager», де користувачі мають змогу зручно управляти задачами для онлайн-магазину. Інтерфейс є зрозумілим і не містить зайвих елементів, що можуть вплинути на якість роботи застосунку, або спричинити конфлікти в програмі. Додатково сторінка містить блоки «For Whom» і «Our Goals», які міститимуть інформацію про розробників застосунку і його цілі. Для візуалізації даних в другому блоці використано карусель, яку було створено в файлі «slider.js» (рис. 3.2). Він містить стандартний код, який використовується зазвичай для таких каруселей. Методи, які відповідають за анімацію перемикання слайдів, яку можна викликати шляхом

натискання на кнопки в вигляді стрілок збоку, або крапок знизу. Також вони автоматично переключаються кожні 5 секунд.

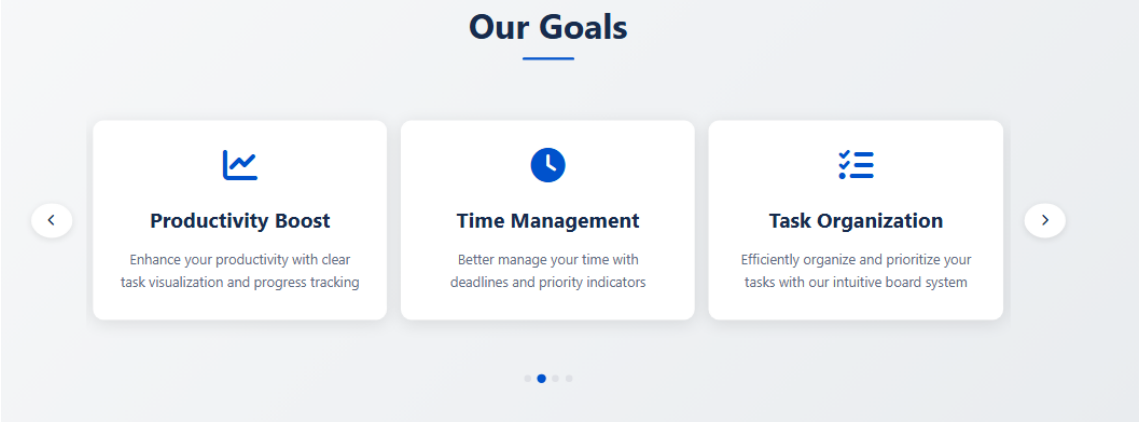

Рисунок 3.2 – Карусель для представлення цілей застосунку

Таким чином, основна сторінка застосунку має 6 блоків: блок для створення задач, блок для дошок з задачами, архів задач, блок для інформації про розробників і блок для цілей застосунку. Ця сторінка є практичною і містить основну інформацію, яка може знадобитись користувачам, які вперше користуються застосунком. Це допомагає покращити користувацький досвід і процес ознайомлення з вебзастосунком.

3.2 Проєктування та розробка бази даних

База даних буде реалізована на документо-орієнтованій нереляційній базі даних MongoDB, вона зберігає дані в форматі JSON, що є зручним при розробці вебзастосунку. Вона дає змогу зберігати складні дані, а також підходить для застосунків з високою доступністю. MongoDB є чудовим для розробки застосунків де потрібне читання великих обсягів неструктурованих даних.

Для роботи з MongoDB буде створено базу даних під назвою «task-manager», в якій буде міститись інформація про задачі, які створюються і зберігаються на сайті, а також зображення, для яких буде виконуватись аналіз і генеруватись опис.

Оскільки застосунок спеціалізується на збереженні задач і зображень, в MongoDB буде створено дві колекції: tasks і images. Для цих колекцій важливо визначити поля, які потрібні для зберігання інформації в базі даних. Також додатково буде створена колекція completed-tasks, для зберігання інформації про задачі, які були виконані і знаходяться вже в архіві. Схеми колекцій наведені в таблицях 3.1 – 3.3.

Таблиця 3.1 – Колекція images

Зміст	**Назва**	**Тип даних**	**Додаткова інформація**
Ідентифікатор зображення	_id	ObjectId	унікальне значення
Ім’я файлу	filename	String	
Шлях до файлу	path	String	
Дата завантаження	uploadDate	Date	
Розмір файлу	size	Int32	

Таблиця 3.2 – Колекція tasks

Зміст	**Назва**	**Тип даних**	**Додаткова інформація**
Ідентифікатор задачі	_id	ObjectId	унікальне значення
Назва задачі	title	String	
Опис задачі	description	String	
Приорітет	priority	String	low, high або medium
Термін виконання	deadline	Date	дата і час
Статус задачі	status	String	
Дата створення	createdAt	Date	

В наступній таблиці (табл. 3.3) буде представлено колекцію «completed-tasks», вона є схожою на основну колекцію tasks, але відрізняється полями «originalId» і «createdAt» замінюється на «completedAt», для того щоб зберігати в ній задачі які вже були виконані. Цей підхід, при якому створюється окрема

колекція для зберігання даних, які вже не будуть оновлюватись і мають просто зберігатись для подальшої звітності допомагає не плутатись в одній колекції і так не виникає ніяких конфліктів у даних.

Також це допомагає зручно переглядати різні задачі в БД, оскільки можна окремо дивитись задачі, які ще не виконані, і окремо переглядати архів.

Таблиця 3.3 – Колекція completed-tasks

Зміст	**Назва**	**Тип даних**	**Додаткова інформація**
Ідентифікатор задачі в архіві	_id	ObjectId	унікальне значення для кожного файлу
Назва задачі	title	String	
Опис задачі	description	String	
Зміст	Назва	Тип даних	Додаткова інформація
Приорітет	priority	String	може бути low, medium і high
Ідентифікатор оригінальної задачі	originalId	String	зовнішній ключ до колекції tasks, який посилається на головний ключ в задачі
Дата створення	createdAt	Date	

Всі створені колекції мають свою роль і зберігають різну інформацію, яка в подальшому буде використана при розробці застосунку. Важливо зберігати конфіденційність і цілісність даних, щоб не виникало конфліктів між застосунком і базою даних, а також слідкувати за постійним оновленням БД.

Таким чином, було розглянуто основні компоненти, які будуть розроблятись при створенні вебзастосунку для управління задачами онлайн-магазину з використанням нейронних мереж і технологій штучного інтелекту. Детальний опис основних функцій і взаємодій значно спростить розробку, оскільки все буде виконуватись поетапно і згідно того, як все має працювати.

Розглянуті діаграми і таблиці допоможуть в розробці, оскільки вони добре демонструють логіку програми, що дає загальне розуміння системи взаємодій і допоможе звертати увагу на ключові моменти. На рисунку 3.3 представлено вигляд зберігання даних у колекції tasks.

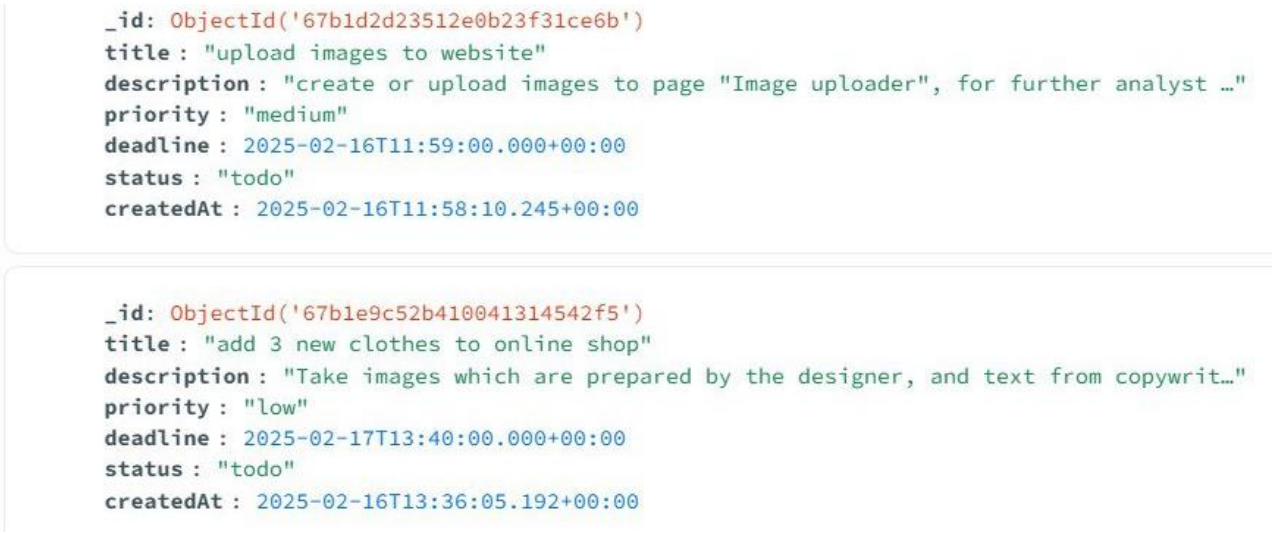

Рисунок 3.3 – Представлення даних в колекції tasks

За схожим принципом зберігаються дані і в інших колекціях, вони потрапляють сюди одразу після створення нових даних. Для того щоб передавати цю інформцію, використовується формат JSON, він надає змогу динамічно зберігати і передавати інформацію між застосунками, або між базою даних і застосунком. Всі ці колекції знаходяться в одній базі, до якої отримується доступ лише адміністратору баз даних, лише через нього користувачі можуть отримати повну інформацію з колекцій, яка їм потрібна.

3.3 Створення нейронної моделі для аналізу зображень

Основною перевагою застосунку є нейронна модель, яка здатна проводити аналіз зображень і виявляти типи одягу та бренди на них. Для початку створення було визначено архітектуру, на основі якої буде будуватись модель, це

ResNet-18, вона є однією з найкращих для створення моделей, які базуються на аналізі зображень. Також було визначено набір даних для навчання. Для аналізу типу одягу було використано набір даних від Fashion Mnist, який містить понад 40000 тренувальних зображень, вони містять зображення різних типів одягу, футболок, штанів, взуття і такі інші. Для аналізу бренду було обрано набір даних з відкритого сервісу Kaggle, який містив понад 100000 різних тестових зображень логотипів за категоріями. Цей набір даних було очищено, щоб залишити лише зображення логотипів відомих брендів одягу. Далі було збільшено наповнення деяких брендів серед яких є такі відомі як: Adidas, Puma, NewBalance, Prada, Lacoste та інші. Після чого було розпочато тренування.

Для тренування було створено дві моделі, brand_training і fashion_mnist_training. Ці файли будуть відповідати за тренування на зібраних тренувальних даних і зберігання навчених ваг, які потім застосуються в основній моделі для класифікації.

В основі навчання моделі за типами одягу лежить виявлення різних типів одягу на кожному тестовому зображенні. Для цього модель спочатку трансформує їх подання в три канали, для розпізнавання різнокольорових зображень, далі трансформує і нормалізує зображення за допомогою методів Resize() і Normalize(). Додатково використано метод CenterCrop(), який обрізає зображення так, щоб елемент одягу був по центру. Після перетворень модель починає тренуватись на кожному зображенні, визначаючи втрату тренування і значень на зображеннях. Для тренування було визначено термін в 10 епох, тобто під час кожної епохи модель навчається, змінює свої параметри щоб покращуватись і в кінці кожної епохи зберігає найкращі ваги. Додатково використано оптимізатор Adam, щоб пришвидшити навчання без втрати важливої інформації. В результаті модель зберігає файл з натренованими вагами, які містять інформацію про 10 класів одягу, які вона може розпізнати. Для навчання використовується набір даних Fashion Mnist, який є частиною бібліотеки tensorflow. Підключення цього набору даних наведено на рисунку 3.4.

```
1   import tensorflow as tf
2   import numpy as np
3   from tensorflow import keras
```

Рисунок 3.4 – Підключення бібліотеки tensorflow з набором даних Fashion Mnist

Для відображення результатів тренування було зроблено візуалізацію результатів по классам. На рисунку 3.5 представлено 10 класів одягу від Fashion Mnist, і як модель працює під час навчання. Можна помітити, що вона допускає помилки при навчанні, але це залежить від часу, витраченого на тренування. Також це залежить від кількості тренувальних зображень, які потрапляють до моделі, їх якості і кольору. Для того щоб зменшувати кількість помилок, потрібно використовувати якнайбільше різних зображень, щоб модель мала змогу аналізувати різноманітні зображення і знаходити закономірності між ними.

Рисунок 3.5 – Візуалізація результатів тренування

Таким чином, після тренування, модель надає ваги, які будуть в подальшому використані в основній моделі для аналізу, тобто заздалегідь виявлені особливості для кожного класу, що значно допоможе пришвидшити аналіз і покращити точність.

Для аналізу брендів було створено схожу модель, яка працює за таким самим принципом, як і для типів одягу. Але оскільки цього датасету не існує, як

для типів одягу, модель не підключає Fashion Mnist. Для підключення використовується створення змінної, яка містить шлях до папок з зображеннями. Шлях до розташування зображень має бути абсолютним і містити лише латинські символи, для того щоб модель могла правильно його знайти.

А також, оскільки в ньому немає заздалегідь підготованих класів, створюється масив з брендів, на яких модель тренується, щоб в подальшому вона могла їх класифікувати. Масив представлений на рисунку 3.6.

```
fashion_brands = ['Adidas', 'Puma', 'NewBalance', 'prada', 'lacoste',
                  'Gucci', 'louis vuitton', 'Balenciaga', 'Chrome Hearts',
                  'Converse', 'Columbia', 'Hugo Boss', 'levis',
                  'Ralph Lauren', 'versace', 'zara']
```

Рисунок 3.6 – Масив брендів для класифікації

Після навчання, модель також зберігає файл з вагами для того, щоб використати їх в основній моделі, але цей файл вже містить ваги для логотипів і особливості кожного логотипу. Всі ці ваги використовуються в основній моделі для того, щоб проводити класифікацію брендів. Оскільки ці фільтри містять інформацію, яку набула модель під час тренування, вони дуже допомагають в аналізі, оскільки вже заздалегідь навчені знаходити візерунки або закономірності в логотипах. Для коректного збереження ваг, потрібно забезпечити правильне тренування моделі, яка підійде під основну модель, тому що найважливішим є при використанні навчених ваг, щоб вони підходили до тих, які використовує сама модель. Після перевірки потрібно зберігати кожен новий варіант ваг на кожній ітерації навчання. Після останньої файл буде збережено за вказаним шляхом і міститиме інформацію про навчання, яку було набуто протягом всіх епох, що було пройдено під час навчання.

Далі ці файли використовуються в основній моделі. Ця модель має клас під назвою «ClothingImageAnalyzer», який виконує такі методи: setup_models(), setup_logo_detection_model(), detect_logo(), preprocess_for_fashion_model(), read_image(), extract_features(), analyze_content,() extract_dominant_colors(), name_color(), generate_description_with_llm(), analyze_image(), clean_path(). Вони

всі взаємопов’язані між собою і працюють послідовно. Ці методи обробляють зображення і виконують певні операції над перетворенням, або збиранням інформації, а також містять допоміжні функції, для покращення роботи програми. Призначення цих методів в класі описано в таблиці 3.4.

Таблиця 3.4 – Основні методи моделі

Метод	Призначення
setup_models()	Підключає основну модель для аналізу зображень і додає натреновані ваги. Також підключає архітектуру ResNet-18
setup_logo_detection_model()	Підключає ваги для аналізу логотипу
detect_logo()	Обробляє логотип і повертає результат аналізу
preprocess_for_fashion_model ()	Допоміжний метод для попередної обробки зображення
read_image()	Метод для зчитування зображення
extract_features()	Метод для витягування основних ознак (колір, сезонність, стиль)
analyze_content()	Метод для аналізу основних ознак
extract_dominant_colors()	Збирає основні кольори на зображенні в hex форматі
name_color()	Дає назву кольору за його hex значенням
generate_description_with_llm()	Підключає тестову модель і генерує опис на основі аналізу
analyze_image()	Головний метод для аналізу зображення, який запускає всі інші методи до зображення і також виводить результаті в консоль
clean_path()	Допоміжний метод для подання правильного шляху

Отже, реалізація цих основних методів створює велику нейронну модель, яка за допомогою навчених ваг проводить аналіз зображення. Під час аналізу вона спочатку виявляє чи є логотип на зображенні, далі витягує основні ознаки, аналізує їх, аналізує колір і дає йому назву. Після чого проводиться основний аналіз на тип одягу, після чого в кінці виконується аналіз бренду, якщо він наявний. По завершенню застосунок підключає текстову модель gpt-2, яка генерує текстовий опис на основі всіх даних, які були отримані в результаті аналізу і передає його до клієнту і серверної частини, щоб зберегти і відобразити. Запит для текстової моделі виглядає так: «Write a compelling product description for a {color} {pattern} {style} {clothing_type}{brand_text} made of {material}». Тобто вона бере результати аналізу і підставляє замість змінних.

Таким чином, виконується аналіз зображення, і модель генерує опис на основі цього аналізу, в результаті чого користувач бачить згенерований опис для товару. Важливим є те, що аналіз проводиться швидко і безкоштовно, створення такої моделі є важливим технологічним рішенням для управління онлайн-магазинами, оскільки це значно розширює і спрощує роботу різним користувачам або працівникам. Створений застосунок є зручним і зрозумілим, а створена нейронна модель є швидкою і точною, що є практичним і надійним рішенням в сфері онлайн-торгівлі.

3.4 Демонстрація застосунку

Результати роботи нейрнонної мережі було апробовано в умовах реального часу для онлайн-магазину. На першій сторінці було випробувано мережу для створення підказок. Після того як користувач створив задачу, її коректно було відображено на сторінці.

Для коректного користування сторінкою «Task manager» користувач повинен виконати наступні дії:

Крок 1. В полях «task name» і «task description» ввести правильні назву і опис для задачі.

Крок 2. Обрати потрібний час, до якого задача має бути завершена.

Крок 3. Обрати пріоритет для задачі.

Після цього, задача відображається в одному з бордів, за замовчуванням існує 3: «To Do», «In Progress» і «Completed».

Крок 4. Натиснути на задачу. Після відкриття меню, натиснути на кнопку «Get Tips», для отримання підказок по виконанню задач.

Крок 5. Після завершення задачі, натиснути на іконку у вигляді галочки, або на хрестик, якщо задача має бути видалено.

Крок 6. В блоці архів задач переглянути всі, які були виконані раніше, за необхідності, при натисканні на кнопку «Delete all», можна видалити всі задачі з архіву.

Таким чином відбувається процес керування задачами за допомогою розробленого вебзастосунку. За потреби, адміністратор застосунку може надавати користувачам доступ до бази даних, де міститься детальна інформація про всі задачі, а саме назва, опис, id, дата створення або виконання, статус і пріоритет. Відображення створених задач в дошках на головній сторінці «Task manager» показано на рисунку 3.7. Ці задачі можна помічати як виконані, або видаляти, або перетягувати між дошками, в залежності від того, на якому етапі вона знаходиться.

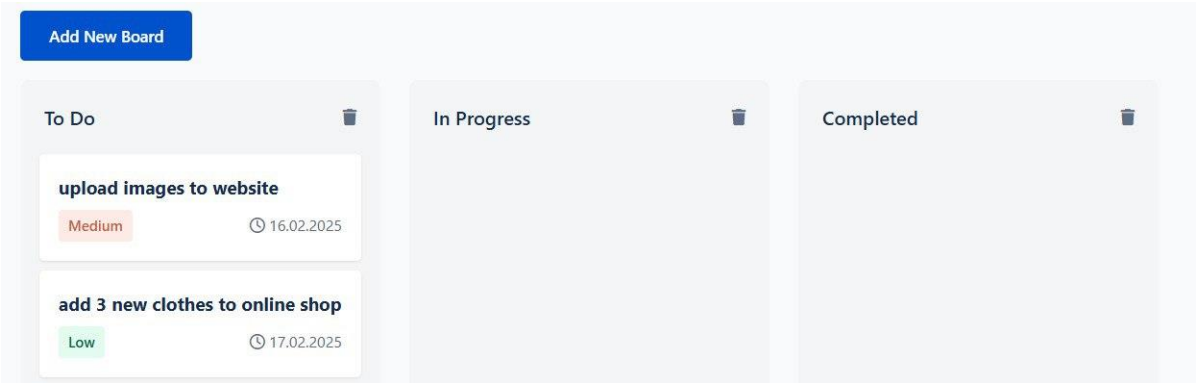

Рисунок 3.7 – Відображення задач в дошці

Додатково, користувачі можуть переглядати виконані задачі, за допомогою архіву задач. В цьому блоці міститься інформація про виконані задачі, які можна переглядати, або видаляти цей список. Також можна видаляти

задачі, які ще не були виконані. Окремою особливістю застосунку є можливість отримати підказки до задач. Для цього користувач має створити задачу, що містить назву і опис до неї, це є важливо, оскільки запит базується саме на основі цих двох полів. Підказки можуть бути створені будь-якою мовою, це залежить від того, якою мовою створена назва і опис задачі. Щоб побачити опис задачі, потрібно на неї натиснути і відкриється спеціальне спливаюче вікно, в якому відображно детальну інформацію (рис. 3.8). Також вікно містить дві кнопки, які відповідають за створення текстових підказок до виконання задачі, і також видалення її з сторінки і бази даних.

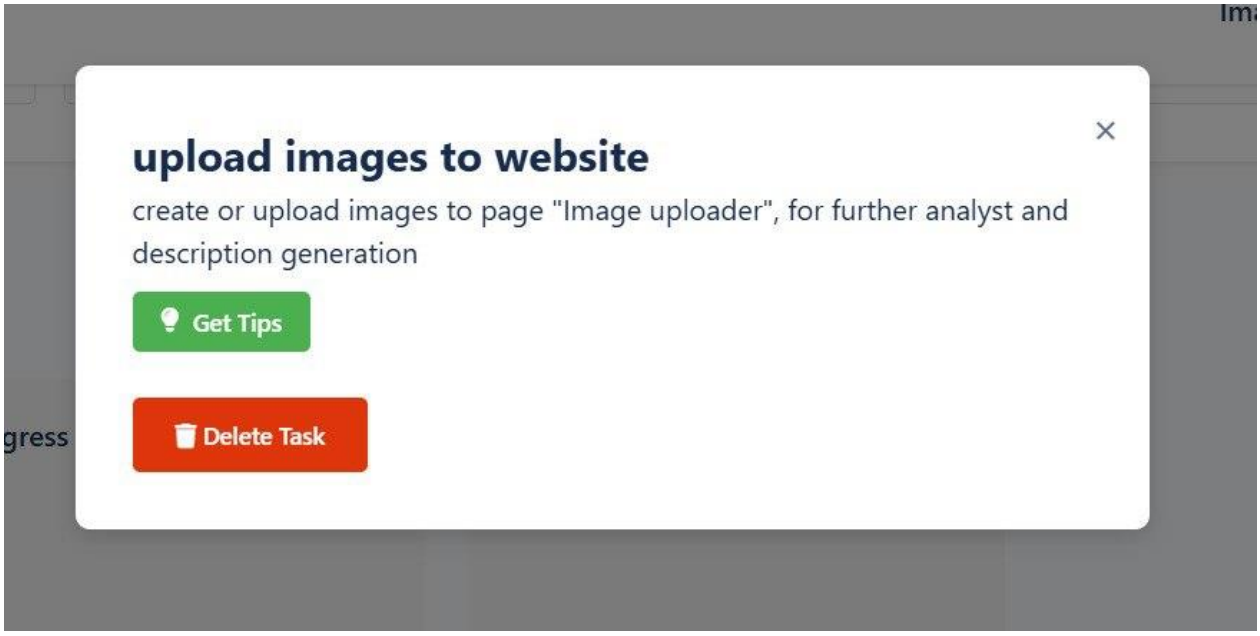

Рисунок 3.8 – Спливаюче меню задачі

Далі, для того щоб отримати підказки до виконання поставленої задачі, користувач має натиснути на кнопку «Get Tips» (рис. 3.9). Або видалити задачу, якщо вона містить непотрібну, або неправильну інформацію. Реалізація відобарження додаткових даних у спливаючому меню є зручним, оскільки це дає можливість додати інформацію і функціонал, який буде заважати користувачам при звичайному перегляді сторінки. Більш розширені можливості зручніше відображати в таких меню, оскільки це також може вплинути на завантаження і оптимізацію сторінки, оскільки програмний код, який виконує відкриття такого меню завантажується вже після повного відкриття сторінки.

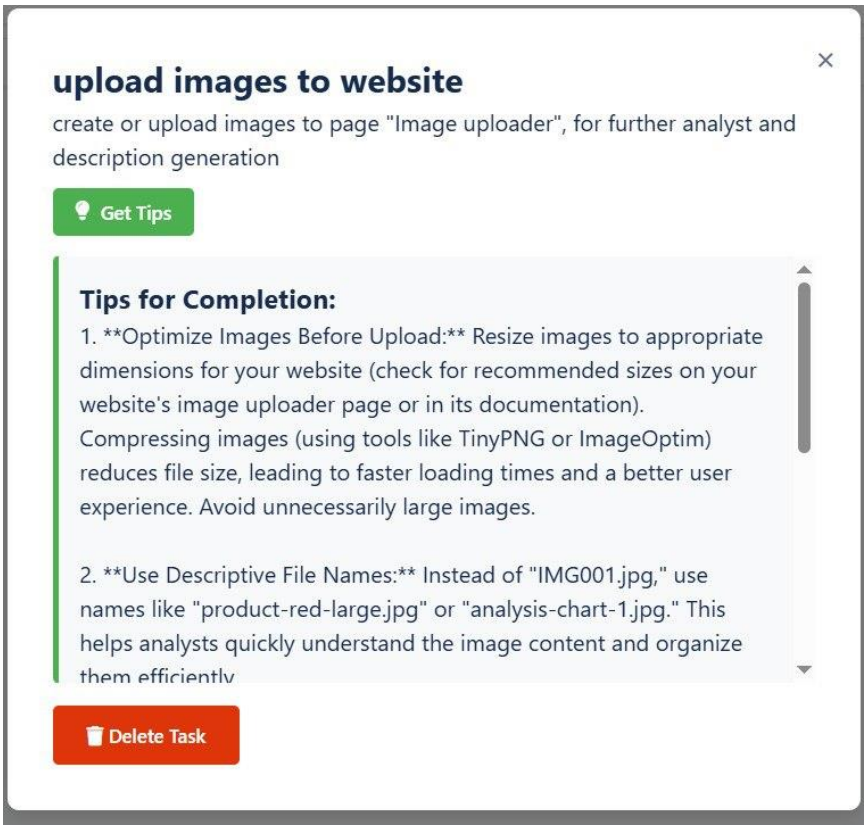

Рисунок 3.9 – Результат створення підказок від LLM Gemini

Ця сторінка зосереджена на управлінні задачами в онлайн-магазинах і надає відповідний функціонал. Користувач отримує розширений список інструментів, які дають змогу створювати, оновлювати і видаляти задачі. Особливо важливим є можливість отримувати підказки по виконанню задач за допомогою текстової нейронної моделі Gemini.

Для правильного користування сторінкою Image uploader, користувач повинен виконати наступні кроки:

Крок 1. Для переходу на сторінку, користувач повинен її обрати у верхньому меню, на сторінці «Task manager» і натиснути на неї. Після чого він потрапляє на сторінку, де можна переглядати існуючі зображення, додавати нові і створювати описи до них.

При переході користувача на сторінку «Image uploader» він бачить наступний інтерфейс (рис. 3.10). Інтерфейс містить головний блок, за допомогою якого можна завантажувати різні зображення, а також блок з зображеннями, які були завантажені раніше, їх можна відображати у вигляді сітки, або окремих рядків.

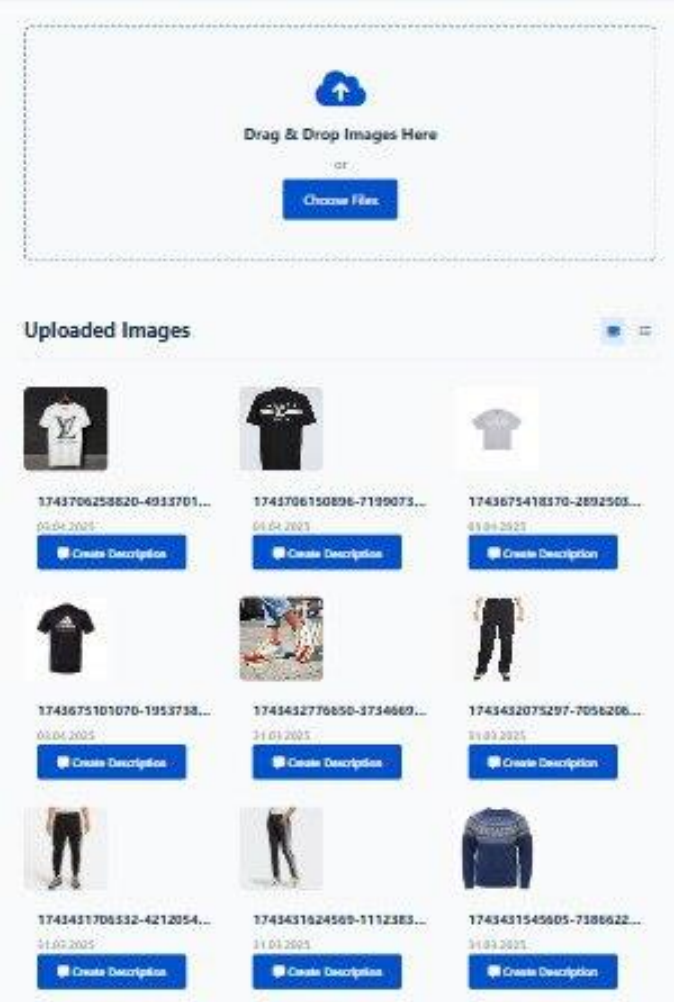

Рисунок 3.10 – Інтерфейс сторінки Image uploader

Крок 2. Натиснути на кнопку «Choose files», або перетягнути потрібне зображення для завантаження.

Крок 3. Переглянути завантажені зображення і натиснути на кнопку «Create description» щоб запустити процес аналізу зображення.

Далі, програма почне виконання і користувач буде бачити підпис «Analyzing», поки програма в роботі, і опис по її завершенню. Також, адміністратор програми може надати користувачу доступ до бази даних, де міститься детальна інформація про кожне зображення, а саме, id, шлях зображення, дата завантаження, вага файлу і створений опис. Процес виконання програми зображено на рисунку 3.11. Користувач має змогу замінити створений опис, натиснувши кнопку повторно, база даних постійно оновлює значення і може зберігати нові описи. Це може допомогти при заповненні інформації про товар на різних платформах, або для того, щоб оновити застарілу інформацію.

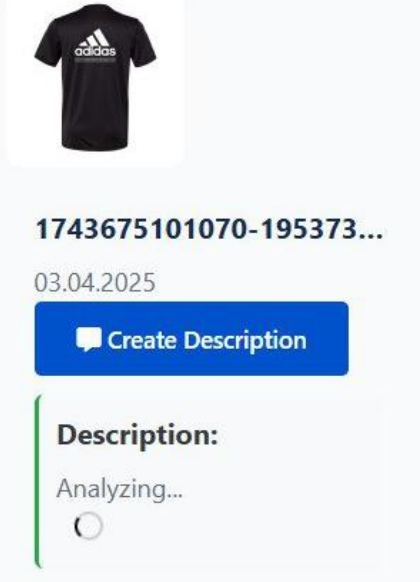

Рисунок 3.11 – Виконання програми

Під час роботи програми користувач бачить надпис «Analyzing…» і іконку завантаження, це відбувається тому, що запущено процес аналізу зображення, і поки програма працює, користувач буде бачити цей надпис. По закінченню програми на екрані користувача відобразиться створений нейронною моделлю опис (рис. 3.12). При перезавантаженні сторінки, процес аналізу зображення процес переривається, тому важливо дочекатись коли опис з’явиться на екрані. Опис також можна перегенерувати, якщо він не підходить, або не подобається. Для цього потрібно просто повторно натиснути на кнопку «Create Description», і програма заново почне аналіз і створення нового опису, не повторюючи попередній.

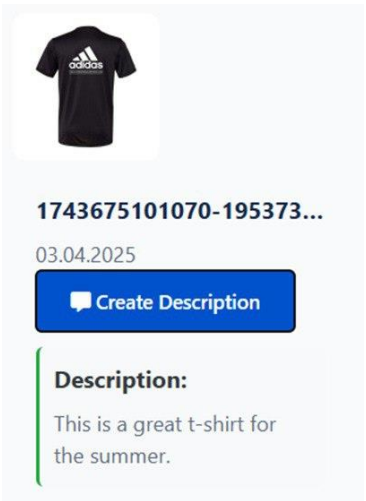

Рисунок 3.12 – Результат роботи програми

Додатково, адміністратор програми може надати детальний процес аналізу зображення, який відображається в терміналі програми, там описаний кожен крок виконання програми, а також відображено більш детальний результат аналізу, який включає в себе тип одягу, бренд, колір, сезон і точність бренду. Для того, щоб побачити як виконувався аналіз програми, потрібно відкрити програмне середовище, на якому розгорнуто проєкт і зайти на вкладку «Terminal». Після натиснення користувачем на кнопку «Create Description», розпочнеться виконання програми. Кожен крок виконання буде відповідно відображатись в терміналі програми, основними етапами буде: знаходження зображення в базі за відповідним шляхом, підключення моделей для аналізу і створення опису, аналіз типу одягу, кольору, сезону і бренду, збирання всіх даних під час аналізу, створення текстового опису для товару. Детальний результат аналізу зображення представлено на рисунку 3.13.

Отже, створена нейронна модель детально аналізує зображення, визначає тип одягу на ньому, і класифікує бренд, та його точність. Додатково мережа визначає колір, матеріал і сезон для типу одягу, щоб покращити опис товару. Всі кроки аналізу описуються в терміналі середовища, а в вебінтерфейсі користувач бачить створений опис до товару.

Рисунок 3.13 – Детальний результат аналізу зображення

Таким чином, застосунок є досить зрозумілим у використанні для будь-якого користувача, і може бути доступним з різних пристроїв. Попри легкість у використанні, він містить багато функцій, які допомагають при управлінні онлайн-магазинами і значно спрощує роботу для програмістів, контент-менеджерів і навіть проджект-менеджерів. У застосунку дотримано балансу між точністю і швидкістю, що робить його гнучким під різні пристрої. А також він є повністю безкоштовним, що робить його доступним для будь-кого. Ще одним важливим плюсом є те, що він є безпечним і зберігає конфіденційність кожного користувача.

При розробці застосунку було використано сучасні алгоритми і технології штучного інтелекту, враховуючи всі аспекти вже існуючих застосунків для аналізу зображень і управлінням задачами.

ВИСНОВКИ

У рамках роботи було створено вебзастосунок для управління онлайн-магазином з провадженням технологій штучного інтелекту для створення текстових підказок, і створення нейронної моделі для аналізу зображень з класифікацією типів одягу і брендів на них.

Результати роботи демонструють важливість моделей штучного інтелекту в різних сферах діяльності, зокрема, в сфері онлайн-торгівлі і маркетингу. Цей застосунок значно полегшує і пришвидшує процес управління онлайн-магазином.

Створена нейронна модель якісно виконує поставлену задачу, має високу точність і забезпечує конфіденційність користувачів при використанні застосунку. Також вона є безкоштовною і може використовуватись в умовах реального часу, що робить її потужним інструментом в сфері онлайн-торгівлі.

Нейронна модель має можливість інтеграції в інші застосунки, що робить її гнучкою. А також вона адаптована під класифікацію зображень з погіршеною якістю і для логотипів, які видно не повністю.

Даний застосунок можна використовувати в різних організаціях, що надають послуги онлайн-магазину, або займаються їх підтримкою. Впровадження цього вебзастосунку позитивно вплине на роботу, оскільки має багато потрібних інструментів для такого роду діяльності. Цей застосунок також є зручним і легким в управлінні, що дає змогу будь-кому ним користуватись.

Подальші дослідження і розробки будуть зосереджені на оптимізації процесів і створенні моделі, що перетворює зображення в 3D-об’єкт, який може відобразити повну модель товару і розпізнати його. Також буде покращено роботу застосунку шляхом постійного оновлення і додавання нових брендів і типів одягу, для збільшення точності і швидкості виконання. Додатково можливе додавання функцій реєстрації і входу користувачів під різними ролями, для того щоб кожен міг виконувати свою певну роботу, або мати певні права і обмеження в застосунку. Окрім вебзастосунку можлива розробка мобільного застосунку в

майбутньому, це дозволить ще більш гнучко користуватись застосунком і аналізувати зображення навіть на слабких або малооптимізованих пристроях.

Ця робота є черговим кроком у застосуванні нейронних мереж для класифікації зображень. Впровадження такої технології не тільки покращує користувацький досвід і конфіденційність, але й відкриває нові шляхи для досліджень і розробок у галузі аналізу зображень. Щорічно з’являються нові застосунки і нейронні моделі, які постійно оновлюються і покращуються, що свідчить про актуальність і важливість досліджуваної теми.

Дослідження було представлене у І етапі Всеукраїнського конкурсу студентських наукових робіт за напрямом «Інформатика і кібернетика» у вигляді звіту.

Результати дослідження апробовано у вигляді тез доповідей під час 29-го Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [32].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

2. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

3. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

4. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S»* (pp. 25-27).

5. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. IEEE Access, 11, 126938-126949.

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

7. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

8. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Mujahed, A. D. (2022). Classification of Images Based on a System of Hierarchical Features.

9. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

10. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

11. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

12. Gorokhovatskyi, V. A., Rusakova, N., & Tvoroshenko, I. S. (2020). The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79(20).

13. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

14. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Vlasenko, N. V. (2020). Using fuzzy clustering in structural methods of image classification. Telecommunications and Radio Engineering, 79(9).

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

16. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

17. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, 7(7), (pp. 25-36).

18. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

19. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

20. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

21. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. In COLINS (3) (pp. 69-86).

22. Yakovleva, O., Slyusar, V., Kushnir, O., & Sabovchyk, A. (2021). New trends in scientific and technological revolution (STR) and transformation of science and education systems in the paradigm of sustainable development. In E3S Web of Conferences (Vol. 277, p. 06006). EDP Sciences.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Scienc*, 33 (1), (pp. 113-125)

25. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 210-230). Springer, Cham.

26. Кобилін, О. А., & Путятіна, О. Є. (2024). Знешумлення зображень, зіпсованих дробовим шумом, у реальному часі. Системи обробки інформації, (1 (176), 46-51.

27. Gorokhovatskyi , V., Chmutov , Y., Tvoroshenko , I., & Kobylin , O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12. https://doi.org/10.20998/2522-9052.2025.1.01 (дата звернення 24.04.2025).

28. Кобилін, О., Вечірська, І., Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63–76, https://doi.org/10.32782/IT/2024-3-7 (дата звернення 15.04.2025)

29. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 97–107, https://doi.org/10.32782/IT/2024-3-10 (дата звернення 12.03.2025)

30. Vechirska, I., Kobylin, O., Prokopiev , S., Vechirska, A., & Kucherenko, M. (2022). Building a logical network for solving the problem of car rental by means algebra of finite predicates. Computer Systems and Information Technologies, (2), 78–87.

31. Kobylin, O.; Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. CEUR Workshop Proceedings ,volume = 3943 , 109-117

32. Федорук М. Ю. (2025) Аналіз, оброблення та подання даних у мультимедійних системах. *Радіоелектроніка та молодь у XXI столітті: Конференція «Комп'ютерний зір та мультимедійні системи»: матеріали 29-го тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.) ХНУРЕ, 2025. Т. 7. С. 154-155.*