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)

ВИКОРИСТАННЯ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ОДЯГУ У ВЕБЗАСТОСУНКУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1

Кравченко Д.С.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кравченку Денису Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання нейронної мережі для класифікації одягу у вебзастосунку

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 26 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, модель розпізнавання об'єктів YOLOv8, базовий набір даних із зразками одягу.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих рішень для класифікації одягу.
2. Вибір архітектури нейронної мережі.
3. Розробка методі для класифікації одягу.
4. Підбір технологічних рішень для розв'язання поставлених задач.
5. Створення програмного рішення, поставленої задачі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розпізнавання одягу на зображеннях, постановка задачі, тестові зображення, схема бази даних.

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	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-16.04.24	
3	Аналіз літератури з досліджуваної проблеми	17.04.24-20.04.24	
4	Аналіз технічних засобів	20.04.24-25.04.24	
5	Розробка методу	26.04.24-14.05.24	
6	Програмна реалізація	15.05.24-23.05.24	
7	Оформлення пояснювальної записки	25.05.24-29.05.24	
8	Перевірка на плагіат	30.05.24	
9	Рецензування	31.05.24	
10	Підготовка презентації та доповіді	01.05.24-04.06.24	
11	Занесення роботи в електронний архів	05.06.24	
12	Попередній захист кваліфікаційної роботи	05.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 71 с., 6 табл., 29 рис., 40 джерел.

КЛАСИФІКАЦІЯ ОДЯГУ, НЕЙРОННІ МЕРЕЖІ, ВЕБЗАСТОСУНОК, АЛГОРИТМИ НАВЧАННЯ, ВІЗУАЛЬНА ОБРОБКА ЗОБРАЖЕНЬ, РОЗРОБКА ВЕБЗАСТОСУНКІВ.

Об’єктом дослідження є зображення одягу, отримані з різних джерел, що включають соціальні мережі, вебсайти рітейлерів та користувацькі фото.

Метою роботи є розробка вебзастосунку, що використовує нейронні мережі для класифікації одягу за зображеннями, що надається користувачами. Використано сучасні архітектури нейронних мереж, такі як YOLOv8, для виявлення та класифікації об’єктів на зображеннях. Проведено аналіз існуючих рішень у цій області, вивчено кращі практики розробки вебзастосунків для моди, та розроблено методи навчання та прогнозування, адаптовані для високої точності та швидкості обробки.

У результаті роботи здійснена програмна реалізація системи для класифікації одягу у вебзастосунку.

CLOTHING CLASSIFICATION, NEURAL NETWORKS, WEB APPLICATION, LEARNING ALGORITHMS, VISUAL IMAGE PROCESSING, WEB APPLICATION DEVELOPMENT.

The object of the work is an image of clothes for online classification.

The aim of the work is to develop a web application that uses neural networks to classify clothes based on images provided by users. Modern neural network architectures, such as YOLOv8, are used to detect and classify objects in images. An analysis of existing solutions in this area was carried out, best practices for developing web applications for fashion were studied, and training and prediction methods adapted for high accuracy and processing speed were developed.

As a result of the work, the software implementation of the system for classifying clothes in the web application was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network (згорткова нейронна мережа)

YOLO – You Only Look Once (метод одноразового перегляду для детекції об’єктів)

MVC – Model-View-Controller (модель-вид-контролер, архітектурний паттерн для створення інтерфейсу користувача)

ASP.NET Core MVC – вебфреймворк для розробки вебзастосунків

SQL – Structured Query Language (структурована мова запитів)

UI – User Interface (інтерфейс користувача)

API – Application Programming Interface (інтерфейс програмування застосунків)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

JS – JavaScript (мова програмування JavaScript)

RGB – Red, Green, Blue (червоний, зелений, блакитний – модель визначення кольору)

HEX – Hexadecimal (шістнадцятковий формат запису чисел, використовується для визначення кольорів в вебдизайні)

HTTPS – Hypertext Transfer Protocol Secure (безпечний протокол передачі гіпертексту)

ID – Identifier (ідентифікатор)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

ВСТУП

Комп’ютерний зір і, зокрема, класифікація образів є критично важливими компонентами сучасної обробки інформації, що знаходить застосування в різноманітних галузях, від автоматичного контролю якості на виробництві до покращення інтерфейсів користувача в мобільних програмах і вебзастосунках. В контексті електронної комерції, зокрема у сфері торгівлі одягом, використання технологій базоване на штучному інтелекті, як-от нейронні мережі для класифікації товарів, стає не просто зручністю, а необхідністю.

Ця потреба випливає з кількох ключових трендів. Перше – збільшення обсягів даних: сучасні магазини пропонують тисячі одиниць одягу, кожна з яких має бути адекватно представлена і класифікована для ефективного пошуку. Друге – зростання вимог до персоналізації шопінгу: споживачі очікують, що системи рекомендацій будуть чутливими до їхніх індивідуальних уподобань. Третє – потреба у покращенні користувацького досвіду, що безпосередньо впливає на лояльність клієнтів та їхню готовність здійснювати повторні покупки.

Актуальність дослідження використання нейронних мереж для класифікації одягу у вебзастосунках безпосередньо пов’язана з динамічними змінами у сфері онлайн-торгівлі та зростаючим зацікавленням до цифрових технологій, що спостерігається серед місцевих підприємців.

Онлайн-торгівля в Україні та світі продовжує свій стрімкий розвиток, обумовлений пандемією COVID-19, яка значно прискорила перехід споживачів від традиційних магазинів до інтернет-платформ. Згідно з дослідженнями, обсяги електронної комерції зростають на 23% щороку, а український ринок онлайн-продажів продемонстрував значні темпи зростання, що робить цю сферу особливо привабливою для інвестицій та інновацій.

З іншого боку, постійне збільшення кількості інтернет-магазинів та розширення асортименту товарів вимагають від бізнесу інтеграції більш

сучасних технологій для підтримання конкурентоспроможності та ефективності обслуговування клієнтів. В цьому контексті, застосування нейронних мереж дозволяє не тільки оптимізувати процеси класифікації та пошуку товарів, але й підвищує якість взаємодії з клієнтом завдяки персоналізації та рекомендаціям.

Дослідження використання нейронних мереж у роздрібній торгівлі одягом через вебзастосунки важливе не тільки з комерційної точки зору, але й з погляду розвитку української науки і технологій. Воно відкриває широкі перспективи для подальших досліджень і можливостей інновацій, підвищуючи потенціал українських дослідницьких інститутів та IT-компаній у створенні новітніх продуктів та сервісів.

Цей напрямок дозволяє інтегрувати передові досягнення у галузі машинного навчання та штучного інтелекту безпосередньо у повсякденну комерційну діяльність, забезпечуючи тим самим практичну цінність теоретичних знань. Окрім цього, розвиток нейронних мереж сприяє формуванню нових академічних програм, курсів та спеціальностей, що зосереджені на вивченні та вдосконаленні технологій глибокого навчання, дата-аналітики та алгоритмічних рішень для різних сфер економіки.

Шляхом критичного аналізу відомих рішень, таких як системи класифікації одягу Amazon або Google, можна виявити значні недоліки у точності і адаптивності до мінливих модних тенденцій. Ці відкриття обґрунтовують необхідність проведення подальших досліджень у цій області, спрямованих на створення більш ефективних та точних систем, які можуть значно покращити якість відбору та класифікації одягу на основі візуальних характеристик.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих рішень для класифікації одягу

У сучасному світі роздрібної торгівлі велике значення має здатність швидко та точно класифікувати товари, особливо коли мова йде про одяг. Програмне забезпечення для класифікації одягу стає незамінним інструментом в онлайн-торгівлі, де важливо не тільки правильно сортувати товар за категоріями, але й забезпечити користувачам зручний пошук та відповідні рекомендації. Ці системи значно спрощують управління великим асортиментом, допомагають автоматизувати введення нових товарів та забезпечують їх відповідність запитам споживачів.

Класифікація одягу за допомогою програмного забезпечення базується на різних параметрах, таких як колір, розмір, стиль, матеріал та інше [1]. Ці критерії дозволяють точно категоризувати продукти, полегшуючи процеси пошуку та фільтрації для кінцевих користувачів. Ефективність цих систем прямо впливає на зручність покупок у магазині, а також на операційну ефективність бізнесу, допомагаючи скоротити час на обробку даних та збільшити задоволеність клієнтів.

Програмне забезпечення для класифікації одягу не лише спрощує управління асортиментом на складі та в електронному каталозі, але й відіграє ключову роль у підтримці стратегій маркетингу та продажу. Воно дозволяє впроваджувати комплексні аналітичні інструменти для вивчення попиту та споживацьких переваг, а також підтримує вдосконалення персоналізації комерційних пропозицій.

ViSenze є одним із провідних рішень у галузі візуального пошуку та автоматичної класифікації товарів, зокрема одягу, для електронної комерції (рис. 1.1) [2]. Цей застосунок використовується для поліпшення взаємодії між онлайн-магазинами та їхніми клієнтами шляхом дозволу споживачам шукати продукти, використовуючи зображення замість тексту.

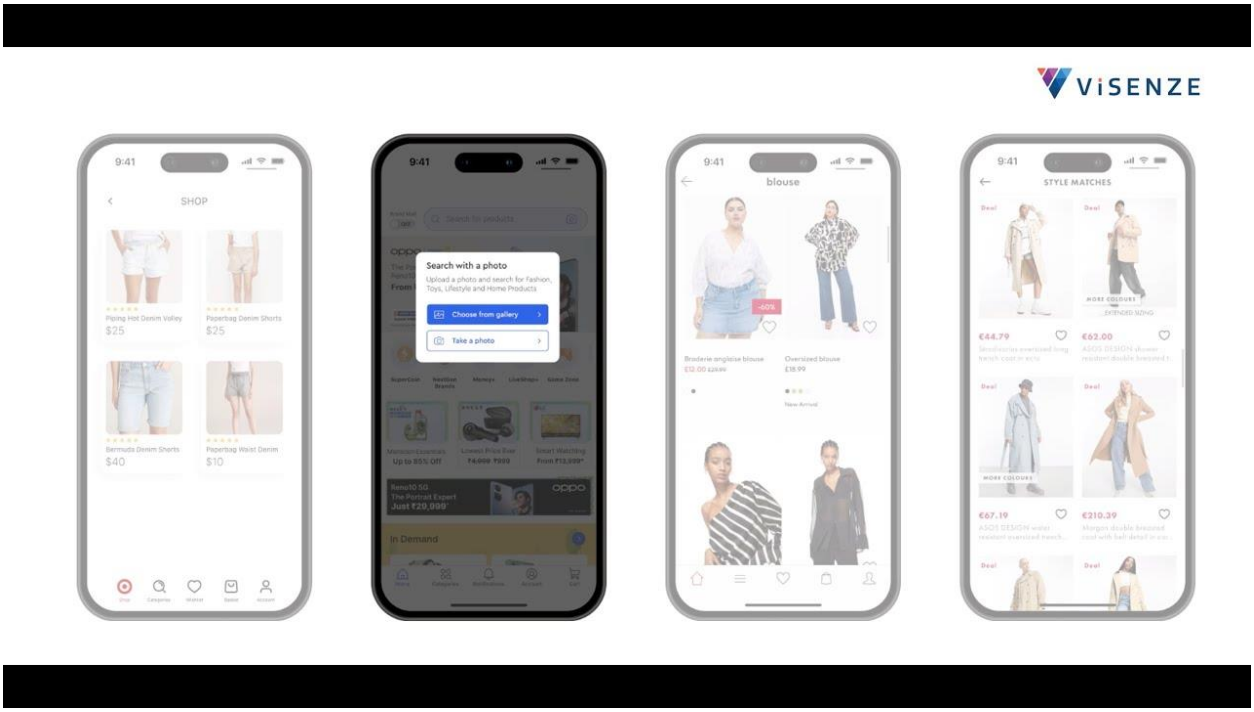

Рисунок 1.1 – Приклад користувацького інтерфейсу «ViSenze»

Архітектура ViSenze базується на глибокому навчанні та комп'ютерному зорі, що дозволяє їй ефективно обробляти та аналізувати великі обсяги візуальних даних [3]. Система використовує нейронні мережі для розпізнавання і класифікації різних атрибутів продуктів, таких як стиль, колір і бренд, з метою забезпечення точних та релевантних результатів пошуку.

ViSenze також інтегрована з різними електронними комерційними платформами, що дозволяє рітейлерам легко впроваджувати її функціональність у свої онлайн-магазини, поліпшуючи тим самим користувацький досвід та підвищуючи конверсію продажів.

Переваги:

– візуальний пошук. ViSenze дозволяє користувачам шукати продукти за допомогою зображень замість тексту, що підвищує зручність і швидкість пошуку;

– інтеграція з платформами електронної комерції [4]. Програма легко інтегрується з різними платформами електронної комерції, що дозволяє

рітейлерам швидко впроваджувати її у свої системи без значних затрат на розробку;

– підвищення конверсії продажів. Функціональність візуального пошуку покращує користувацький досвід, що може призвести до збільшення конверсії продажів та загальної ефективності онлайн-магазинів;

– точність розпізнавання. Використання передових технологій глибокого навчання та комп'ютерного зору забезпечує високу точність розпізнавання та класифікації продуктів за різними категоріями.

Недоліки:

– залежність від якості зображень. Ефективність системи залежить від якості наданих зображень. Низька якість або недостатня деталізація зображень може негативно впливати на результати пошуку;

– обмеження на кількість визначень атрибутів [5]. Попри те що система добре справляється з розпізнаванням базових атрибутів, може бути обмежена у визначенні більш складних або незвичайних атрибутів продуктів;

– висока вартість. Хоча точна вартість може варіюватися, впровадження та підтримка таких рішень можуть бути витратними, особливо для малих та середніх підприємств;

– потреба в технічному супроводі. Інтеграція та ефективне управління ViSenze може вимагати кваліфікованого технічного персоналу для налаштування та підтримки системи, що може бути бар'єром для деяких компаній.

Отже, ViSenze є потужним інструментом для візуального пошуку та класифікації продуктів, зокрема одягу, у сфері електронної комерції. Його здатність використовувати технології глибокого навчання для аналізу зображень значно покращує досвід користувачів, дозволяючи їм швидко знаходити товари за допомогою фотографій. Це не тільки сприяє підвищенню конверсії продажів, але й оптимізує управління продуктовим асортиментом. Проте, ViSenze вимагає високоякісних зображень для досягнення оптимальної точності та може бути вартісним у впровадженні та підтримці, що робить його

більш підходящим для великих компаній з достатніми технічними та фінансовими ресурсами.

Cortexica є застосунком для візуального пошуку і аналізу зображень, спеціалізованим на розпізнаванні та класифікації об’єктів у роздрібній торгівлі, зокрема в секторі моди та одягу (рис. 1.2) [6]. Призначенням Cortexica є забезпечення підприємств та їхніх клієнтів зручними інструментами для швидкого і точного виявлення та порівняння товарів через мобільні пристрої та онлайн-платформи. За допомогою передових технологій штучного інтелекту та машинного навчання, Cortexica дозволяє користувачам завантажувати фотографії товарів, щоб швидко знайти подібні предмети або отримати додаткову інформацію про них в онлайн-магазинах.

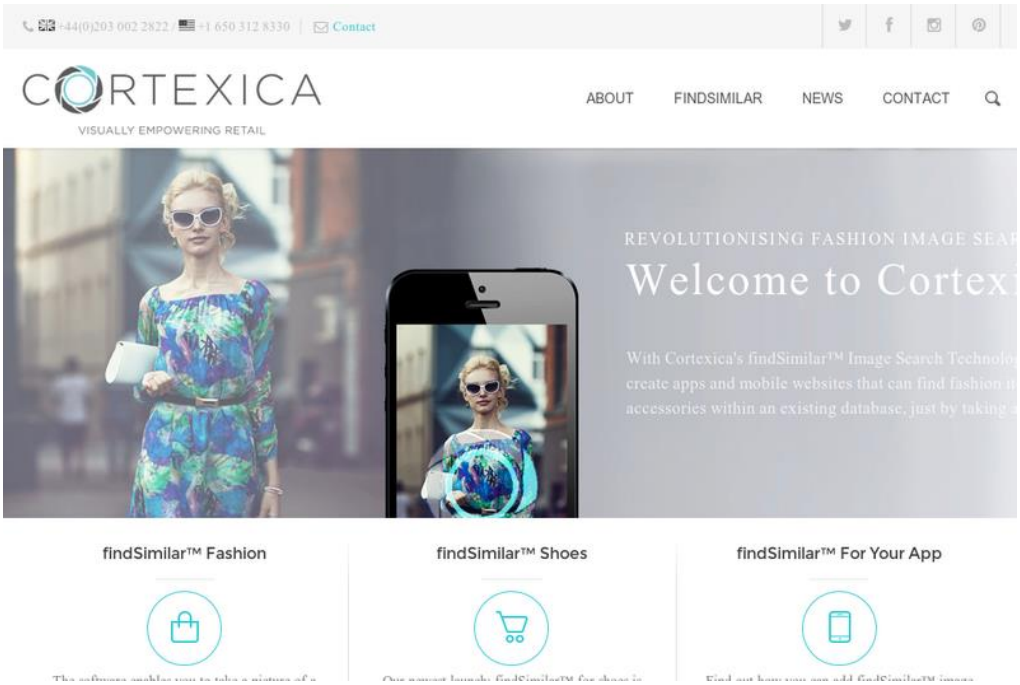

Рисунок 1.2 – Приклад користувацького інтерфейсу «Cortexica»

Архітектура Cortexica заснована на використанні нейронних мереж, що дозволяє ефективно обробляти великі обсяги візуальних даних, виявляючи закономірності та особливості в зображеннях [7]. Система здатна адаптуватися до різноманітних стилів і трендів у моді, постійно оновлюючись завдяки накопиченню і аналізу користувацьких даних. Ця функціональність не тільки

підвищує точність пошуку та класифікації, але й дозволяє рітейлерам краще розуміти переваги своїх клієнтів, оптимізуючи тим самим маркетингові стратегії та управління запасами.

Переваги сервісу Cortexica:

– точність візуального пошуку [8]. Cortexica використовує передові технології штучного інтелекту та машинного навчання для точного розпізнавання зображень, що значно підвищує якість візуального пошуку;

– покращення користувацького досвіду. Здатність швидко і точно знаходити товари за зображеннями робить шопінг більш зручним і приємним для клієнтів;

– гнучкість інтеграції [9]. Cortexica легко інтегрується з існуючими платформами електронної комерції, що дозволяє рітейлерам впроваджувати функціонал візуального пошуку без необхідності великих змін в їхніх системах;

– збір та аналіз даних про користувацькі переваги. Інструменти аналітики дозволяють збирати цінну інформацію про переваги і поведінку клієнтів, що може бути використане для оптимізації маркетингових стратегій.

Недоліки Cortexica:

– залежність від якості зображень. Низька якість зображень може негативно вплинути на точність розпізнавання та класифікації продуктів, обмежуючи ефективність застосунку;

– складність управління великими даними [10]. Обробка великих обсягів даних потребує значних обчислювальних ресурсів, що може становити виклик для деяких користувачів або малих підприємств;

– вартість впровадження. Початкові витрати на ліцензування та інтеграцію Cortexica можуть бути високими, що робить її менш доступною для малих та середніх підприємств;

– технічне обслуговування та оновлення. Підтримка і оновлення системи вимагають регулярного технічного обслуговування, що може бути додатковим фінансовим та організаційним тягарем.

Отже, Cortexica є високоефективним інструментом для візуального пошуку та класифікації товарів, який забезпечує точне і швидке розпізнавання продуктів на основі зображень. Цей застосунок підвищує користувацький досвід, збільшує конверсію продажів та допомагає рітейлерам оптимізувати свої маркетингові стратегії завдяки точному аналізу даних. Незважаючи на потребу у високій якості зображень та вартість впровадження, Cortexica вважається важливим рішенням для тих, хто шукає ефективні технології в області електронної комерції, здатні впоратися з викликами сучасного ринку.

1.2 Огляд використання нейронних мереж в класифікації зображень

Класифікація зображень, як класична наукова тема у останні роки, є однією з ключових проблем комп'ютерного зору та основою різноманітних областей візуального розпізнавання. Покращення продуктивності мережі класифікації має тенденцію значно підвищувати її рівень застосування [11], наприклад, для виявлення об'єктів, сегментації, оцінки пози людини [12], класифікації відео, відстеження об'єктів та технології суперроздільності [13]. Покращення технології класифікації зображень є важливою частиною просування розвитку комп'ютерного зору. Її основний процес включає попередню обробку даних зображень [14], вилучення та представлення ознак і проєктування класифікатора [15].

Увага дослідження класифікації зображень завжди була спрямована на вилучення ознак зображень, яке є основою класифікації зображень. Традиційні алгоритми вилучення ознак зображень більше зосереджуються на ручному встановленні конкретних ознак зображень. Цей метод має низьку загальну здатність узагальнення та переносу. Таким чином, надання комп'ютеру здатності обробляти зображення аналогічно біологічному зору – це те, чого мріють дослідники. Штучна нейронна мережа (ANN) є абстрактною біологічною нейронною мережею, яка є математичною моделлю операцій,

складених з великої кількості взаємопов’язаних нейронів. Вона приблизно імітує обробку нейронною мережею нейрональних сигналів. Спочатку Маккалох та Пітс проаналізували біологічні нейронні мережі та запропонували внутрішню логічну операційну математичну модель активності нейрона – модель нейрона MP [16]. Розенблатт додав функції навчання до моделі MP та запропонував модель одношарового персептрону, вперше втіливши дослідження нейронних мереж у практику [17].

Після цього Хубер та Візе та ін. вивчали візуальний кортекс мозку кота та виявили, що біологічні візуальні нейрони сприймають інформацію на основі локальної регіональної стимуляції, вони прийшли до висновку, що візуальне сприйняття стимулюється шар за шаром через багаторівневі поля прийому. Пізніше дослідники спробували використовувати багаторівневий персептрон для вивчення ознак та тренували модель з використанням алгоритму зворотного поширення помилки [18]. Це відкриття надихнуло дослідників на створення комп’ютерної нейронної мережі, подібної до біологічної системи зору, що стало реальністю, і CNN було створено. Лекун та ін. представили першу партію моделей CNN – LeNet-5 [19]. Однак через відсутність масштабних навчальних даних ця модель також обмежується теоретичними основами та обчислювальною потужністю комп’ютера, результати визнання LeNet-5 на складних зображеннях не були ідеальними. У той час ця модель показувала відмінні результати лише на завданнях розпізнавання почерку.

Джефрі Хінтон та ін. запропонували ефективний алгоритм навчання для подолання труднощів у багаторівневих нейронних мережах [20], тим самим відкривши нову сторінку у глибокому навчанні. Пізніше дослідники реалізували операцію згортки на GPU, що значно покращило обчислювальну ефективність мережі. Порівняно зі швидкістю операцій CPU, вона зросла від 2 до 24 разів [18]. З того часу глибоке навчання привернуло все більше уваги. Крижевський та ін. побудували модель AlexNet на основі LeNet-5. На конкурсі ILSVRC2012 ImageNet вона перевершила другий найкращий внесок значним перевагом. Після того як AlexNet досягла відмінних результатів у змаганні з

класифікації зображень ImageNet, дослідники почали глибше вивчати CNN. Зейлер та Фергус запропонували техніку візуалізації для розуміння CNN та запропонували модель ZFNet [21]. Мін Лін та ін. запропонували мережу NIN, що сприяла контролю кількості параметрів та кількості каналів. З 2017 року і до сьогодні, одна за одною з'являлися моделі з відмінною продуктивністю. CNN все більше демонструють незамінну перевагу в класифікації зображень.

З успішним застосуванням CNN у великомасштабних завданнях візуальної класифікації, близько 2015 року, застосування CNN нарешті набирає обертів у галузі аналізу зображень дистанційного зондування [22]. З'явилася різноманітність методів класифікації сцен на основі CNN, які використовують різні стратегії експлуатації CNN [23]. Загалом, методи класифікації сцен зображень дистанційного зондування на основі CNN можна поділити на три типи:

- використання попередньо навчених CNN як екстрактора ознак;
- тонке налаштування попередньо навчених CNN на наборі даних;
- глобальна ініціалізація ваг CNN для навчання.

Як відомо, метод класифікації зображень на основі CNN спочатку був призначений для комп'ютерного зору. Однак багато дослідників успішно застосували їх у галузі дистанційного зондування. Необхідно систематично узагальнити методи класифікації зображень на основі CNN, щоб дати дослідникам натхнення для їх нових робіт. Хоча існують деякі огляди стосовно CNN [24], вони не введені повністю всі класичні архітектури CNN. Цей огляд присвячений детальному розгляду розвитку практично всіх типових архітектур CNN у завданнях класифікації зображень і сподівається надати більше допомоги для натхнення при проєктуванні моделей CNN у галузі класифікації сцен зображень дистанційного зондування.

Біологічна нервова система – це мережа, складена з багатьох нейронів. Так само, нейрони є базовою обчислювальною одиницею штучних нейронних мереж. Принцип роботи полягає в тому, що кілька вхідних значень проходять математичне перетворення для отримання вихідного значення (рис. 1.3).

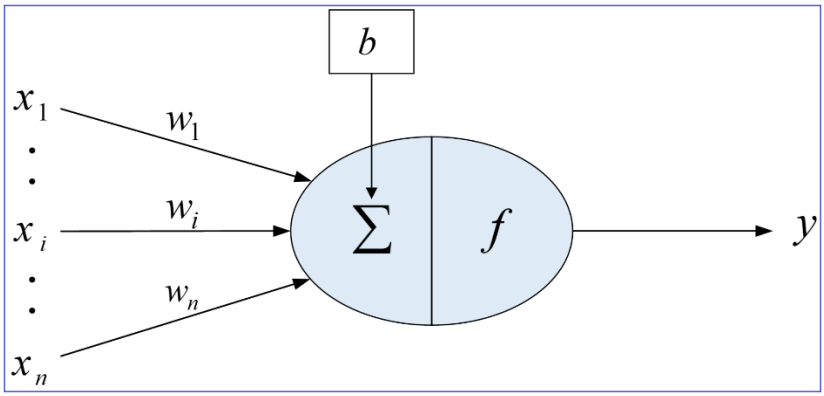

Рисунок 1.3 – Модель нейрона

Математичний взаємозв'язок між вхідним сигналом та вихідним значенням виглядає наступним чином:

$$f(b + \sum_{i=1}^{n}(x_i + w_i)) , \quad (1.1)$$

де, $f(\cdot)$ – це функція активації, існує багато функцій активації;

x_i – це вхідний сигнал;

n – кількість сигналів;

w_i – вага вхідного сигналу;

b – зсув;

y – вихід нейронів.

Штучна нейронна мережа (ШНМ) є одним з найпоширеніших типів нейронних мереж та складається з трьох основних компонентів: вхідного шару, прихованого шару (який може містити один або декілька шарів) і вихідного шару. Вхідний шар приймає вхідні дані, такі як значення пікселів зображення або вхідні ознаки, і передає їх до прихованого шару. Прихований шар акумулює та оброблює ці вхідні дані, використовуючи ваги та функцію активації, які визначаються під час навчання мережі. Кожен нейрон у прихованому шарі приймає вагову суму вхідних сигналів, додає до неї зсув та застосовує функцію активації для створення вихідного сигналу, який

передається наступному шару. Вихідний шар оброблює вихідні сигнали від прихованого шару та генерує остаточні вихідні значення. На рисунку 1.4 зображено приклад структури MLP, де стрілки вказують напрям передачі сигналів через різні шари мережі.

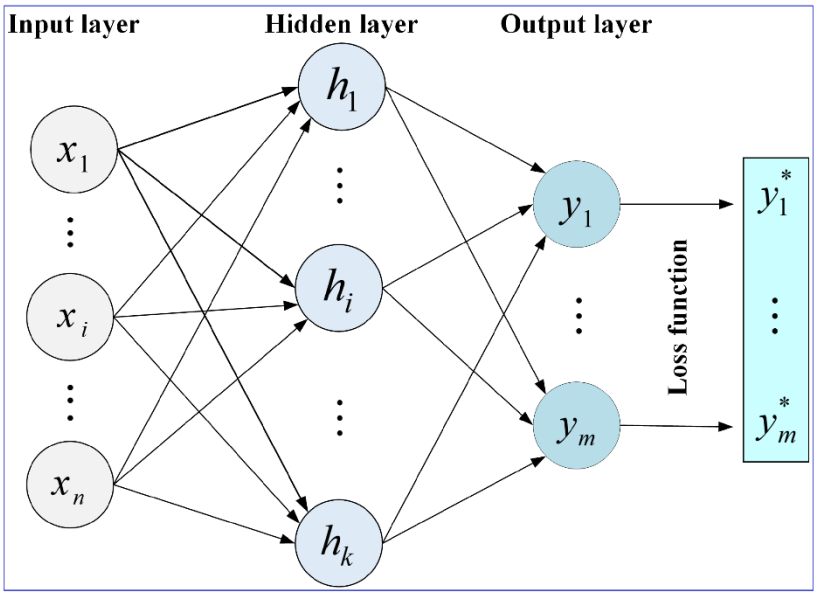

Рисунок 1.4 – Структура штучної нейронної мережі

Зважаючи на структурні аспекти ШНМ, процес обчислення вихідних значень прихованих та вихідних одиниць можна розглядати як послідовні кроки. Спочатку, для прихованого шару визначаються вихідні значення (позначені як H), які є результатом застосування функції активації до зваженої суми вхідних значень та зсуву. Це виконується за допомогою математичної формули:

$$H = F(W_h X + B_h)\,, \tag{1.2}$$

де W_h – це матриця ваг між вхідним і прихованим шарам;

X – матриця вхідних значень;

B_h – матриця зсуву для прихованого шару.

Далі, для вихідного шару обчислюються вихідні значення (позначені як Y) за допомогою аналогічного підходу, де використовується функція активації для обробки вагової суми вхідних значень прихованого шару та зсуву вихідного шару. Формула для цього процесу виглядає наступним чином:

$$Y = F(W_y H + B_y)\,, \tag{1.3}$$

де W_y – це матриця ваг між прихованим і вихідним шарам;

B_y – матриця зсуву для вихідного шару.

Зазначеної мережі властиві n вхідних значень (позначених як x_n) та m вихідних значень (позначених як y_m), крім того, вона включає k прихованих одиниць (позначених як h_k).

Стрілки на рисунку 1.4 вказують на напрям передачі вхідних значень через різні шари мережі. Прихована одиниця (h_k) отримує вхідне значення від попереднього шару та обчислює вихідне значення за допомогою зазначеного вище процесу. Вихідна одиниця (y_m) генерує вихідне значення, яке використовується для подальших аналітичних або класифікаційних завдань. Реальне значення, що отримується, може бути позначене як y_m^*.

Алгоритм зворотного поширення помилки поділяється на дві стадії: пряме поширення і зворотне поширення. Після обчислення вихідного шару пряме поширення завершується. Зворотне поширення включає оновлення різних параметрів, що є важливою частиною навчання мережі. Перш за все, для зворотного поширення потрібно визначити функцію втрат моделі:

$$Loss(y, y^*) = \frac{1}{m}\sum_{i=1}^{m}(y_i^* - y_i)^2\,, \tag{1.4}$$

Шляхом обчислення середньоквадратичної помилки між y^* та y, ваги мережі w та зсув b можна оновити, отримавши часткову похідну від функції втрат:

$$w' = w - \eta \cdot \frac{\partial Loss}{\partial w} \text{ і } b' = b - \eta \cdot \frac{\partial Loss}{\partial w} \quad (1.5)$$

З відповідним коефіцієнтом навчання η, втрата між y та y^* може бути поступово мінімізована. Це означає, що y може наблизитися до y^*, щоб досягти ефекту навчання мережі.

Загалом, класифікація зображень описує всі зображення шляхом вручну виділення ознак або методами навчання ознак та потім використовує класифікатор для ідентифікації категорії об'єкта. Тому особливо важливо вміти виділяти ознаки зображення. Класифікація об'єктів на основі моделі Bag of Words [25] широко використовується до глибокого навчання. Найпростіший каркас цієї моделі може бути розроблений як три процеси: екстракція ознак низького рівня, кодування ознак та проєктування класифікатора. Традиційний метод класифікації зображень до 2012 року може бути завершений за цими три кроки, але повне створення моделі класифікації зображень, як правило, включає кілька процесів, таких як навчання ознак низького рівня, кодування ознак, просторові обмеження, проєктування класифікатора та об'єднання моделі.

Поява CNN зробила серію проривів у галузі класифікації зображень та досягла відмінних результатів на великомасштабних візуальних завданнях [26]. Великий успіх глибоких CNN (DCNNs) приписується його сильним можливостям навчання ознак. У відмінність від традиційного методу класифікації зображень, метод класифікації на основі CNN – це процес навчання від початку до кінця, де вхідними даними є лише початкове зображення, навчання та прогнозування відбуваються в мережі, і результат виводиться на виході. Цей метод відмовляється від ручного виділення конкретних ознак зображення та руйнує проблему традиційних методів класифікації. Це також головна перевага CNN у класифікації зображень. Цей

розділ в основному представляє модель класифікації зображень на основі CNN та по черзі представляє представницькі класичні моделі за хронологією.

1.3 Вивчення найкращих практик розробки вебзастосунків для моди

Розвиток цифрових технологій має важливе значення також і для модної індустрії, особливо в контексті вебзастосунків, які служать як фундаментальний інструмент для приваблення клієнтів та підвищення продаж. Вебзастосунки для моди мають унікальні вимоги до дизайну, функціональності та користувацького досвіду. Вивчення і застосування найкращих практик у розробці дозволяють створювати вебзастосунки, які не тільки задовольняють основні потреби користувачів, але й підкреслюють унікальність бренду. На рисунку 1.5 представлено діаграму, яка ілюструє найкращі практики розробки вебзастосунків для моди.

Рисунок 1.5 – Діаграма поширених практик розробки вебзастосунків

Розробка вебзастосунків для модної індустрії вимагає особливого підходу, який залучає увагу користувачів та забезпечує ефективну взаємодію найкращих практик у цій сфері:

– адаптивний дизайн. Адаптивний дизайн забезпечує, що вебзастосунок виглядає добре та функціонує коректно на всіх типах пристроїв – від смартфонів до десктопів. Це критично важливо для модної індустрії, де багато покупок здійснюються через мобільні пристрої;

– візуальний сторітеллінг. Модні бренди часто використовують візуальний сторітеллінг для показу своїх колекцій та акцій. Використання якісних зображень, відео та анімацій може значно підвищити залученість користувачів та підкреслити естетику бренду;

– швидкість завантаження. Оптимізація швидкості завантаження сторінок є важливою для утримання користувачів, особливо в мобільному середовищі. Це можна досягти за допомогою стиснення зображень, мініфікації коду та використання технологій кешування.

– інтерактивні елементи. Інтеграція інтерактивних елементів, таких як віртуальні примірочні, кастомізація продуктів або інтерактивні покази мод, може значно покращити користувацький досвід та залученість;

– персоналізація. Використання даних про користувача для персоналізації контенту та пропозицій може значно підвищити ефективність вебзастосунку. Це може включати рекомендації продуктів, персональні акції, та адаптований контент;

– оптимізація для пошукових систем (SEO). Для модних брендів важливо займати високі позиції у пошукових системах. Оптимізація контенту, структури сторінок та метаданих для пошукових систем може допомогти залучити більше органічного трафіку;

– безпека. Забезпечення безпеки транзакцій та захисту даних користувачів є важливим для будь-яких вебзастосунків. Це особливо важливо для інтернет-магазинів, які обробляють особисті та платіжні дані;

– прогресивні вебзастосунки (PWAs). Прогресивні вебзастосунки надають користувачам досвід, схожий на роботу з нативними мобільними програмами, але без необхідності завантаження з App Store або Google Play. Вони можуть працювати офлайн та надсилати сповіщення, що робить їх ідеальними для модних ритейлерів. Застосування цих практик в розробці вебзастосунків може допомогти модним брендам не тільки збільшити продажі та покращити залученість користувачів, але й створити більш стійкий і надійний цифровий продукт.

1.4 Постановка задачі

В результаті даної роботи буде створено вебзастосунок, що дозволяє виконувати автоматизоване розпізнавання одягу на фотографіях за допомогою сучасних технологій глибокого навчання, використовуючи моделі yolov8m для виявлення об’єктів та clothesClassificationbest для класифікації типів одягу.

Об’єктом дослідження є зображення одягу, отримані з різних джерел, що включають соціальні мережі, вебсайти рітейлерів та користувацькі фото.

Метою роботи є розробка вебзастосунку, що використовує нейронні мережі для класифікації одягу за зображеннями, що надається користувачами. Використано сучасні архітектури нейронних мереж, такі як YOLOv8, для виявлення та класифікації об’єктів на зображеннях. Проведено аналіз існуючих рішень у цій області, вивчено кращі практики розробки вебзастосунків для моди, та розроблено методи навчання та прогнозування, адаптовані для високої точності та швидкості обробки. Для досягнення цієї мети необхідно вирішити наступні завдання:

– провести аналіз існуючих нейронних мереж для виявлення та класифікації об’єктів на зображеннях;

– розробити алгоритм для детектування одягу використовуючи модель yolov8m;

– реалізувати алгоритм класифікації одягу з використанням моделі clothesClassificationbest;

– інтегрувати обидва алгоритми в єдиний вебзастосунок, забезпечуючи високу точність та швидкість обробки зображень.

– провести тестування системи для оцінки її ефективності та точності.

Ці завдання будуть вирішені через розробку комплексного рішення, яке інтегрує передові алгоритми машинного навчання та комп’ютерного зору, щоб створити інтерфейс, який відповідає вимогам сучасного ринку електронної комерції.

2 НЕЙРОННІ МЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ОДЯГУ

2.1 Вибір архітектури нейронної мережі

Для вирішення задачі класифікації одягу в рамках вебзастосунку необхідно обрати архітектуру нейронної мережі, яка оптимально поєднує високу точність розпізнавання з ефективним використанням обчислювальних ресурсів. Застосування нейронних мереж у вебзастосунках ставить додаткові вимоги до швидкості обробки даних і мінімізації затримок, що важливо для забезпечення зручності кінцевих користувачів.

Вибір архітектури нейронної мережі залежить від багатьох факторів, включаючи обсяг та різноманітність об’єму даних, необхідність роботи з деталізованими характеристиками одягу, а також обмеження, накладені технічними ресурсами вебсервера. Наступні архітектури були вибрані з урахуванням цих аспектів і відомі своєю здатністю ефективно вирішувати подібні задачі: YOLOv8, MobileNetV3 та EfficientNetB4.

YOLOv8 [27], випущений у січні 2023 року компанією Ultralytics, являє собою новітнє продовження знаменитої серії архітектур YOLO (You Only Look Once). YOLO, відомий своєю здатністю виконувати швидке і точне виявлення об’єктів в реальному часі, був значно вдосконалений у своїх попередніх версіях, починаючи від YOLOv1 і до YOLOv5, також розробленого Ultralytics.

OLOv8 став результатом глибоких досліджень і розробок у галузі комп’ютерного зору. Нововведення YOLOv8 полягає у впровадженні п'яти масштабованих версій, які дозволяють точно підібрати потрібну модель залежно від потреб у точності та обчислювальних ресурсах. Ці версії включають OLOv8n (нано), YOLOv8s (малий), YOLOv8m (середній), YOLOv8l (великий) і YOLOv8x (дуже великий). YOLOv8 підтримує широкий спектр завдань у галузі комп’ютерного зору, таких як: виявлення об’єктів, сегментація, оцінка пози, відстеження та класифікація.

На рисунку 2.1 показана архітектура YOLOv8. YOLOv8 використовує схожу базову структуру, що й YOLOv5 з деякими змінами у CSPLayer, який тепер називається модулем C2f. Модуль C2f (перехресне ступеневе часткове обмеження з двома згортками) поєднує високорівневі особливості з контекстуальною інформацією для покращення точності виявлення.

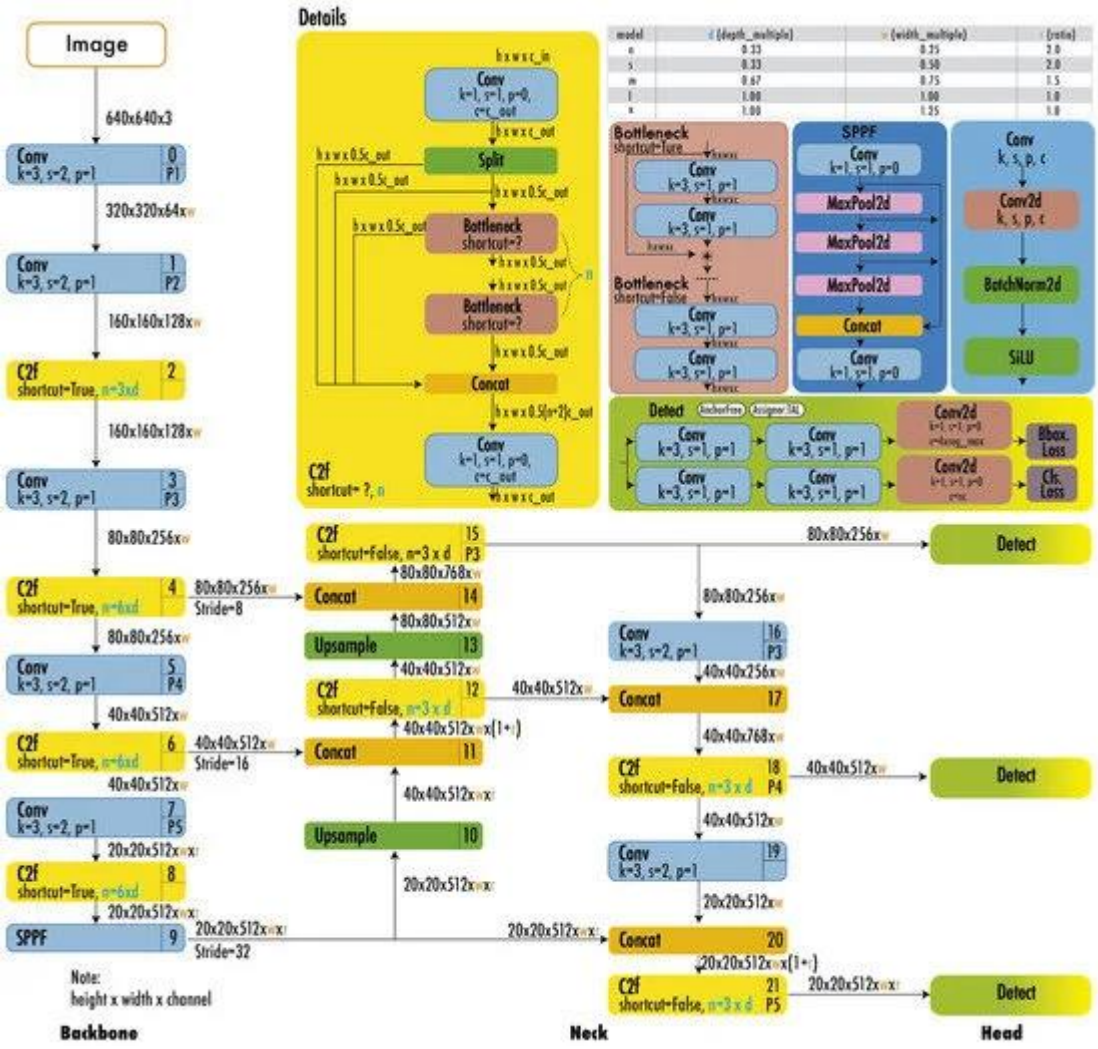

Рисунок 2.1 – Архітектура YOLOv8m [28]

YOLOv8 є передовою архітектурою нейронної мережі, призначеною для задач виявлення об’єктів. Ця архітектура використовує модифіковану версію CSPDarknet53 як основу (backbone), яка забезпечує потужне і ефективне виявлення об’єктів за мінімального часу обробки.

CSPDarknet53 – це мережа, яка використовується в YOLOv4 та покращена для YOLOv8. Основна особливість цієї мережі полягає в застосуванні Cross-Stage Partial (CSP) структур, які розділяють потік характеристик на дві частини, що дозволяє зменшити кількість операцій і поліпшити ефективність навчання за рахунок зменшення завантаження на пам’ять.

У версії YOLOv8 використовується новий модуль C2f, який замінює традиційний CSPLayer, застосований у YOLOv5. C2f модуль оптимізований для кращої передачі інформації між шарами мережі, що забезпечує більшу точність та швидкість обробки.

Для прискорення обчислень в YOLOv8 впроваджено шар Spatial Pyramid Pooling Fast (SPPF). Цей шар використовується для агрегації характеристик з різних регіонів вхідного зображення, створюючи фіксовану карту розмірів, незалежно від їхнього первісного розміру. Це дозволяє зберігати просторову ієрархію характеристик та поліпшує точність виявлення об’єктів на зображеннях різних розмірів. Кожен конволюційний шар у YOLOv8 має нормалізацію за партіями та активацію SiLU. Нормалізація за партіями допомагає стабілізувати навчання, нормалізуючи розподіл вхідних даних для кожного шару, тоді як активація SiLU (Sigmoid-weighted Linear Unit) забезпечує нелінійність, необхідну для виявлення складних патернів у даних.

Головний блок у YOLOv8 роз’єднаний, що означає незалежну обробку завдань визначення наявності об’єкту, класифікації та регресії. Це дозволяє спеціалізувати частини мережі на конкретні завдання, що підвищує загальну ефективність та точність системи.

Завдяки вдосконаленням у кожному аспекті архітектури, від спінової мережі до кінцевих шарів обробки, YOLOv8 забезпечує високу точність і швидкість, що робить її ідеальною для застосування у різноманітних областях, включаючи, але не обмежуючись, вебзастосунками для класифікації одягу.

До переваг YOLOv8 відносять:

– висока швидкість обробки. YOLOv8 продовжує традиції серії YOLO забезпечувати швидку обробку даних, що дозволяє виконувати виявлення об’єктів у реальному часі. Ця особливість є критичною для застосунків, де швидкість реакції системи є важливою, таких як відеонагляд та автономне керування транспортними засобами;

– покращена точність. Завдяки вдосконаленням у архітектурі, таким як модуль C2f та використання SiLU активації, YOLOv8 показує вищу точність у виявленні об’єктів порівняно з попередніми версіями, що забезпечує краще розпізнавання складних об’єктів та сцен;

– гнучкість у виборі моделі. Архітектура YOLOv8 пропонує декілька варіантів моделей з різними розмірами (від нано до дуже великого), що дозволяє користувачам оптимально вибирати модель залежно від вимог до точності та обчислювальних ресурсів;

– підтримка різноманітних завдань комп’ютерного зору. YOLOv8 не обмежений лише виявленням об’єктів, а також підтримує сегментацію, оцінку пози, відстеження та класифікацію, роблячи його універсальним інструментом для широкого спектру застосунків у галузі комп’ютерного зору.

Недоліки YOLOv8 складаються із:

– високі вимоги до обчислювальних ресурсів. Більші моделі YOLOv8, такі як YOLOv8l та YOLOv8x, потребують значної обчислювальної потужності, що може бути обтяжливим для обмежених систем або пристроїв з низькою продуктивністю;

– складність тренування. Через велику кількість параметрів та складності архітектури, тренування моделі YOLOv8 може вимагати значних обчислювальних ресурсів та часу, особливо при роботі з великими наборами даних;

– можливість перенавчання на специфічних об’ємах даних. Як і багато інших складних моделей, YOLOv8 схильний до перенавчання, особливо коли використовується з обмеженими або надмірно специфічними

об’ємами даних, що може призвести до зниження загальної здатності узагальнення моделі;

– відносна новизна. Будучи відносно новою архітектурою, YOLOv8 має менше спільноти та менш вивчені випадки застосування порівняно з більш стабілізованими архітектурами, що може вплинути на швидкість прийняття та інтеграцію у великі проєкти.

Отже, YOLOv8 є потужною нейронною мережею, яка вносить значні вдосконалення у швидкість обробки і точність розпізнавання, що робить її цінним інструментом у галузі комп’ютерного зору. Ця архітектура відрізняється своєю гнучкістю, оскільки пропонує кілька варіантів моделей для різних обчислювальних потреб та завдань. Окрім виявлення об’єктів, YOLOv8 підтримує сегментацію, оцінку пози, відстеження та класифікацію, забезпечуючи широкий спектр потенційних застосувань. Проте, важливо зазначити, що використання YOLOv8 може бути обтяжливим для систем з обмеженими обчислювальними ресурсами, особливо при використанні більших моделей. Складність тренування та потенційне перенавчання також можуть бути викликами для розробників.

Загалом, YOLOv8 пропонує передові можливості у вирішенні завдань комп’ютерного зору, але потребує ретельного вибору моделі та підходу до тренування для оптимального використання в конкретних проєктах та застосунках.

У 2017 році Google представив MobileNets, сімейство моделей комп’ютерного зору на основі TensorFlow [29]. Нова архітектура MobileNets була представлена кілька днів тому і містить кілька цікавих ідей для покращення мобільних моделей комп’ютерного зору.

MobileNetV3 – це третя версія архітектури, яка використовується для аналізу зображень в багатьох популярних мобільних застосунках. Цю архітектуру також включено в популярні фреймворки, такі як TensorFlow Lite. MobileNets повинні ретельно збалансувати прогреси у комп’ютерному зорі та глибокому навчанні загалом із обмеженнями мобільних середовищ. Google

регулярно випускає оновлення архітектури MobileNets, в яких впроваджуються найновітніші ідеї в галузі глибокого навчання.

Останні вдосконалення архітектури MobileNets були підсумовані в науковій статті [30], опублікованій у серпні 2019 року. Основний внесок MobileNetV3 полягає у використанні AutoML для визначення найкращої можливої архітектури нейронної мережі для даної задачі. Це контрастує з ручним проєктуванням попередніх версій архітектури. Зокрема, MobileNetV3 використовує дві техніки AutoML: MnasNet та NetAdapt. Спочатку MobileNetV3 шукає грубу архітектуру за допомогою MnasNet, який використовує навчання з підкріпленням для вибору оптимальної конфігурації з дискретного набору варіантів. Після цього модель доопрацьовує архітектуру за допомогою NetAdapt, доповнюючої техніки, яка обрізає недостатньо використовувані канали активації невеликими інкрементами.

Однією з нововведень у MobileNetV3 є включення блоків стискання та виклику у основну архітектуру [31], що сприяє значному підвищенню ефективності та точності моделі. Блоки стискання та виклику були вперше представлені у архітектурі MobileNetV3 з метою покращення якості репрезентацій, створених мережею (рис. 2.2).

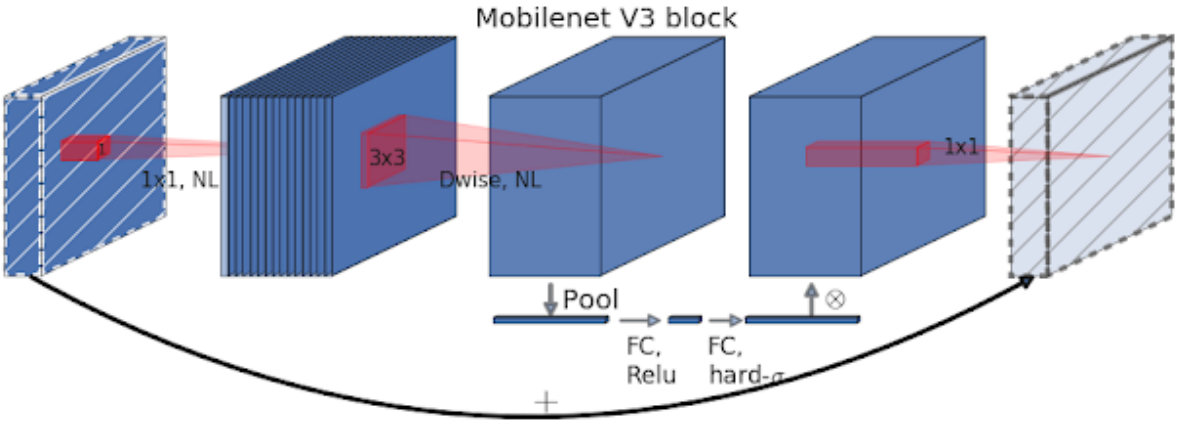

Рисунок 2.2 – Архітектура YOLOv8m [29]

Ці блоки функціонують на принципі моделювання взаємозалежності між каналами згорткових характеристик. Завдяки цьому механізму мережа

може виконувати ре-калібрування ознак, тобто адаптувати активність каналів залежно від того, наскільки інформативні вони для конкретного завдання або вхідного зразка

Цікавою оптимізацією MobileNetV3 стала переробка деяких витратних шарів в архітектурі. Деякі з шарів в MobileNetV2 були фундаментальними для точності моделей, але також вносили певний рівень затримки. Шляхом включення деяких базових оптимізацій MobileNetV3 вдалося вилучити три витратні шари з попередньої архітектури, не жертвуючи точністю.

MobileNetV3 показав значні покращення порівняно з попередніми архітектурами. Наприклад, у завданнях виявлення об'єктів MobileNetV3 працював з 25% меншою затримкою та такою ж точністю, як і попередні версії. Подібні поліпшення спостерігалися в завданнях класифікації, як показано на наступному рисунку:

MobileNets залишаються однією з найбільш передових архітектур у мобільному комп'ютерному зорі. Включення AutoML в MobileNetV3 безперечно відкриває двері до різноманітних цікавих архітектур. Найновіший реліз MobileNets доступний на GitHub, а реалізація MobileNetV3 включена в API виявлення об'єктів Tensorflow.

Переваги MobileNetV3:

– енергоефективність та оптимізація під мобільні пристрої. MobileNetV3 спроєктована з урахуванням ефективності, що робить її ідеальною для застосування в мобільних пристроях і пристроях з обмеженими обчислювальними ресурсами. Це досягається завдяки використанню легких згорткових блоків та оптимізованих шарів активації;

– використання AutoML для оптимізації архітектури. Застосування методів автоматизованого машинного навчання (AutoML), таких як MnasNet і NetAdapt, дозволяє MobileNetV3 знаходити оптимальні структури для конкретних завдань, забезпечуючи високу точність при мінімальному споживанні ресурсів;

– підвищена точність та адаптивність завдяки блокам стискання та виклику. Блоки стискання та виклику дозволяють мережі більш точно моделювати взаємозалежності між каналами, що забезпечує вищу якість репрезентацій. Це підвищує точність і адаптивність моделі до різних видів даних;

– швидкість виконання. MobileNetV3 розроблена для швидкої роботи, що критично важливо для реального часу застосунків, таких як обробка зображень або відео на мобільних пристроях.

Недоліки MobileNetV3:

– обмеження на складність завдань. Попри значні оптимізації, MobileNetV3 може мати обмеження при роботі з дуже складними або ресурсоємними завданнями, які вимагають великої обчислювальної потужності;

– потенційна втрата інформації у спрощених моделях. Оптимізація для мінімізації ресурсів може призвести до втрати деяких важливих інформаційних сигналів у процесі стискання та виклику, особливо у випадках з високою різноманітністю даних;

– залежність від якості тренувальних даних. Ефективність моделі значно залежить від якості та репрезентативності даних, використаних під час тренування. Недостатньо різноманітний або обмежений набір даних може знизити загальну здатність моделі до узагальнення;

– складність інтеграції та оптимізації. Незважаючи на автоматизацію деяких процесів налаштування через AutoML, додаткова оптимізація архітектури під специфічні застосунки може вимагати глибоких знань у галузі машинного навчання та системного програмування.

Отже, MobileNetV3 є значним внеском у розвиток невеликих архітектур нейронних мереж, оптимізованих для мобільних пристроїв і пристроїв з обмеженими обчислювальними ресурсами. Ця архітектура втілює інноваційні підходи, такі як застосування AutoML для оптимізації структури мережі та впровадження блоків стискання та виклику, які значно підвищують

ефективність і точність обробки даних. Також ця мережа демонструє вражаючу швидкість обробки, що критично для застосунків, які вимагають роботи в реальному часі.

Однак, певні недоліки, такі як обмеження на складність завдань та потенційна втрата інформації у спрощених моделях, вказують на необхідність обережного вибору варіанту використання MobileNetV3, залежно від конкретних вимог до точності та обчислювальних ресурсів. Все ж, MobileNetV3 залишається відмінним вибором для розробників, які шукають ефективні та адаптивні рішення у сфері мобільного машинного навчання.

Представлений у 2019 році командою дослідників у Google AI, EfficientNet став однією з основних архітектур для багатьох складних завдань, включаючи визначення об’єктів, сегментацію зображень, а навіть обробку мови [32]. Його успіх полягає у здатності балансувати два критичних фактори у глибокому навчанні: обчислювальну ефективність та продуктивність моделі.

Традиційні моделі глибокого навчання часто мають компроміс між точністю та споживанням ресурсів. EfficientNet вирішує цей виклик, впроваджуючи новий підхід, званий «комплексне масштабування».

Систематичне масштабування розмірів моделі (ширина, глибина і роздільна здатність) за принциповими методами дозволяє EfficientNet досягати безпрецедентних рівнів ефективності, не жертвуючи точністю. Цей метод дозволяє моделі забезпечити оптимальний баланс, роблячи її адаптивною до різних обчислювальних бюджетів та можливостей обладнання.

EfficientNet використовує шари Mobile Inverted Bottleneck (MBConv), які є поєднанням згорток з роздільними по глибині та оберненими залишковими блоками (рис. 2.3). Крім того, архітектура моделі використовує оптимізацію Squeeze-and-Excitation (SE), щоб ще більше покращити продуктивність моделі.

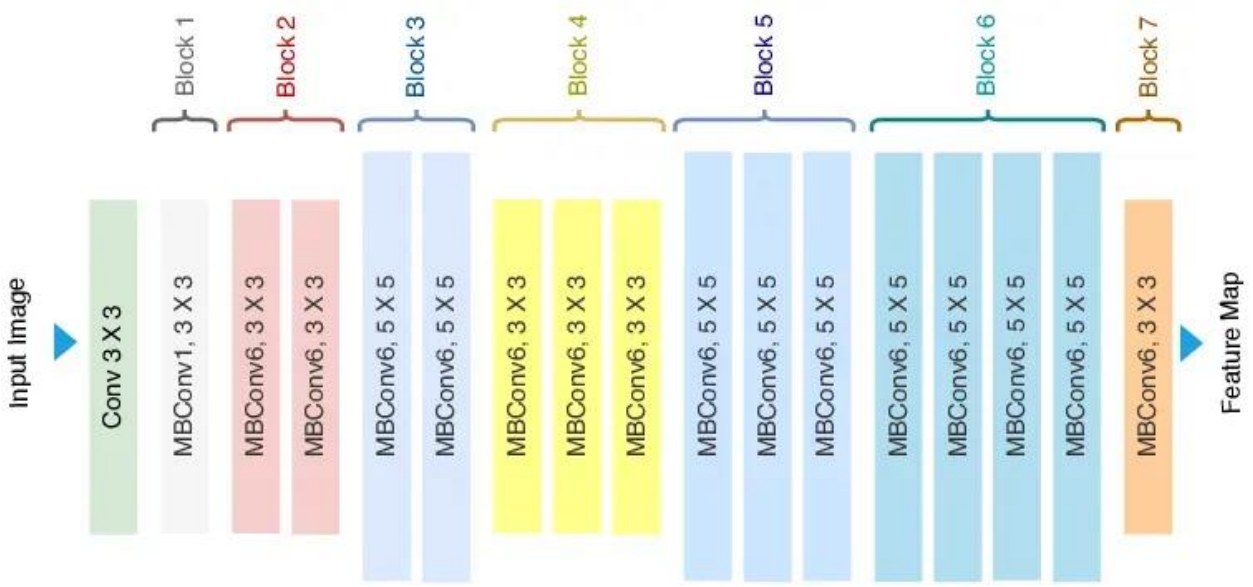

Рисунок 2.3 – Архітектура EfficientNet [33]

Шар MBConv є основним будівельним блоком у структурі EfficientNet. Він взяв ідею з обернених залишкових блоків у MobileNetV2, але зазнав деяких змін і покращень.

Шар MBConv починається з згортки з роздільним по глибині згортанням, за якою слідує згортка з точковим згортанням (згортка 1x1), яка розширює кількість каналів, і, нарешті, ще одна згортка 1x1, яка зменшує кількість каналів до початкової кількості. Це конструктивне обмеження дозволяє моделі ефективно навчатися, при цьому зберігаючи високий рівень представлення.

Крім шарів MBConv, EfficientNet включає блок SE, який допомагає моделі навчатися фокусуватися на важливих ознаках та пригнічувати менш значущі. Блок SE використовує глобальне середнє пулінгування для зменшення просторових розмірів карти ознак до одного каналу, за яким слідують два повністю зв’язаних шари.

Ці шари дозволяють моделі навчатися залежностям функцій за каналами та створювати ваги уваги, які множаться на початкову карту ознак, підкреслюючи важливу інформацію.

EfficientNet має різні варіанти, такі як EfficientNet-B0, EfficientNet-B1 і так далі, з різними коефіцієнтами масштабування. Кожен варіант представляє

різний компроміс між розміром моделі та точністю, що дозволяє користувачам вибирати відповідний варіант моделі залежно від їх конкретних вимог.

Переваги EfficientNet:

– масштабування моделі. EfficientNet використовує новаторський підхід до масштабування моделей, який включає балансування ширини мережі, її глибини та роздільної здатності вхідних зображень. Це дозволяє досягти вищої точності без непропорційного збільшення кількості параметрів;

– висока точність та ефективність. Завдяки оптимізованому масштабуванню, EfficientNet досягає значно кращих показників точності порівняно з іншими моделями при аналогічному або навіть меншому обчислювальному навантаженні, роблячи її ефективним вибором для широкого спектру застосунків;

– економічність використання ресурсів. За рахунок більш раціонального масштабування, EfficientNet використовує менше обчислювальних ресурсів порівняно з іншими моделями зі схожою точністю. Це робить її ідеальною для застосунків, де важливі як висока продуктивність, так і обмеженість ресурсів;

– гнучкість застосування. EfficientNet може бути ефективно адаптована до різних задач та умов, включаючи роботу на обмежених пристроях, таких як мобільні телефони, і в складних системах з обробкою зображень великої роздільної здатності.

Недоліки EfficientNet:

– складність тренування. Через високу кількість параметрів та складність архітектури, тренування EfficientNet може виявитись часомістким та вимагати значних обчислювальних ресурсів, особливо для більших версій моделі;

– залежність від великої кількості даних. Для досягнення високої точності EfficientNet потребує великих тренувальних наборів даних, що може бути обмежуючим фактором у сценаріях з обмеженою кількістю даних;

– вразливість до перенавчання. Як і багато складних моделей, EfficientNet може бути схильна до перенавчання, особливо при використанні обмежених об’ємів даних, що може знизити її здатність до узагальнення на нових даних;

– потреба у налаштуванні гіперпараметрів. Хоча EfficientNet і є адаптивною до різних задач, оптимальне налаштування гіперпараметрів може виявитись складним і вимагати додаткових зусиль для досягнення найкращої можливої продуктивності на конкретній задачі або даних.

Отже, EfficientNet представляє собою передову архітектуру нейронної мережі, яка забезпечує високу ефективність і точність завдяки інноваційному підходу до масштабування моделей. Ця архітектура вирізняється балансуванням ширини, глибини та роздільної здатності, що дозволяє їй досягати вражаючих результатів навіть на обмежених обчислювальних ресурсах. EfficientNet ефективно використовує свої ресурси, роблячи її придатною для широкого спектру застосунків, від мобільних пристроїв до складних систем обробки зображень.

Однак, певні складнощі, такі як вимогливість до обчислювальних ресурсів при тренуванні більших версій, залежність від великих обсягів даних для тренування та потенційне перенавчання, потребують уважного розгляду при виборі цієї архітектури для конкретних застосунків. Незважаючи на ці виклики, EfficientNet залишається однією з найбільш ефективних нейронних мереж для створення потужних і масштабованих рішень у галузі машинного навчання.

У таблиці 2.1 представлено порівняльний аналіз трьох популярних архітектур нейронних мереж: YOLOv8, MobileNetV3 та EfficientNetB4. Цей аналіз дозволяє ідентифікувати ключові характеристики кожної з архітектур, де важлива висока швидкість обробки, точність виявлення об’єктів, масштабованість архітектури та універсальність в різних задачах комп’ютерного зору. Порівняння допомагає оцінити потенційну ефективність

кожної моделі для специфічних сценаріїв використання, зокрема для класифікації одягу у вебзастосунку.

Таблиця 2.1 – Порівняльний аналіз нейронних мереж

Характеристика	YOLOv8	MobileNetV3	EfficientNetB4
Швидкість виявлення об’єктів	Висока, реальний час	Середня	Низька
Точність виявлення об’єктів	Дуже висока	Висока в мобільних застосунках	Висока в класифікації зображень
Масштабованість архітектури	Доступні різні розміри моделей	Обмежена масштабованість	Висока масштабованість
Підтримка різних задач CV	Виявлення, сегментація, класифікація, оцінка пози, відстеження	Основно виявлення і класифікація	Переважно класифікація зображень
Використання в реальних умовах	Для відеонагляду та автономних систем	Для мобільних пристроїв	Для серверних застосунків
Вимоги до обчислювальних ресурсів	Високі для великих моделей	Низькі, оптимізовані для мобільних пристроїв	Середні, залежно від розміру моделі

Вибір YOLOv8 для класифікації одягу у вебзастосунку можна обґрунтувати кількома ключовими перевагами цієї архітектури порівняно з іншими популярними моделями, такими як MobileNetV3 та EfficientNetB4. По-перше, YOLOv8 відрізняється своєю високою швидкістю обробки даних, що є критично важливим для вебзастосунків, де користувачі очікують миттєвої відповіді. Завдяки оптимізації для роботи в реальному часі, YOLOv8 може швидко класифікувати одяг на зображеннях, що надсилаються користувачами, мінімізуючи затримки та покращуючи користувацький досвід. Крім того, YOLOv8 демонструє виняткову точність у виявленні об’єктів, що є особливо важливим для класифікації одягу, де потрібно точно ідентифікувати різні

стилі, кольори та особливості дизайну. Ця модель здатна ефективно розпізнавати складні візуальні патерни та текстури, що робить її ідеальною для задач, де необхідно детальне розуміння змісту зображення. Крім того, масштабованість архітектури YOLOv8 забезпечує гнучкість у виборі конфігурації моделі, що дозволяє адаптувати рівень деталізації та обчислювальні вимоги до специфіки вебзастосунку. Можливість вибору з декількох варіантів (від YOLOv8n до YOLOv8x) дозволяє оптимізувати споживання ресурсів без втрати продуктивності.

Також YOLOv8 підтримує різноманітні задачі комп'ютерного зору, включаючи класифікацію, виявлення об'єктів, сегментацію та оцінку поз, що робить її універсальним рішенням для розширених вебзастосунків. Ця багатофункціональність забезпечує високу адаптивність моделі до різних вимог та сценаріїв використання.

2.2 Розробка методу для класифікації

Архітектура YOLO використовує згорткові нейронні мережі, які ефективно обробляють зображення завдяки своїй здатності вловлювати просторові ієрархії характеристик. YOLOv8, зокрема, реалізує покращення у вигляді оптимізації архітектури та збільшення швидкості і точності, що є критичним для застосунків реального часу.

Алгоритм навчання YOLOv8 заснований на концепції «regression problem», де мережа намагається безпосередньо прогнозувати координати об'єктів та їх класифікації на одному етапі обробки. Це значно спрощує процес і знижує затримки, порівняно з традиційними методами, які вимагають кількох послідовних кроків для виявлення та класифікації об'єктів.

На рисунку 2.4 представлено алгоритм навчання нейронної мережі YOLOv8.

Рисунок 2.4 – Алгоритм навчання

Алгоритм навчання YOLOv8 працює наступним чином:

Крок 1. Поділ зображення на сітку. Алгоритм YOLOv8 розпочинає з поділу вхідного зображення на S×S сітку (парціальних блоків). Кожна клітинка сітки відповідає за виявлення об’єктів, які мають свій центр у цій клітинці. Це дозволяє локалізувати та класифікувати кілька об’єктів в одному зображенні ефективно.

Крок 2. Прогнозування обмежувальних рамок. Кожна клітинка сітки прогнозує певну кількість обмежувальних рамок (bounding boxes) і впевненість, що відображає імовірність наявності об’єкта в рамці та точність цієї рамки. Кожна рамка описується чотирма координатами (центр *x*, центр *y*, ширина, висота) та значенням впевненості.

Крок 3. Визначення класів. Крім координат і впевненості рамки, кожна клітинка також прогнозує ймовірності класів для об’єкта у кожній рамці. Це дозволяє алгоритму не лише локалізувати об’єкти, але й класифікувати їх серед певного числа визначених класів.

Крок 4. Функція втрат. YOLOv8 використовує комбіновану функцію втрат, яка мінімізує помилки у координатах обмежувальних рамок, впевненості та класифікації. Ця функція втрат включає:

– квадратичну помилку для координат центру та розмірів рамок;

– logistic regression loss для впевненості, яка підкреслює різницю між наявністю та відсутністю об’єктів;

– cross-entropy loss для помилок класифікації об’єктів.

Крок 5. Не максимальне придушення (non-max suppression). Після виявлення об’єктів алгоритм використовує метод не максимального придушення для видалення дублюючих рамок (рис. 2.4). Цей процес забезпечує, що кожен об’єкт ідентифікується однією, найкращою рамкою.

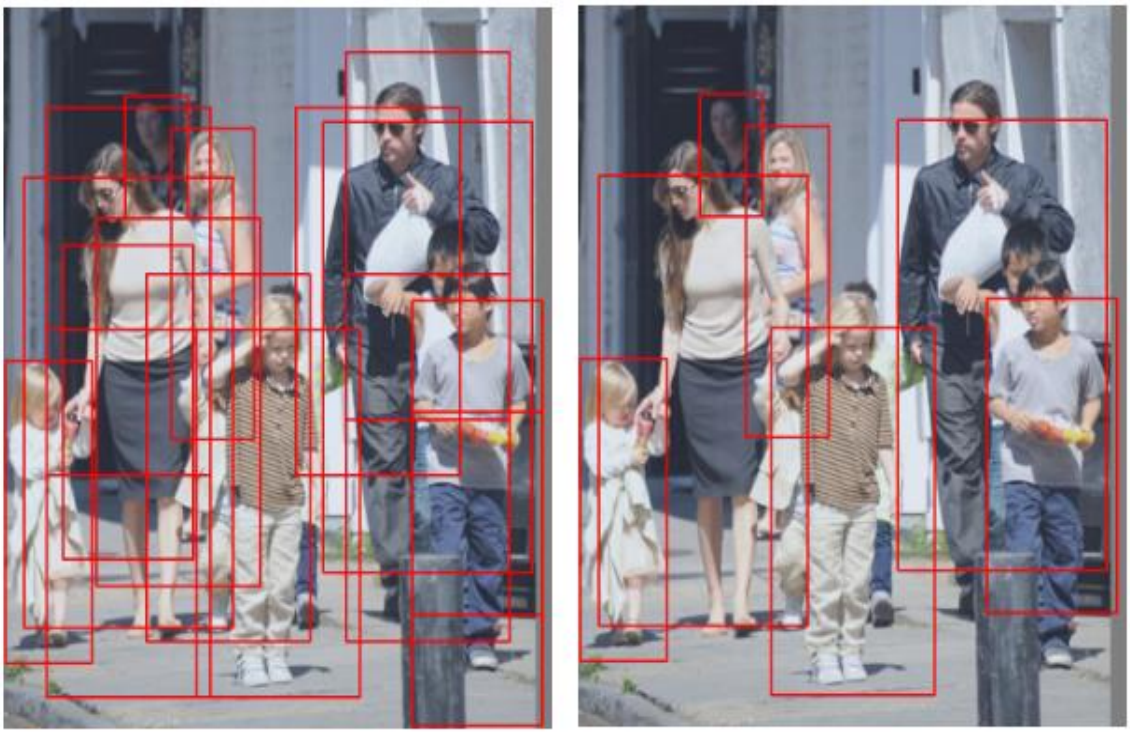

Рисунок 2.4 – Приклад видалення дублюючих рамок [34]

Алгоритм розпізнавання та класифікації одягу за допомогою YOLOv8, включає кілька ключових кроків, що оптимізують процес для швидкого та ефективного виконання (рис. 2.5).

Рисунок 2.5 – Алгоритм розпізнавання та класифікації одягу

Алгоритм YOLOv8 розпізнавання та класифікації одягу працює наступним чином:

Крок 1. Попередня обробка зображення. Перед подачею зображення в нейронну мережу, зображення потрібно підготувати та нормалізувати. Це включає масштабування розмірів зображення до вхідного розміру, який приймає модель (наприклад, 416x416 пікселів для стандартних конфігурацій YOLO), та нормалізацію кольорових каналів шляхом віднімання середніх значень та поділу на стандартне відхилення.

Крок 2. Поділ зображення на сітку. YOLO розділяє зображення на сітку фіксованого розміру (наприклад, 13x13 клітин). Кожна клітинка сітки відповідає за виявлення об’єктів, чий центр знаходиться у даній клітинці.

Крок 3. Виявлення об’єктів і прогнозування обмежувальних рамок. Кожна клітинка сітки вихідного зображення прогнозує кілька обмежувальних рамок. Для кожної рамки прогнозуються такі параметри:

– координати центру рамки (x, y);

– ширина та висота рамки;

– впевненість, що об’єкт існує в рамці;

– умовні ймовірності класу для кожної рамки.

Крок 4. Фільтрація рамок за допомогою порогу впевненості. Рамки, які мають впевненість нижче певного порогу (наприклад, 0.5), відкидаються, щоб зменшити кількість хибних виявлень.

Крок 5. Застосування не максимального придушення. Цей метод допомагає видалити перекриваючі рамки, що вказують на один і той самий об’єкт. NMS залишає лише рамку з найвищою впевненістю, придушуючи інші, що значно перекриваються з нею.

Крок 6. Класифікація одягу. На заключному етапі, для кожної рамки, яка залишилася після NMS, визначається клас об’єкта з найвищою умовною ймовірністю. Класи можуть включати різні типи одягу, такі як сорочки, штани, сукні тощо.

У вебзастосунках YOLO може бути інтегрована на сервері або використовуватись у клієнтській частині за допомогою різноманітних бібліотек, таких як TensorFlow та ін.. Користувачі можуть завантажувати зображення через вебінтерфейс, після чого система обробляє зображення та відображає результати класифікації в реальному часі.

3 РОЗРОБКА ВЕБЗАСТОСУНКУ

3.1 Вибір технологічного стеку для вебзастосунку

Визначення технологічної основи системи є ключовим етапом, який впливає на архітектуру проєкту, його продуктивність та ефективність обслуговування. Під час вибору компонентів технологічного стеку необхідно врахувати кілька важливих аспектів: вимоги до функціональності та масштабованості системи, забезпечення безпеки даних та фінансові обмеження проєкту. Технології мають не тільки відповідати початковим потребам проєкту, але й дозволяти системі гнучко адаптуватися до змін і новацій у майбутньому. З огляду на динамічність технологічного світу, важливо вибрати такі рішення, які забезпечать довготривалу підтримку і легку інтеграцію з іншими системами та сервісами, а також зможуть витримати зростання обсягів даних та користувачів без втрати продуктивності.

ASP.NET Core MVC – це високопродуктивний фреймворк для розробки вебзастосунків, що базується на архітектурному шаблоні Model-View-Controller (MVC). Він надає розробникам потужні інструменти для створення сучасних та масштабованих вебзастосунків [35]. Завдяки розділенню логіки за допомогою контролерів, представлень та моделей, ASP.NET Core MVC сприяє покращенню організації коду та забезпечує простоту розширення проєктів. Цей фреймворк також підтримує вбудовану інтеграцію з різноманітними технологіями, такими як Entity Framework Core, що робить його ідеальним вибором для різних типів проєктів.

Node.js – це зручна та потужна платформа для розробки серверних програм на JavaScript. Вона ґрунтується на JavaScript runtime Chrome V8 і дозволяє розробникам створювати швидкі та масштабовані застосунки [36]. Завдяки асинхронному програмуванню та подіям, Node.js дозволяє забезпечити ефективне використання ресурсів сервера. Ця платформа також

підтримує велику кількість розширень (пакетів), що дозволяє розробникам легко розширювати функціональність своїх програм.

Python – це високорівнева мова програмування, яка відома своєю простотою та читабельністю коду [37]. Вона широко використовується для розробки різноманітних програм, від вебзастосунків до штучного інтелекту. Завдяки великій кількості бібліотек та фреймворків, таких як Django та Flask, Python є популярним вибором для швидкого та ефективного створення програмних рішень. Крім того, Python має активну спільноту розробників, яка постійно розширює його функціональність та підтримує нові технології.

У таблиці 3.1 приведений аналіз функціональних можливостей кожної із розглянутих вебтехнологій.

Таблиця 3.1 – Порівняльний аналіз технологій

Характеристика	ASP.NET Core MVC	Node.js	Python
Продуктивність	Висока завдяки оптимізованому коду та вбудованій підтримці асинхронності	Залежить від використання асинхронного коду, може бути менш ефективним на CPU-інтенсивних задачах	Зазвичай нижча через динамічну типізацію та інтерпретацію
Безпека	Сильні вбудовані функції безпеки, такі як автентифікація, авторизація, захист від XSS та CSRF	Менше вбудованих засобів безпеки, залежить від сторонніх пакетів	Залежить від використовуваних фреймворків, потребує додаткової конфігурації
Масштабованість	Підтримка високого рівня масштабованості, легко інтегрується	Добре масштабується на великій кількості одночасних з’єднань, але може мати проблеми з потужністю обчислень	Масштабування можливе, але може потребувати додаткових зусиль з оптимізації

ASP.NET Core MVC для розробки вебзастосунку класифікації одягу забезпечує високу продуктивність і безпеку, які є критичними для обробки великих обсягів даних та зображень. Він також відомий своїми вбудованими засобами безпеки та ефективним управлінням ресурсами, що дозволяє легко масштабувати застосунок відповідно до зростаючих потреб користувачів. Фреймворк також відрізняється гнучкою інтеграцією з іншими системами та хмарними платформами, особливо з продуктами Microsoft, що полегшує розширення функціональності. Постійна підтримка від Microsoft та активна спільнота розробників гарантують доступність ресурсів для вирішення будь-яких технічних викликів.

Визначення відповідної системи управління базами даних (СУБД) є критичним аспектом для будь-якого проєкту, де необхідно здійснювати зберігання, обробку та аналіз великих обсягів даних. Кожна СУБД пропонує унікальні характеристики, переваги та потенційні обмеження, які мають бути вивчені з урахуванням потреб конкретного проєкту. MS SQL Server, MySQL та Oracle виступають як три популярні варіанти СУБД, що застосовуються для різних цілей, від розробки простих вебсайтів до впровадження складних корпоративних рішень.

MS SQL Server – це потужна та надійна реляційна система керування базами даних, розроблена корпорацією Microsoft. Вона використовується для зберігання, керування та обробки великих обсягів даних у корпоративних та підприємницьких середовищах [38]. MS SQL Server надає розширені можливості для виконання операцій з базами даних, включаючи транзакції, індексацію та резервне копіювання даних. Завдяки вбудованій підтримці для бізнес-аналітики та інтеграції з іншими продуктами Microsoft, MS SQL Server є популярним вибором для організацій, які потребують потужного та масштабованого рішення для управління даними.

MySQL – це відкрита реляційна система керування базами даних, яка широко використовується для зберігання та керування даними у вебзастосунка та інших програмних рішеннях. Вона відома своєю надійністю, швидкодією та

простотою у використанні. MySQL підтримує багато мов програмування та платформ, що робить її популярним вибором для різних типів проєктів. Завдяки великій спільноті розробників та активному розвитку, MySQL постійно оновлюється та покращується, щоб задовольнити потреби сучасних застосунків.

Oracle – це потужна корпорація, що спеціалізується на розробці програмного забезпечення, включаючи бази даних, хмарні рішення, програмне забезпечення для управління бізнесом та інші інструменти IT [40]. Серверні продукти Oracle, такі як Oracle Database, вважаються одними з найбільш потужних та масштабованих в галузі управління базами даних. Крім того, Oracle надає різноманітні послуги хмарного обчислення, що дозволяє організаціям швидко розгортати, масштабувати та управляти своїми програмами та сервісами. Завдяки широкому спектру продуктів та послуг, Oracle займає провідні позиції у світі корпоративних технологій та IT-рішень.

У таблиці 3.2 подано порівняльний аналіз цих СУБД, який сприятиме вирішенню проблеми вибору найбільш оптимальної системи для зберігання та обробки даних, враховуючи особливості проєкту та його вимоги.

Таблиця 3.2 – Порівняльний аналіз сховищ баз даних

Функція / Характеристика	MS SQL Server	MySQL	Oracle
Ліцензія	Комерційна	GPL або комерційна	Комерційна
Версії	Express, Standard, Enterprise	Community, Standard, Enterprise	Standard, Enterprise, Express
Платформи	Windows	Windows, Linux, macOS	Windows, Linux, Unix, macOS
Швидкість	Висока	Висока	Дуже висока
Масштабованість	Дуже висока	Висока	Дуже висока
Підтримка розподіленої роботи	+	+	+
Відкритість	–	+	–

Вибір MS SQL Server для розробки вебзастосунку для класифікації одягу можна виправдати його перевіреною продуктивністю та високим рівнем масштабованості, що є вирішальними для ефективної обробки і зберігання великих обсягів даних. MS SQL Server надає розширені можливості для обробки транзакцій та аналітики, що дозволяє реалізувати складні запити та отримання звітів у реальному часі, що є критично важливим для динамічного вебзастосунку. Комерційна підтримка від Microsoft гарантує стабільність, регулярні оновлення та безпеку, що робить MS SQL Server надійним вибором для бізнес-критичних застосунків. Завдяки тісній інтеграції з іншими продуктами Microsoft, особливо з ASP.NET, SQL Server забезпечує гладку інтеграцію та оптимізацію для вебзастосунків, що розробляються на Microsoft платформах.

3.2 Проєктування та розробка бази даних

При проєктуванні бази даних для вебзастосунку, призначеного для класифікації одягу, основним методом, який було обрано для створення структури даних, є метод «сутність-зв’язок» (Entity-Relationship, ER). Цей метод дозволяє наочно представити всі ключові компоненти даних та їхні взаємозв’язки, що є фундаментальним для забезпечення зрозумілості та систематизації інформації в базі даних. Використання ER-методу сприяє логічному проєктуванню бази даних, де кожна сутність визначається своїми унікальними атрибутами та зв’язками з іншими сутностями.

Важливо, що під час розробки логічної моделі бази даних було звернено особливу увагу на оптимізацію моделі для підтримки швидких запитів та обробки великих обсягів даних, що є ключовим для вебзастосунків реального часу одягу. Це забезпечує не лише ефективність роботи з даними, але й високу продуктивність самого вебзастосунку.

В рамках проєкту бази даних для було виділено наступні сутності:

Таблиця «AppUser» (табл. 3.3) створена для зберігання основної інформації про користувачів вебзастосунку. Структура цієї таблиці включає основні дані про користувачів, такі як нікнейм, ім’я, прізвище, пароль та ідентифікатор ролі.

Таблиця 3.3 – Структура таблиці «AppUser»

Назва поля	Тип даних	ПК	ЗК	Опис поля
nickname	varchar(50)	+	-	Унікальний нікнейм користувача, використовується як первинний ключ
name	varchar(50)	-	-	Ім’я користувача
surname	varchar(50)	-	-	Прізвище користувача
password	nchar(10)	-	-	Пароль користувача
roleId	int	-	-	Ідентифікатор ролі користувача, визначає права доступу в системі

Таблиця «UserRequest» (табл. 3.4) призначена для зберігання інформації про запити, які роблять користувачі, включаючи дані про об’єкти та кольори, виявлені на фотографіях. Вона дозволяє відслідковувати та аналізувати взаємодію користувачів з вебзастосунком.

Таблиця 3.4 – Структура таблиці «UserRequest»

Назва поля	Тип даних	ПК	ЗК	Опис поля
userRequestId	int	+	-	Унікальний ідентифікатор запиту користувача
colorsInPhoto	varchar(1000)	-	-	Інформація про кольори, виявлені на фотографії
objectsInPhoto	varchar(1000)	-	-	Інформація про об’єкти, виявлені на фотографії
nickname	varchar(50)	-	+	Нікнейм користувача, який здійснив запит

Таблиця «UserRole» (табл. 3.5) зберігає інформацію про ролі користувачів у системі. Ця таблиця використовується для визначення рівнів доступу користувачів до різних частин вебзастосунку.

Таблиця 3.5 – Структура таблиці «UserRole»

Назва поля	Тип даних	ПК	ЗК	Опис поля
roleId	int	+	-	Унікальний ідентифікатор ролі
roleName	varchar(100)	-	-	Назва ролі користувача в системі

Під час проєктування бази даних для вебзастосунку була створена ER-діаграма, яка наглядно відображає всі сутності бази даних та зв’язки між ними (рис. 3.3).

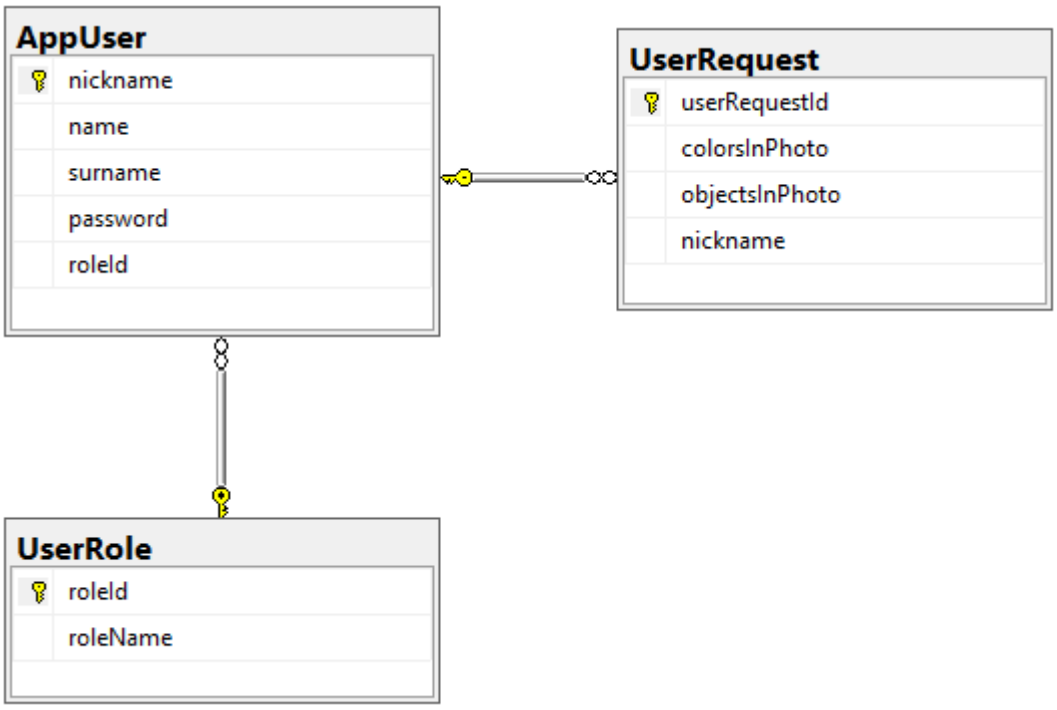

Рисунок 3.3 – Діаграма бази даних

ER-діаграма є ключовим елементом для подальшої розробки функціоналу системи. Вона надає уявлення про структуру даних, взаємозв’язки між ними та вимоги системи до зберігання та обробки інформації.

3.3 Реалізація вебзастосунку для класифікації одягу

Починаючи розробку інтерфейсу користувача для класифікації одягу, особлива увага приділяється створенню зручного та інтуїтивно зрозумілого інтерфейсу, який би відповідав потребам користувачів та сприяв ефективному взаємодію з системою. Інтерфейс користувача проєктується з метою максимального спрощення процесу завантаження зображень та отримання результатів класифікації. Для підтримки функціональності вебзастосунку і збереження даних про користувацькі запити, розробляється клас «ImageProcessorDbContext», код якого представлено на рисунку 3.4. Цей клас є ключовим компонентом в архітектурі системи, оскільки він виконує роль мосту між базою даних та логікою програми, забезпечуючи ефективне зберігання та організацію даних.

```
public ImageProcessorDbContext(DbContextOptions<ImageProcessorDbContext> options,
    IConfiguration appConfig)...
8 references
public virtual DbSet<AppUser> AppUsers { get; set; }
2 references
public virtual DbSet<UserRequest> UserRequests { get; set; }
4 references
public virtual DbSet<UserRole> UserRoles { get; set; }
0 references
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)...
0 references
protected override void OnModelCreating(ModelBuilder modelBuilder) ...
1 reference
partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

Рисунок 3.4 – Реалізація класу «ImageProcessorDbContext»

Клас «ImageProcessorDbContext» є частиною контексту бази даних Entity Framework Core та складається із:

– конструктори: Цей клас має два конструктори. Перший конструктор «ImageProcessorDbContext» не приймає параметрів та використовується за замовчуванням. Другий конструктор приймає параметри, включаючи

параметр options, що містить конфігурацію контексту бази даних, та параметр appConfig, який використовується для зчитування конфігураційних даних програми;

– властивості AppUsers, UserRequests та UserRoles представляють сутності бази даних, з якими можна взаємодіяти за допомогою Entity Framework Core. Кожна з цих властивостей відображається на відповідну таблицю бази даних;

– метод «OnConfiguring» встановлює параметри конфігурації для контексту бази даних, зокрема використовує підключення до SQL Server, яке зчитується з конфігураційного файлу застосунку. Цей метод викликається автоматично під час створення контексту бази даних;

– метод «OnModelCreating» визначає модель даних для контексту бази даних за допомогою об’єкта «ModelBuilder». Він налаштовує відносини між сутностями (таблицями) бази даних, встановлює властивості та обмеження. Цей метод викликається автоматично при визначенні моделі даних.

У фрагменті коду зображеному на рисунку 3.5 ініціалізується вебзастосунок за допомогою ASP.NET Core, використовуючи зручний API для конфігурації та реєстрації сервісів. Спочатку створюється новий екземпляр застосунку за допомогою «WebApplication.CreateBuilder», який включає в себе аргументи командного рядка.

```
var builder = WebApplication.CreateBuilder(args);
//Add file whith configuration, which contains paths
builder.Configuration.AddJsonFile("pathsSettings.json");
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<ImageProcessorDbContext>();
builder.Services.AddScoped<IDetectObject, YoloV8ProcessImgService>();
builder.Services.AddScoped<IPhotoInformation, PhotoInfYoloV8Service>();

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/LoginLogoutReg/Login";
        options.AccessDeniedPath = "/Home/Index";
    });
```

Рисунок 3.5 – Ініціалізація вебзастосунку

Для керування налаштуваннями, конфігурація застосунку розширюється шляхом додавання JSON-файлу з конфігурацією, що містить

шляхи до важливих ресурсів або налаштувань, за допомогою методу «AddJsonFile». Це дозволяє централізовано управляти конфігурацією і забезпечує легке внесення змін у майбутньому.

Потім, в контейнер сервісів додаються необхідні залежності: «AddControllersWithViews» використовується для додавання підтримки контролерів та відображень, що є стандартним для MVC-застосунків. Також реєструється контекст бази даних «ImageProcessorDbContext», що використовується для інтеграції з базою даних через Entity Framework.

Для обробки зображень і збору інформації про фотографії використовуються спеціалізовані сервіси: «IDetectObject» та «IPhotoInformation», для яких реєструються їх реалізації YoloV8ProcessImgService та PhotoInfYoloV8Service відповідно. Ці сервіси відповідають за виявлення об’єктів на зображеннях та аналіз фотографій, використовуючи модель YOLOv8. Залежності з AddScoped гарантують, що для кожного HTTP-запиту створюються нові екземпляри цих сервісів, що забезпечує ізоляцію даних між запитами.

Рисунок 3.6 зображає вміст конфігураційному файлі «pathsSettings.json» вебзастосунку.

```
{
  "DbConnectionString": {
    "DefaultConnection": "Server=COMPUTER;Database=testDB;Trusted_Connection=True;TrustServerCertificate=True;"
  },

  "NeuralMode": {
    "DefaultModel": "NeuralNetworkModel/yolov8m.onnx",
    "ClothesClassifierModel": "NeuralNetworkModel/clothesClasificationbest.onnx"
  }
}
```

Рисунок 3.6 – Вміст конфігураційного файлу

У конфігураційному файлі визначено дві основні секції: параметри підключення до бази даних і параметри нейронних мереж. Секція «DbConnectionString» містить параметри для підключення до бази даних, де «DefaultConnection» включає інформацію про сервер, назву бази даних, а також налаштування для довіреного з’єднання. Секція «NeuralMode» визначає

шляхи до моделей нейронних мереж, які використовуються в системі. «DefaultModel» вказує на модель YOLOv8m, що зберігається у вказаному каталозі, а «ClothesClassifierModel» містить шлях до спеціалізованої моделі для класифікації одягу. Ці налаштування дозволяють гнучко керувати підключенням до бази даних та використанням нейронних мереж у вебзастосунку.

У наведеному на рисунку 3.7 фрагменті коду показано процес налаштування та запуску вебзастосунку використовуючи ASP.NET Core.

```
builder.Services.AddAuthorization();
var app = builder.Build();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseCookiePolicy();
app.UseAuthorization();
app.MapControllerRoute(
    name: "admin",
    pattern: "{area:exists}/{controller=AdminHome}/{action=Index}/{id?}");
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

Рисунок 3.7 – Код процесу налаштування та запуску

Після реєстрації необхідних сервісів у контейнері сервісів, вебзастосунок будується за допомогою методу «Build». Далі, застосунок налаштовується на використання HTTPS перенаправлень, що забезпечує безпечне з’єднання шляхом автоматичного перенаправлення HTTP запитів на HTTPS. Включення обробки статичних файлів дозволяє серверу правильно обслуговувати зображення, CSS файли та JavaScript, що є необхідним для багатьох вебсторінок.

Функціональність маршрутизації активується за допомогою «UseRouting», що дозволяє вебзастосунку правильно визначати, який контролер та дія мають бути викликані на основі URL запиту. Це критично важливо для функціонування MVC архітектури. Використання політики кукі та системи авторизації забезпечує відповідність запитів умовам безпеки та

доступу до ресурсів. Конфігурація маршрутів вебзастосунку включає два основних маршрути. Перший, «admin», призначений для обробки запитів в адміністративну зону з встановленою точкою входу в контролер «AdminHome» та дію «Index». Другий, «default», задає стандартні маршрути до головного контролера «Home» та його дії «Index». Обидва маршрути дозволяють передавати опціональні параметри, як ідентифікатори сутностей.

Завершується налаштування запуском вебзастосунку через метод «Run», що змушує застосунок постійно слухати вхідні запити та обслуговувати їх відповідно до налаштованих маршрутів та конфігурацій.

На рисунку 3.8 представлено код ініціалізації списку об’єктів в мові програмування C#, який використовується для створення навігаційного меню

```
@using WebImageProcessor.ViewModel.Models
@{
    var menuItems = new List<MenuItemModel>
    {
        new MenuItemModel("Home Page", "ft-home", "Головна", "/Home/Index"),
        new MenuItemModel("Statistic Page", "ft-pie-chart", "Статистика", "/StatisticPage/Index"),
        new MenuItemModel("Documentation Page", "ft-book", "Про проект", "/DocumentationPage/Index"),
    };
}
```

Рисунок 3.8 – Код ініціалізації списку об’єктів

Код починається з директиви «using», яка підключає простір імен «WebImageProcessor.ViewModel.Models», де ймовірно знаходиться визначення класу «MenuItemModel». Далі, в блок коду, який виконується в контексті вебсторінки або компонента, вводиться змінна «menuItems». Ця змінна представляє собою список елементів меню, де кожен елемент меню ініціалізується як новий екземпляр класу «MenuItemModel». Кожен екземпляр цього класу містить чотири параметри: назву сторінки, іконку (використовується для візуального представлення в інтерфейсі користувача), текст пункту меню, що буде відображений користувачу, та маршрут (URL), по якому буде здійснено перехід при виборі даного пункту.

На рисунку 3.9 представлено фрагмент HTML коду, який описує частину інтерфейсу користувача для аналізу фотографій.

```
<!-- Опис функціоналу -->
<div class="card">
    <div class="card-content">
        <div class="card-body">
            <div class="container">
                <div class="row justify-content-center">
                    <div class="col-md-8 text-center">
                        <h1>Ласкаво просимо до Photo Analyzer</h1>
                        <p>
                            Цей веб застосунок створений для обробки та аналізу ваших фотографій. Виберіть або зробіть фото,
                            після чого натисніть кнопку опрацювати зображення і застосунок видасть інформацію по вашому зображенню.
                        </p>
                        <button class="btn btn-primary" id="uploadBtn">Завантажити фото</button>
                        <button class="btn btn-success" id="captureBtn">Зробити фото</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Рисунок 3.9 – Фрагмент коду інтерфейсу користувача

Інтерфейс використовує структуру картки (card), що є популярним елементом дизайну у багатьох фреймворках для фронтенду, таких як Bootstrap. Основний контент картки вміщений у блок, що містить заголовок «Ласкаво просимо до Photo Analyzer», який оформлено як основний заголовок (h1) для привернення уваги користувача. Під заголовком розміщений абзац тексту, який інформує користувачів про основний функціонал вебзастосунку: можливість завантаження або зйомки фотографій та подальшого аналізу зображень за допомогою застосунку.

Для інтеракції з користувачем в коді передбачено дві кнопки: одна з них («Завантажити фото») призначена для завантаження фотографій з пристрою користувача, а друга («Зробити фото») – для активації камери та зйомки фото безпосередньо через інтерфейс застосунку. Обидві кнопки стилізовані відповідними класами для візуального виділення та зручності використання.

На рисунку 3.10 представлено код методу «DrawBoxesOnImage», який призначений для обробки зображення, завантаженого користувачем, та відображення на ньому рамок, які позначають розташування об’єктів, виявлених за допомогою нейронної мережі YOLOv8.

```
public async Task<Image> DrawBoxesOnImage(IFormFile file, IConfiguration appConfig) {
    Image resultImage;
    using (var stream = file.OpenReadStream()) {
        using (Image image1 = Image.Load(stream)) {
            // Повертаємо потік на початок для завантаження другого зображення
            stream.Seek(0, SeekOrigin.Begin);
            using (Image image2 = Image.Load(stream)) {
                YoloV8 predictor = new YoloV8(GetModelPath(appConfig));
                var result = await predictor.DetectAsync(image1);
                resultImage = await result.PlotImageAsync(image2);
                return resultImage;
            }
        }
    }
}
```

Рисунок 3.10 – Код методу «DrawBoxesOnImage»

Метод приймає файл зображення та конфігурацію застосунку як параметри. метод відкриває потік читання з файлу зображення. Він завантажує зображення двічі з того ж потоку: перше зображення (image1) використовується для детекції об’єктів за допомогою моделі YOLOv8, а друге (image2) – для відображення результатів. Це може бути корисним, наприклад, якщо потрібно зберегти оригінал зображення не зміненим або використовувати різні зображення для аналізу та візуалізації. Після детекції об’єктів на першому зображенні за допомогою методу DetectAsync класу YoloV8, результати (об’єкти та їх координати) використовуються для рисування рамок на другому зображенні. Функція PlotImageAsync відповідає за це рисування. На завершення, друге зображення з нарисованими рамками повертається як результат виконання методу.

Рисунок 3.11 відображає код методу «AnalysePhotoAsync», який призначений для аналізу фотографії, що завантажується користувачем, з метою визначення середніх значень кольорів у всіх визначених рамках (боксах), а також ідентифікації назв об’єктів у цих рамках.

```
public async Task<(string, string)> AnalysePhotoAsync(IFormFile file, IConfiguration appConfig) {
    using (var stream = file.OpenReadStream()) {
        using (SixLabors.ImageSharp.Image image = SixLabors.ImageSharp.Image.Load(stream)) {

            YoloV8 predictor = new YoloV8(YoloV8ProcessImgService.GetModelPath(appConfig));
            Bitmap bitmap = ConvertImageSLToBitmap(image);

            var result = await predictor.DetectAsync(image);

            if (result.Boxes.Count == 0) {
                return ("", "");
            } else {
                return (CheckColorsAllBoxes(bitmap, result), CheckNameAllBoxes(result));
            }
        }
    }
}
```

Рисунок 3.11 – Код методу «AnalysePhotoAsync»

Метод розпочинає роботу з відкриття потоку читання для файлу, отриманого як параметр. Зображення завантажується у пам’ять за допомогою бібліотеки «SixLabors.ImageSharp». Далі, ініціалізується екземпляр класу YoloV8, який використовується для виявлення об’єктів на зображенні. Шлях до моделі машинного навчання отримується з конфігурації застосунку. Після завантаження зображення, воно конвертується в формат «Bitmap», який використовується для подальшого аналізу кольорів. Процес виявлення виконується асинхронно, і результати, зокрема, рамки з виявленими об’єктами, зберігаються у змінній «result».

Якщо рамки не були знайдені (тобто їх кількість дорівнює нулю), метод повертає пару порожніх рядків. Якщо ж рамки є, виконується аналіз кольорів у всіх виявлених рамках за допомогою функції «CheckColorsAllBoxes» та отримання назв об’єктів з функції «CheckNameAllBoxes». Результати цих функцій повертаються як кортеж двох рядків, що містять інформацію про середні кольори та назви об’єктів відповідно.

На рисунку 3.12 представлено метод «CheckColorsAllBoxes» призначений для аналізу кольорів у всіх виявлених рамках (боксах) на зображенні. Цей метод приймає два параметри: об’єкт «Bitmap», що представляє зображення, та об’єкт «IDetectionResult», який містить результати виявлення, включаючи інформацію про рамки.

```
static string CheckColorsAllBoxes(Bitmap bitmap, IDetectionResult detectionResult) {
    string result = "";

    if (detectionResult.Boxes.Count == 0) {
        return result;
    } else {
        for (int i = 0; i < detectionResult.Boxes.Count; i++) {
            int x = detectionResult.Boxes[i].Bounds.X;
            int y = detectionResult.Boxes[i].Bounds.Y;
            int width = detectionResult.Boxes[i].Bounds.Width;
            int height = detectionResult.Boxes[i].Bounds.Height;
            result += "|" + ConvertRGBToHEX(GetAvgColorInBox(bitmap, x, y, width, height));
        }
        return result;
    }
}
```

Рисунок 3.12 – Код методу «CheckColorsAllBoxes»

На початку методу ініціалізується порожній рядок «result», який буде використовуватись для зберігання результатів аналізу. Якщо детектор не виявив жодної рамки на зображенні, метод відразу повертає порожній рядок. Якщо ж рамки є, метод перебирає кожну з них у циклі.

Для кожної рамки визначаються її координати та розміри за допомогою властивостей X, Y, Width і Height. Використовуючи ці дані, викликається функція «GetAvgColorInBox», яка обраховує середній колір у відповідній області зображення. Результат цього обчислення, який є значенням кольору в форматі RGB, конвертується у формат HEX за допомогою функції «ConvertRGBToHEX». Конвертоване значення HEX додається до рядка результатів, розділене символом | для кожної рамки.

Після обробки всіх рамок, рядок «result», що містить послідовність HEX кодів кольорів для кожної рамки, повертається як результат роботи методу. Такий підхід дозволяє ефективно визначити та зібрати інформацію про колірну палітру об’єктів на зображенні.

На рисунку 3.13 представлено метод «GetAvgColorInBox» визначає середнє значення кольору у певній області зображення, яку визначають параметри координати початкової точки (startX, startY), ширина (width) та

висота (height). Цей метод приймає зображення у форматі Bitmap та вказані координати та розміри області для аналізу.

```
// Метод для отримання середнього значення кольору у боксі з одягом
1 reference
static System.Drawing.Color GetAvgColorInBox(Bitmap bitmap, int startX, int startY, int width, int height) {
    // Змінні для сумарних значень кольорів
    int totalR = 0, totalG = 0, totalB = 0;

    // Перебір пікселів у вказаному квадраті
    for (int x = startX; x < startX + width; x++) {
        for (int y = startY; y < startY + height; y++) {
            // Отримання кольору пікселя
            System.Drawing.Color pixelColor = bitmap.GetPixel(x, y);
            // Додавання кольорів для подальшого обрахунку середнього
            totalR += pixelColor.R;
            totalG += pixelColor.G;
            totalB += pixelColor.B;
        }
    }
    // Обчислення середнього кольору
    int averageR = totalR / (width * height);
    int averageG = totalG / (width * height);
    int averageB = totalB / (width * height);
    // Створення та повернення середнього кольору
    return System.Drawing.Color.FromArgb(averageR, averageG, averageB);
}
```

Рисунок 3.13 – Код методу «GetAvgColorInBox»

У методі використовується вкладений цикл для перебору всіх пікселів у вказаній прямокутній області. Для кожного пікселя за допомогою методу «GetPixel» отримується його колір, і компоненти кольорів RGB цього пікселя додаються до загальних сумарних значень totalR, totalG, totalB.

Після завершення перебору всіх пікселів, вираховуються середні значення для кожного кольорового компонента, ділячи загальні суми на кількість пікселів в області (що дорівнює width * height).

На завершення, створюється новий колір за допомогою методу «Color.FromArgb», використовуючи обчислені середні значення RGB, і цей колір повертається як результат роботи методу. Це дозволяє отримати узагальнений колір для великої області зображення, що може бути корисно для аналізу та візуалізації характеристик об’єктів на фото.

Отже, у даному підрозділі було частково описано реалізацію вебзастосунку.

3.4 Інструкція користувача

Одразу після запуску застосунку, користувачу буде представлено стартова сторінка із головним меню у лівій частині сторінки (рис. 3.14).

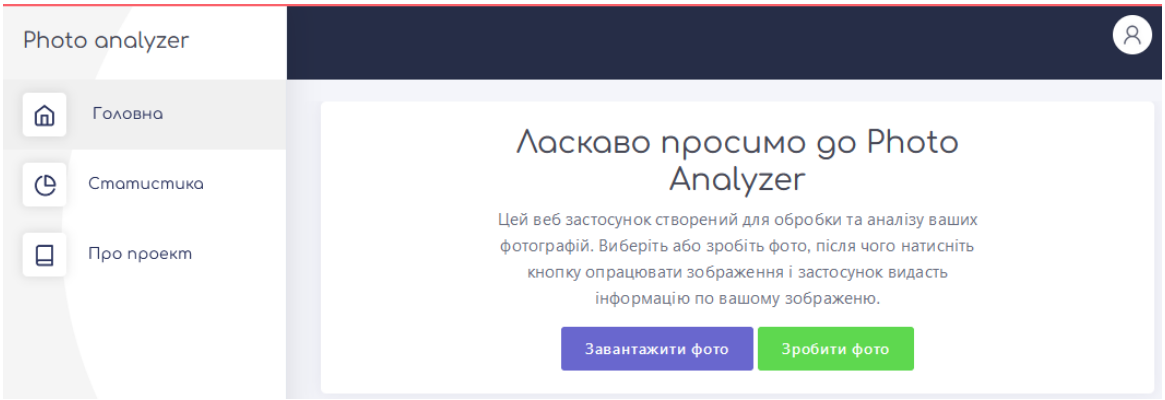

Рисунок 3.14 – Стартова сторінка

Для завантаження фотографії користувач повинен перейти на головну сторінку програми та вибрати фотографію одягу на своєму пристрої. Після цього він має натиснути кнопку «Завантажити фото» та дочекатися завантаження фотографії на сервер (рис. 3.15).

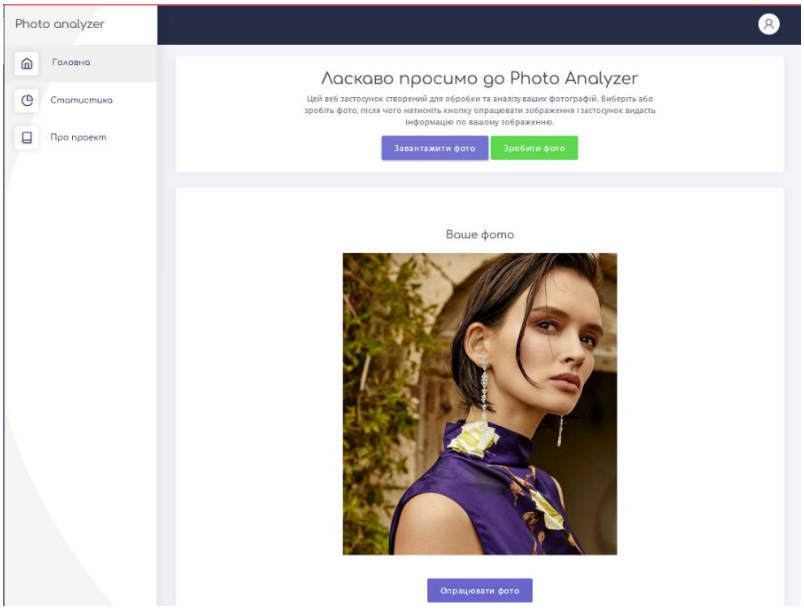

Рисунок 3.15 – Завантаження фото

Після завантаження фотографії система автоматично розпочне її обробку. З використанням алгоритмів машинного навчання програма аналізує елементи на зображенні, класифікуючи типи одягу та їх характеристики. Закінчується процес виявленням одягу на фотографії (рис. 3.16).

Рисунок 3.16 – Виявлення одягу на завантаженому зображенні

Система позначить ключові області на одязі, які будуть аналізуватися. Виявлений на зображенні одяг буде виділений рамкою, а в верхній його частині буде виведено надпис із класом одягу. У даному випадку було виявлено сукню.

Натиснувши кнопку «Зберегти» користувач зможе завантажити зображення на свій комп’ютер. Процес може зайняти декілька секунд, в залежності від складності зображення і навантаження на сервер.

Для реєстрації нового користувача у системі, користувачу потрібно натиснути іконку у правій частині вікна, перейти по посиланню «Реєстрація» та заповнити форму реєстрації користувача (рис. 3.17).

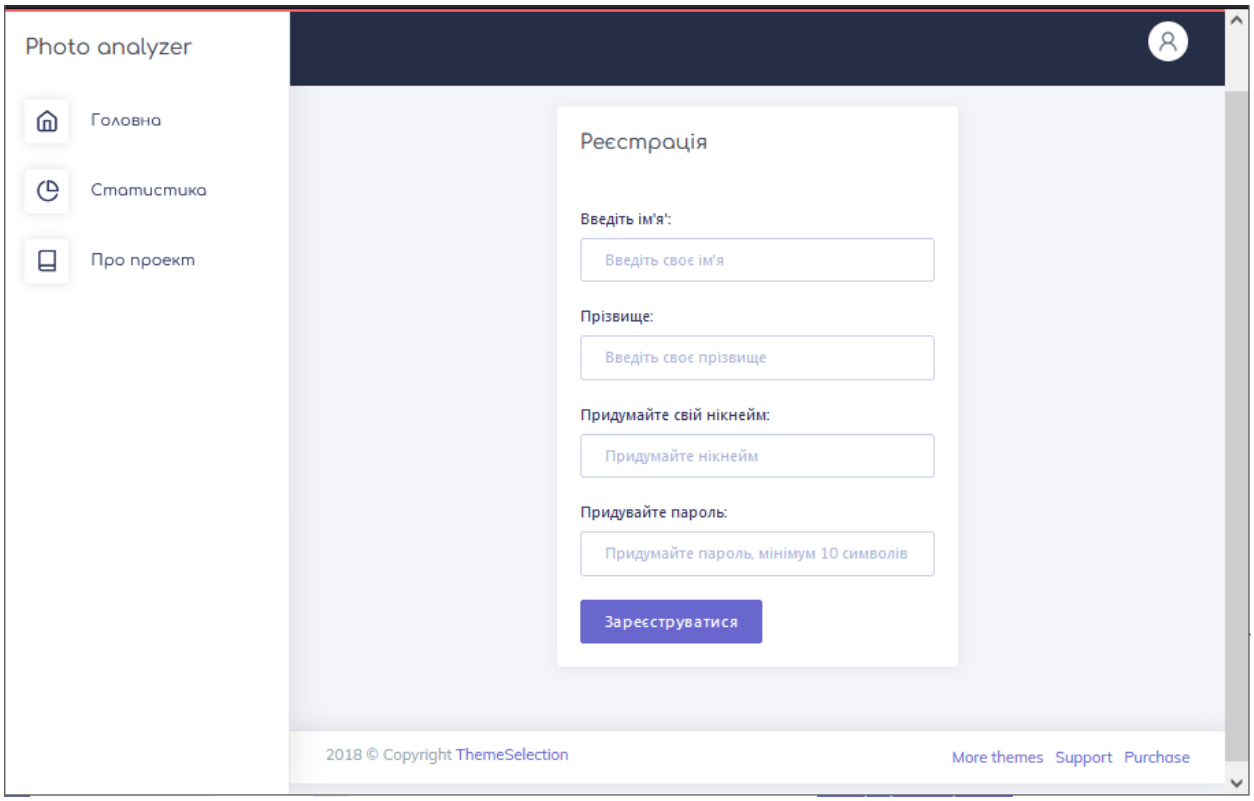

Рисунок 3.17 – Сторінка для реєстрації користувача

Форма реєстрації на даному вебінтерфейсі містить такі поля, які користувач має заповнити для створення акаунту:

– ім’я. Користувач має ввести своє ім’я. Це поле може використовуватися для персоналізації звернень у майбутньому;

– прізвище. користувач повинен вказати своє прізвище;

– електронна адреса. Необхідно вказати електронну адресу, яка буде використовуватися для входу в систему та комунікації;

– пароль. Користувач повинен створити пароль, що містить мінімум 10 символів, для забезпечення безпеки свого акаунту.

Після заповнення всіх полів користувач має натиснути на кнопку «Зареєструватися» для завершення процесу реєстрації.

Зареєстровані користувачі можуть бачити статистику сайту (рис. 3.18).

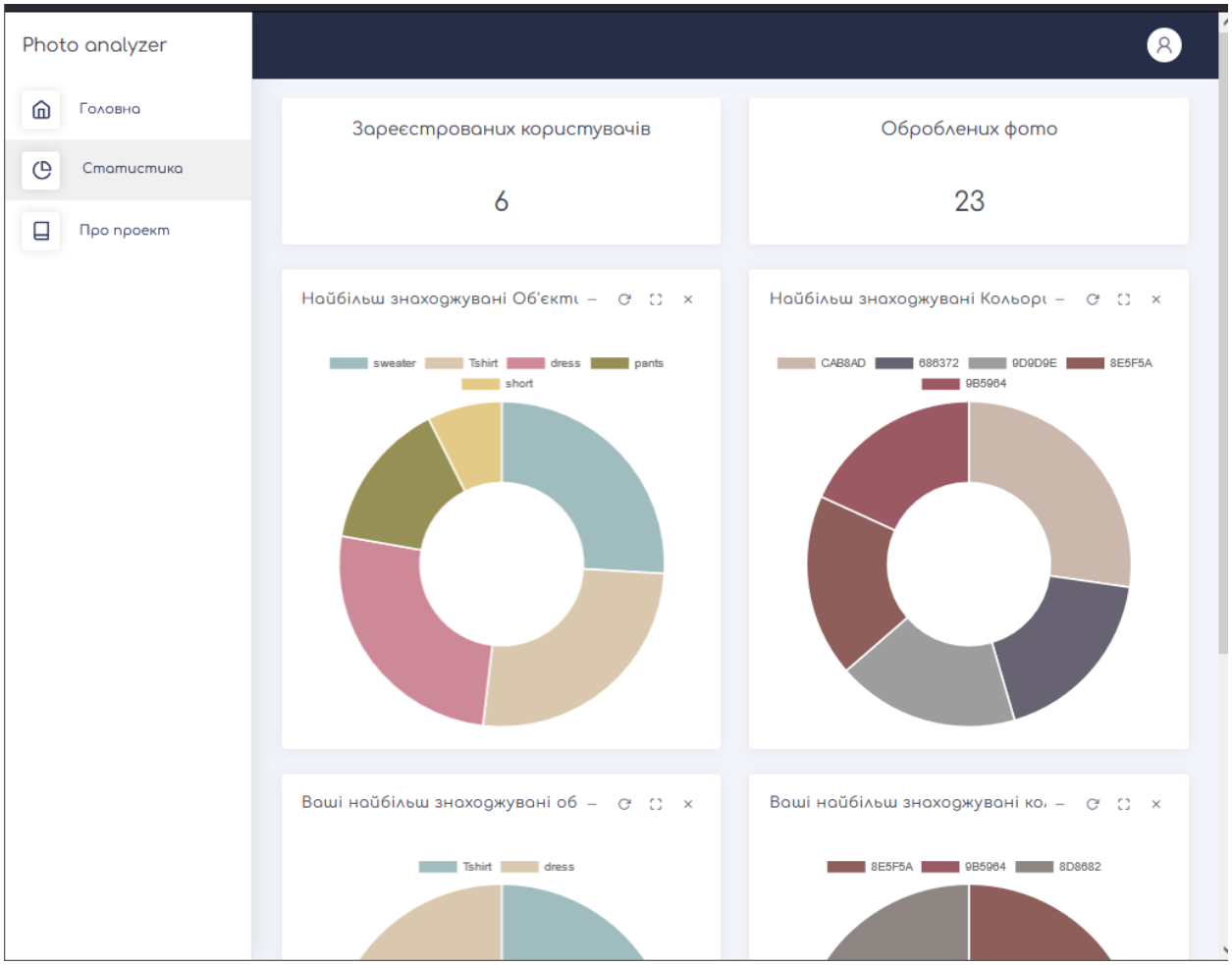

Рисунок 3.18 – Статистика вебзастосунку

Статистична сторінка для зареєстрованих користувачів вебінтерфейсу відображає різні типи даних через графічний контент та цифрові індикатори, зокрема:

– у верхній частині сторінки є два індикатори: один показує кількість зареєстрованих користувачів, а інший — кількість опрацьованих фотографій.

– нижче розташовані два кругові діаграми. Перша діаграма демонструє розподіл найбільш аналізованих типів одягу, таких як светри, футболки, сукні, штани та шорти, вказуючи їхню популярність серед користувачів;

– друга кругова діаграма показує розподіл найбільш зустрічаних кольорів на аналізованих фотографіях, кожен колір представлений унікальним HEX-кодом;

– внизу сторінки знаходяться ще дві діаграми. Одна демонструє типи одягу, які найчастіше виявляють користувачі, а інша вказує на найбільш популярні кольори, вибрані користувачами.

Ця статистична сторінка дозволяє користувачам оцінити загальні тренди і переваги у використанні застосунку, що може бути корисно для аналізу модних тенденцій та вибору одягу.

Обравши пункт меню «Про проект», користувачу буде представлена детальна інформація про особливості вебзастосунку для класифікації одягу (рис. 3.19).

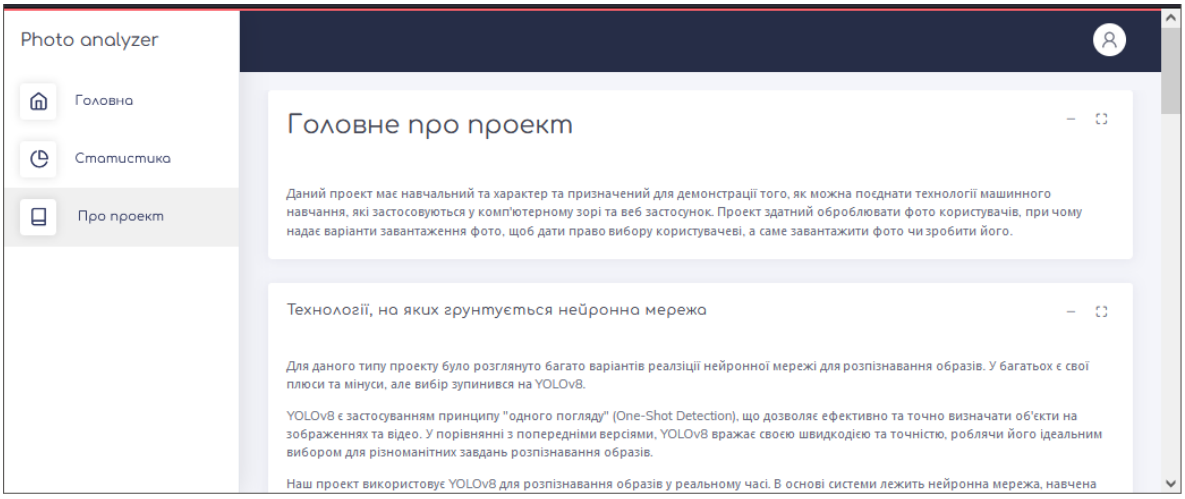

Рисунок 3.19 – Сторінка «Про проект»

На сторінці «Про проект» вебінтерфейсу Photo Analyzer надається інформація про цілі та технічні аспекти проєкту.

3.5 Тестування функціональності користувацького інтерфейсу

Перевірка якості програмного продукту є критичною фазою у його розробці, яка відіграє ключову роль у забезпеченні відповідності системи зазначеним стандартам та вимогам користувачів. Через процес тестування відбувається детальний аналіз функціонування програми для ідентифікації та

усунення збоїв, що значно підвищує якість кінцевого продукту та його стабільність у роботі.

Окрім того, тестування охоплює оцінку різноманітних аспектів програмного продукту, включаючи його ефективність, безпеку та відмовостійкість, що дозволяє виявити та виправити потенційні недоліки. Такий комплексний підхід є невід'ємною частиною процесу розробки, спрямованого на успішний реліз та оптимізацію системи. В результаті, процес тестування не лише забезпечує високу якість програмного продукту, але й сприяє розширенню його функціональних можливостей.

Тест-кейс №1. Реєстрація користувача.

Мета: Перевірити процес реєстрації нового користувача на платформі «Photo analyzer» для забезпечення правильної роботи системи авторизації.

Кроки сценарію:

– відкрити головну сторінку програми;

– перейти до форми реєстрації;

– ввести ім'я в поле «Введіть ім'я»;

– ввести прізвище в поле «Введіть прізвище»;

– ввести електронну адресу в поле «Введіть електронну адресу»;

– створити пароль, згідно з вимогами системи, та ввести його в поле «Введіть пароль»;

– натиснути кнопку «Зареєструватися».

Очікуваний результат: Після заповнення усіх полів і натискання кнопки «Зареєструватися», користувач отримає повідомлення про успішну реєстрацію та буде перенаправлений на головну сторінку програми з входом в свій новий аккаунт.

Отриманий результат: успішний вхід до системи із обліковими даними.

Тест-кейс №2. Виявлення сукні на завантаженому зображенні.

Мета: Перевірити здатність програми «Photo Analyzer» правильно ідентифікувати та класифікувати сукню на фотографії.

Кроки сценарію:

– відкрити головну сторінку програми «Photo Analyzer»;

– натиснути на кнопку «Завантажити фото» та вибрати зображення, на якому присутня сукня;

– дочекатися автоматичного завантаження зображення в систему;

– система повинна автоматично аналізувати зображення та визначати предмети одягу, зокрема сукню;

– перевірити, чи з’являються на зображенні відповідні мітки та анотації, що вказують на присутність сукні.

Очікуваний результат: Програма має коректно ідентифікувати сукню на фотографії, виділяти її мітками, та відображати інформацію про тип одягу (сукня) у відповідному полі під зображенням.

Отриманий результат: успішне виявлення сукні на завантаженому зображенні та відображати інформацію про тип одягу.

ВИСНОВКИ

У рамках кваліфікаційної роботи було реалізовано вебзастосунок для класифікації одягу, що використовує передові методи машинного навчання з використанням нейронних мереж. Цей застосунок спроєктовано так, щоб допомогти користувачам швидко і точно класифікувати одяг за фотографіями, що може бути корисним у різних сферах.

У першому проведено аналіз рішень, існуючих у галузі класифікації одягу, включаючи огляд рішень, таких як ViSenze та Cortexica, із якого були виявлені їхні основні переваги та недоліки. Огляд використання нейронних мереж, зокрема таких як LeNet-5, AlexNet, і CNN, дозволив обрати найбільш ефективні підходи для подальшої розробки.

В другому розділі було проведено детальне порівняння та вибір архітектур нейронних мереж, де YOLOv8 було визнано оптимальним варіантом для задач класифікації одягу. Розробка методу класифікації з використанням YOLOv8 включала створення алгоритмів навчання та прогнозування, що забезпечують високу точність і швидкість обробки зображень.

Третій розділ був присвячений технічній реалізації вебзастосунку, включаючи вибір технологічного стеку, де було обрано ASP.NET Core MVC і MS SQL Server як основу для розробки рішення. Проєктування та розробка бази даних забезпечила ефективне зберігання і обробку даних, а інструкція користувача та тестування функціональності користувацького інтерфейсу підтвердили зручність та ефективність вебзастосунку.

Проведена робота демонструє ефективне застосування сучасних технологій нейронних мереж та веброзробки для створення рішень у галузі моди, що забезпечують значні переваги як для кінцевого користувача.

Результати роботи апробовано у вигляді 3 тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yue, X., Zhang, C., Fujita, H., & Lv, Y. (2021). Clothing fashion style recognition with design issue graph. Applied Intelligence, 51(6), 3548-3560.

2. ViSenze Reviews & Product Details. URL: https://www.g2.com/products/visenze/reviews (дата звернення 04.05.2024).

3. ViSenze Employee Reviews. URL: https://sg.indeed.com/cmp/Visenze/reviews (дата звернення 04.05.2024).

4. What is ViSenze? URL: https://www.capterra.com/p/181026/ViSenze/ (дата звернення 04.05.2024).

5. ViSenze Reviews, Running Projects, Clients & Founder Details. URL: https://companies.makeanapplike.com/profile/visenze (дата звернення 04.05.2024).

6. Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. IEEE signal processing magazine, 35(1), 53-65.

7. What is Cortexica? URL: https://www.capterra.com/p/180782/Cortexica/ (дата звернення 04.05.2024).

8. Cortexica Vision Systems. URL: https://www.glassdoor.com.hk/Reviews/Cortexica-Vision-Systems-Researcher-Reviews-EI_IE918426.0,24_KO25,35.htm (дата звернення 04.05.2024).

9. Emambakhsh, M., Bay, A., & Vazquez, E. (2019). Convolutional recurrent predictor: Implicit representation for multi-target filtering and tracking. IEEE Transactions on Signal Processing, 67(17), 4545-4555.

10. Murtaza, F., Yousaf, M. H., Velastin, S. A., & Qian, Y. (2018). End-to-end temporal action detection using bag of discriminant snippets. IEEE Signal Processing Letters, 26(2), 272-276.

11. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

12. Toshev, A., & Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1653-1660).

13. Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13* (pp. 184-199). Springer International Publishing.

14. Bhattacharyya, S. (2011). A brief survey of color image preprocessing and segmentation techniques. Journal of Pattern Recognition Research, 1(1), 120-129.

15. Zhang, D., Liu, B., Sun, C., & Wang, X. (2011). Learning the Classifier Combination for Image Classification. J. Comput., 6(8), 1756-1763.

16. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5, 115-133.

17. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

18. Матвійчук, А. В. (2003). Дослідження залежності якості прогнозування курсів цінних паперів нейронними мережами від форми подання вхідних даних. *Збірник наукових праць Черкаського державного технологічного університету. Серія: Економічні науки*, 380.

19. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

20. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7), 1527-1554.

21. Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13 (pp. 818-833). Springer International Publishing.

22. Zhu, X. X., Tuia, D., Mou, L., Xia, G. S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. IEEE geoscience and remote sensing magazine, 5(4), 8-36.

23. Zhong, Y., Fei, F., & Zhang, L. (2016). Large patch convolutional neural networks for the scene classification of high spatial resolution imagery. Journal of Applied Remote Sensing, 10(2), 025006-025006.

24. Wang, W., Yang, Y., Wang, X., Wang, W., & Li, J. (2019). Development of convolutional neural network and its application in image classification: a survey. Optical Engineering, 58(4), 040901-040901.

25. Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004, May). Visual categorization with bags of keypoints. In Workshop on statistical learning in computer vision, ECCV (Vol. 1, No. 1-22, pp. 1-2).

26. Wang, W., Yang, Y., Wang, X., Wang, W., & Li, J. (2019). Development of convolutional neural network and its application in image classification: a survey. Optical Engineering, 58(4), 040901-040901.

27. Mukherjee, S., Beard, C., & Li, Z. (2024). MODIPHY: Multimodal Obscured Detection for IoT using PHantom Convolution-Enabled Faster YOLO. arXiv preprint arXiv:2402.07894.

28. Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.

29. Prasad, S. B. R., & Chandana, B. S. (2023). Mobilenetv3: a deep learning technique for human face expressions identification. International Journal of Information Technology, 15(6), 3229-3243.

30. Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

31. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to

training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

32. Jois, P. S., Manjunath, A., & Fevens, T. (2021, December). Boosting segmentation performance across datasets using histogram specification with application to pelvic bone segmentation. In 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 1364-1369). IEEE.

33. Grossman, R., Haim, O., Abramov, S., Shofty, B., & Artzi, M. (2021). Differentiating small-cell lung cancer from non-small-cell lung cancer brain metastases based on MRI using efficientnet and transfer learning approach. Technology in cancer research & treatment, 20, 15330338211004919.

34. Salscheider, N. O. (2021, January). Featurenms: Non-maximum suppression by learning feature embeddings. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 7848-7854). IEEE.

35. Freeman, A. (2016). Pro Asp. net core MVC. Apress.

36. Cantelon, M., Harter, M., Holowaychuk, T. J., & Rajlich, N. (2014). Node. js in Action (pp. 17-20). Greenwich: Manning.

37. De Smedt, T., & Daelemans, W. (2012). Pattern for python. The Journal of Machine Learning Research, 13(1), 2063-2067.

38. MacLennan, J., Tang, Z., & Crivat, B. (2011). Data mining with Microsoft SQL server 2008. John Wiley & Sons.

39. Greenwald, R., Stackowiak, R., & Stern, J. (2013). Oracle essentials: Oracle database 12c. " O'Reilly Media, Inc.".

40. Кравченко, Д. (2024). Переваги та недоліки класифікації одягу на відвідувачах по їх зображенням. *У Матеріали 28-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті»* (том 7, стор. 63-65).