Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Натальчишин Олександр Вікторович
студент групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Система для автоматичного усунення фону і ретушування зображень на основі генеративно-змагальных глибинних нейронних мереж

Спеціальність:
121 – Інженерія програмного забезпечення
Освітня програма:
Інженерія програмного забезпечення

Керівник:
Рувінська Вікторія Михайлівна
канд. техн. наук, професор

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____проф. Любченко В.В.
"_____" _____ 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Натальчишину Олександру Вікторовичу, АС-161

1. Тема проекту (роботи) Система для автоматичного усунення фону і ретушування зображень на основі генеративно-змагальних глибинних нейронних мереж.

Керівник роботи: Рувінська Вікторія Михайлівна, кандидат технічних наук, професор.

затверджені наказом ректора від "25"жовтня 2021 р.№374-в

2. Строк подання студентом проекту (роботи) 30.11.2021

3. Вихідні дані по проекту (роботі) Технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належіть розробити) Вступ. Особливості предметної області. Аналіз існуючих рішень. Вимоги до системи. Розробка автоматичних моделей. Проектування системи. Тестування програмної системи. Висновки. Список літератури

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

У відповідності до слайдів електронної презентеції.

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Результат прийняв

7. Дата видачі завдання 28.08.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вивчення предметної області та аналіз програм-аналогів	10.09.2021	Виконано
2.	Формування вимог	21.09.2021	Виконано
4.	Розробка моделей	26.09.2021	Виконано
5.	Розробка класів	20.10.2021	Виконано
6.	Реалізація інтерфейсу програми	06.11.2021	Виконано
7.	Тестування програми	14.11.2021	Виконано
8.	Оформлення документації	17.11.2021	Виконано

Здобувач вищої освіти _____ *О. В. Натальчишин*

Керівник роботи _____ *В. М. Рувінська*

РЕФЕРАТ

Метою роботи є підвищення якості автоматичного очищення заднього фону і ретушування зображення за рахунок використання змагально-генеративних глибиних нейронних мереж (GAN).

Було розроблено моделі GAN для виконання автоматичного ретушування і очищення заднього фону зображення, а також проведено зменшення їх розмірів для підвищення продуктивності роботи без істотних погіршень у точності за допомогою методів квантування та дистиляції знань.

Розроблено архітектуру програмної системи, спроектовано графічний інтерфейс користувача і виконано розробку веб-застосування на основі створених моделей.

Нейронні мережі було реалізовано засобами мови програмування Python за допомогою бібліотек машинного навчання TensorFlow і PyTorch. Вони були імпортовані до інтерфейсу користувача, що було реалізовано засобами JavaScript за допомогою TensorFlowJS.

Ключові слова: генеративно-змагальні нейронні мережі (GAN), TensorFlow, Python, JavaScript, веб-застосування, фоторедактор.

ABSTRACT

The aim of the work is to improve the quality of automatic image background cleaning and image retouching using deep generative adversarial neural networks (GAN).

GAN models developed to automatically retouch and clean the background of the image, and their size reduced to increase productivity without significant loss in accuracy using quantization and distillation techniques.

The architecture of the software system was developed, the graphical user interface was designed and the development of a web application based on these models was performed.

In the work analyzes the existing solutions, identifies their shortcomings, formed functional and non-functional requirements for software and describes the behavior of the system. Developed models for automatically retouch and clear imge background. Developed the architecture of software system and designed user interface.

Neural networks were implemented by means of Python languages using machine-learning libraries – TensorFlow and PyTorch. They imported into the user interface, which was implemented using JavaScript, using TensorFlowJS.

Keywords: generative adversarial networks (GAN), TensorFlow, Python, JavaScript, web application, foto editor.

ВСТУП

В сучасному світі щодня публікуються мільйони зображень найрізноманітніших товарів, використовуючи тисячі різних платформ. Кожен користувач намагається виділяти свій товар на фоні мільйонів інших. Існують різні методики привернення уваги потенційних покупців: зниження ціни, акційні пропозиції, рекламні компанії та інше. І одним з найважливіших у даному переліку є безпосередньо якість зображення (фотографії), що представляє відповідний товар.

Це перше, що потрапляє у вічі покупцю, коли він зустрічає даний об’єкт. А лише потім йде його ціна, різноманітні характеристики та ознаки, відгуки від людей, що уже придбали і мають досвід його використання. Отже покупець радше перейде до більш детального розгляду саме того товару, зображення якого сподобається йому більше, яке є більш приємним для нього, ніж до товарів, що преставлені, наприклад, надто темними чи навпаки яскравими фотографіями.

Отже дуже важливо якісно виділяти зображення, що представляє власне товар для привернення уваги потенційного покупця на фоні десятків, а то і тисяч товарів-конкурентів. Для цього необхідно перед процесом завантаження зображення на платформу провести його попереднє редагування.

Проте більшість програм для редагування зображень мають певний набір інструментів, якість роботи яких повністю залежить від навичок і знань користувача у роботі з даним програмним продуктом, а також вимагають витрачати час на редагування кожного зображення.

Тобто задача створення автоматичного фоторедактора зображень є актуальною задачею.

Для створення автоматичного фоторедактора запропоновано використовувати генеративно змагальні нейронні мережі (GAN) [1]. На даний момент уже наявні різномітні методи з використанням GAN для вирішення різних задач з редагування зображень: від збільшення розміру зображення без втрати у якості до створення зображення місцевості з карти Google Maps [2].

Проте якість автоматичної роботи більшості з них недостатньо висока для створення рекламних фотографій, вони потребують багато ресурсів для своєї роботи, відсутній графічний інтерфейс для використання кінцевими користувачами, більшість інструментів не є безкоштовними.

Метою роботи є підвищення якості автоматичного очищення заднього фону і ретушування зображення за рахунок використання змагально-генеративних глибиних нейронних мереж (GAN).

Для досягнення мети в роботі потрібно вирішити наступні задачі:

- провести аналіз наявних методів і моделей для вирішення задач очищення і ретушування, порівняти їх внутрішню структуру і результати роботи;
- розробити моделі для автоматичного редагування;
- спроектувати архітектуру застосунку;
- спроектувати інтерфейс для користувача;
- розробити веб-застосування.

Об'єктом дослідження є фоторедактори.

Предметом дослідження є методи для створення змагально-генеративних мереж для автоматичного очищення заднього фону зображення і його ретушування.

В першому розділі розглянуто сучасний ринок фоторедакторів і алгоритмів для автоматичного редагування зображень, їх особливості і недоліки, створено таблицю порівняння програмних аналогів.

У другому розділі визначена мета і призначення розробки програмної системи, основні її технічні вимоги і функціональні можливості. Було проведено планування етапів розбробки програмної системи.

У третьому розділі визначено методи та моделі, взяті для розробки змагально-генеративних мережей для очищення фону і автоматичного ретушування зображеного об’єкта. Проведено аналіз та тестування на різних наборах даних наявних змагально-генеративних нейроних мережей і методів

для вибору найкращих, що взяті за основу розроблювальних. Описані їх головні особливості, внутрішня структури мержі в цілому і окремо для блоків.

В четвертому розділі визначено вимоги користувачів до системи в цілому. Визначені нефункціональні вимоги до програмної системи. Описано програмні класи, що створені на основі моделі концептуальних класів. Надано детальний опис методів класів, які розподілені на частини відповідно архітектурному шаблону MVC [3]. Проведено проектування інтерфейсу користувача.

В пятому розділі представлено інструкцію з використання програмного застосування. Проведено тестування програмної системи під час її використання користувачем відповідно до діаграми варіантів використання і описаних прецедентів. Проведено навчання і подальше тестування створених мереж для їх порівнянні з наявними аналогами на різних наборах даних.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Одним з найважливіших етапів розробки інформаційних систем є аналіз вже існуючих рішень, який дозволяє зрозуміти наявний науковий прогрес та ситуацію на ринку у заданій предметній області. В даній роботі розглянуто існуючі рішення для фоторедакторів та для нейронних мереж, що були взяті за основу.

1.1 Ринок фоторедакторів

Згідно з останніми дослідженнями кількість активних користувачів тих чи інших соціальних мереж за останні 5 років збільшилась майже в двічі. Так у 2015 році цей показник дорівнював 2.078 мільярдів активних користувачів, тоді як вже у 2020 цей показник становив 3.96 мільярдів. Це призвело до буму публікації різних зображень користувачами, від власних фото до зображень природи. В одному лише Instagram щодня публікуються більш ніж 100 мільйонів фотографій. Ураховуючи велику конкуренцію, кожен користувач намагається зробити своє фото особливим, щоб воно виділялося на фоні мільйонів інших. Тому постобробка зображень є невід'ємною частиною процеса створення якісного зображення. Вона необхідна для усунення поширених недоліків фотографій, таких як, наприклад:

- низька якість;
- недоекспонування/переекспонування;
- низька контрастність;
- переосвітлення об'єкта зйомки;
- неправильний баланс кольору;
- розфокус.

Для цього на сьогоднішній день у магазинах застосунків Google Play [4] та AppStore [5] можна знайти величезну кількість різних програм для фоторедагування та фотокорекції. Фоторедактори пропонують величезний

спектр інструментів для обробки та редагування зображень. Інструменти можна розділити на чотири основні групи:

1) Покращення якості зображення – найчастіше використовується для старих фото з низькою роздільною якістю, сучасні камери навіть на бюджетних смартфонах дозволяють отримувати зображення високої якості.
2) Накладання масок для створення креативних художніх зображень і нових образів.
3) Редагування основних функцій зображення, таких як налаштування контрастності, тону зображення.
4) Ретушування та корекція зображень.

Так більшість фоторедакторів має відповідний набір інструментів для кожної з цих груп. Але основною проблемою є те, що цей процес є ручною роботою користувача. Так візьмемо за приклад Adobe Photoshop [6]. Це графічний редактор, розроблений компанією Adobe Systems. Продукт є одним з лідером ринку в галузі комерційних засобів редагування зображень і найвідомішим продуктом фірми Adobe. Усі з вище перерахованих інструментів присутні в даному продукті, але якість роботи напряму залежить від знань та навичок користувача у роботі з даним програмним забезпеченням. До того ж вимагає від користувача доволі велику кількість часу на відповідне редагування кожного зображення вручну. Отож для автоматизації цього процесу почали використовувати глибокі нейроні мережі [7], генеративно-змагальні мережі та інші методи машинного навчання [8]. Сучасні сервіси для автокорекції зображень значно полегшують процес ретушування зображень для усіх користувачів, не залежно від їх навичок. Фоторедактори, що побудовані на основі штучного інтелекту виконують ту ж саму роботу, що і ретушер робив би вручну у Adobe Photoshop або подібних продуктах. Процес автоматизації допомагає:

1) Перетворити фото зі звичайної камери, так ніби воно було знято на професійну камеру з високою роздільною здатністю. Так алгоритм

Super-Resolution [9] дозволяє за допомогою алгоритмів глибинного навчання збільшувати розмір зображення у чотири рази при цьому не втрачаючи у якості зображення.

2) Економити час, що було б витрачено на редагування кожного зображення вручну, адже програма буде виконувати цей процес автоматично.
3) Налаштувати автоматичне коригування корекції кольору, тону, насиченості зображень. Виконати ретушування зображення, видалення шуму, використання різних масок або фільтрів.

1.2 Критичний аналіз аналогів

Фоторедактори з використанням змагально-генеративних мереж є відносно молодою областю використання машинного навчання. Попри це, вже існує деяка кількість різних типів нейронних мереж для вирішення відповідно різних задач для покращення якості зображення, наприклад, ретушування, видалення зайвих шумів, збільшення роздільної якості зображення, налаштування балансу кольору, тощо.

Deep Photo Enhancer [10] використовує класичний GAN для покращення якості зображень на основі вивчення фотографій. Нейрони відповідної мережі навчаються знаходити узагальнені характеристики у наборі зразків зображень (колірна гама, яскравість, рівень контрастності) і потім застосовувати отримані знання для покращення зображення так, щоб при цьому зберігався вид оригінального зображення. Але цей метод вимагає використання оригінальних зображень високої якості, що може призвести до великої ресурсозатратності і поганих результатів при роботі з зображеннями низької якості.

Алгоритм Super-Resolution також має GAN структуру з генератором і дискримінатором для збільшення розміру вхідних зображень в два або чотири рази в залежності від мережі, що використовується. Цей процес дозволяє підвищити якість зображень з низькою роздільною здатністю. Генератор

навчається відновлювати фотореалістичні текстури з зображень з дуже низькою роздільною здатністю і відмальовувати їх у збільшеному масштабі. Проте результати ще далекі від 100%, і час від часу мережа створює вкрай неточну реконструкцію зображення. Ще одним вагомим недоліком є висока часозатратність на роботу навіть з зображеннями з низькою роздільною здатністю, вже не кажучи про зображення більших розмірів. Не дозволяє ретушувати зображення, а лише збільшує його розмір.

Fotor – це повноцінний онлайн-фоторедактор [11] доступний як у браузері, так і має власне мобільне застосування. Призначений для професійних фотографів, а також любителів. Дозволяє використовувати набір інструментів та фільтрів, а також автоматичні ефекти для редагування зображення. Містить у собі базові та професійні інструменти для покращення зображення, включаючи різні фільтри та текстури. Є також можливість автоматичного видалення заднього фону зображень і перетворення зображення в стилі класичної живописі, імпресіонізм, фовізму. Оскільки даних продукт є повноцінною системою, більшість інструментів не є безкоштовними, і користувач може отримати до них доступ лише за додаткову плату. Також відсутня можливість автоматичного ретушування зображень для створення рекламних фотографій.

CycleGAN [12] – це підхід до навчання глибоких згорткових мереж для завдань перекладу зображення в зображення. На відміну від більшості GAN мережей, що використовуються для роботи з зображеннями, CycleGAN проходить процес узагальнення знань, вивчаючи зіставлення між одним зображенням та іншим, використовуючи підхід без вчителя. Даний підхід дозволяє замінювати об’єкти на одному зображені відповідними об’єктами інших зображень. Наприклад, дозволяє перетворити фотографію коня на зебру використовуючи той самий фон і позу, при цьому для навчання нам не потрібне відповідне зображення зебри. Проте цей підхід на даний момент не є дуже точним і не підходить для редагування зображень.

pix2pix [13] являє собою комплексний підхід для умовно генеративної змагальної мережі, що використовує U-подібні блоки, їх структура буде описана в наступних розділах. Даний метод не є специфічним для вузької області застосування, його можна застосовувати для широкого кола завдань, включаючи синтез фотографій з карт, створення кольорових фотографій з чорно-білих зображень, перетворення знімків Google Maps на аерофотозображення і навіть перетворення ескізів у фотографії. Даний підхід було взято за основу у нашій мережі для ретушування зображень із невеликою зміною структури. Не має можливості видалення заднього фону, і відсутній графічний інтерфейс для загального використання користувачами з будь-яким рівнем навичок.

У таблиці 1.1 наведено порівняння розробленої системи з наведеними аналогами.

Таблиця 1.1 – Порівняння аналогів

	Наша система	Deep Photo Enhancer	Super-Resolution	Fotor	CycleGAN	Pix2Pix
Повноцінна система з графічним інтерфейсом	+	-	-	+	-	-
Автоматичне видалення заднього фону	+	-	-	+	-	-
Можливість збільшення зображення	+	+	+	-	-	+

Продовження таблиці 1.1

Автоматичне ретушування зображення	+	-	-	+	-	+
Автоматичне налаштування основних параметрів	+	+	-	+	-	-
Перетворення стилю зображення	-	-	-	+	+	+
Перетворення ескізів у фотографії	-	-	-	-	+	+
Робота із зображеннями різних розмірів	+	-	-	+	+	+
Безкоштовна	+	+	+	-	+	+

Отже, відповідно до таблиці порівнянь, розроблена система має можливість автоматичного видалення заднього фону зображення, редагування основних параметрів зображення таких як, яскравість, контрастність, баланс кольору, насиченість, тощо. І на відміну від багатьох з аналогів має графічний інтерфейс користувача, що дозволяє використовувати систему навіть для користувачів без спеціальних знань та навичок. На відміну від Fotor, де більшість інструментів є платними, розроблена система є повністю безкоштовною.

2 ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ

2.1 Мета проведення роботи

Підставою розробки є завдання на дипломне проектування. Темою роботи є «Система для автоматичного усунення фону і ретушування фото на основі генеративно-змагальних глибинних нейронних мереж». Метою дослідження є підвищення якості автоматичного редагування зображень та усунення заднього фону для надання фотографії рекламного виду, тобто високоякісних зображень з якісним виділенням характеристик зображеного об’єкта для привернення додаткової уваги потенційного покупця. Таким чином, завдяки глибинним генеративно-змагальним мережам, що спочатку проходять процес навчання на великій кількості навчальних даних, звичайне зображення з камери смартфону перетворюється на зображення рекламного виду.

2.2 Призначення розробки

Задачею розробки є створення програмної системи, яка представляє собою веб-застосування [14], що служить для автоматичного усунення заднього фону і ретушування відповідних зображень

Система дає користувачу змогу завантажити зображення, отримане з камери смартфона. Наступним кроком система дозволяє обрати один або декілька методів обробки даного зображення. І як результат користувач отримує на вихід відретушоване зображення.

Основною задачею є розробка, навчання і тестування відповідних алгоритмів машинного навчання і подальше створення на їх основі програмної системи, яка представляє собою веб-застосування. Для створення автоматичного редагування зображень нейронні мережі спочатку мають проти процес навчання на великому наборі даних і тестування. Необхідно проаналізувати існуючі структури різних нейронних мереж, їх гіперпараметри. Порівняти їх результати роботи відповідно до цільової задачі. Обрати найкращу з них і зменшити її загальний розмір для підвищення

продуктивності веб-застосування і використання у подальшому для мобільних версій. На основі отриманих результатів потрібно спроектувати та розробити систему, яка буде виконувати описані нижче функції.

2.3 Вимоги до програми

Розроблена програмна система повинна мати наступні функціональні можливості:

- Можливість створювати аккаунт для нового користувача та увійти до уже існуючого.
- Можливість завантажити зображення для подальшого редагування.
- Можливість для користувача обирати необхідні йому способи коригування зображень: автоматичне налаштування яскравості, контрастності, балансу кольору.
- Можливість видалення заднього фона зображень.
- Можливість ретушування зображення.
- Можливість зберегти фінальну версію зображення.

Нефункціональні вимоги – це вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи.

До програмної системи висуваються наступні нефункціональні вимоги:

- Процес налаштування редагування не повинен займати у користувача більше 5 кліків.
- Максимальний розмір завантажувального фото 1024 * 1024.
- Користувач не повинен спостерігати помітних затримок під час редагування зображення. Максимальний час на виконання будь-якої функції не має перевищувати 25 секунд.

2.4 Вимоги до програмної документації

В документації до системи повинна міститься наступна інформація:

- специфікація функціональних і нефункціональних вимог до системи;

- опис можливих параметрів редагування;
- опис формату вхідного і вихідного зображень;
- опис архітектури програмної системи;
- опис графічного інтерфейсу програмної системи.

2.5 Технічно-економічні показники

Розроблена програмна система може бути застосована будь-яким користувачем в залежності від його цілей. Окремі модулі програмної системи можуть бути повторно використані у схожих застосуваннях, а метод, що було взято за основу, може бути застосований до різних предметних областей. Мережі розроблені, пройшли процес навчання і тестування використовуючи мову програмування Python [15] і бібліотеки машинного навчання Tensorflow [16] та Torch [17] в середовищі Google Colab. А веб-застосування було відповідно створено з використанням JavaScript [18]. Навчані моделі було імпортовано до середи JS за допомогою бібліотеки машинного навчання TensorflowJS.

2.6 Технічні вимоги

До застосування висунуті наступні технічні вимоги:

- Веб-застосування повинно коректно працювати в браузерах Google Chrome, Opera, Mozilla Firefox.
- Об’єм пам’яті для моделі не має перевищувати 20 Мб.

2.7 Стадії та етапи розробки програмної системи

Стадії та етапи розробки, а також заплановані сроки реалізації програмної системи наведено у таблиці 2.1.

Таблиця 2.1 – Стадії та етапи розробки

Стадія	Етап	Зміст робіт	Строки
Огляд існуючих рішень	Вивчення існуючих рішень автоматичного ретушування зображень	Вивчення функціональних можливостей та особливостей уже наявних програмних рішень.	15.09.2021 – 25.09. 2021
Технічне завдання	Визначення наявних структур для нейронної мережі	Опис наявних різних видів нейронних мереж в залежності від їх структури	19.09.2021 – 26.09.2021
	Створення специфікації вимог до програмної системи.	Складання документації щодо функціональних і нефункціональних вимог до програмної системи.	15.09.2021– 20.10.2021
Технічний проект	Визначення методики для автоматизації усунення заднього фону зображення	Навчання, тестування і порівняння різних архітектур нейронної мережі для усунення заднього фону. Визначення найкращої.	5.10.2021 – 15.10.2021

Продовження таблиці 2.1

Технічний проект	Визначення методики для автоматичного ретушування зображення	Навчання, тестування і порівняння різних архітектур нейронної мережі для ретушування зображень. Визначення найкращої.	16.10.2021 – 25.10.2021
	Проектування програмної системи	Проектування та створення діаграми Класів	20.10.2021 – 23.10.2021
Робочий проект	Реалізація програмної системи	Написання коду програмної системи, завершення програмної документації	01.11.2021 – 27.11.2021
Готовий проект	Тестування системи	Тестування системи в цілому і окремо частин, аналіз результатів	27.11.2021 – 01.12.2021

3 РОЗРОБКА МОДЕЛЕЙ АВТОМАТИЧНОГО ВИДАЛЕННЯ ЗАДНЬОГО ФОНУ І РЕТУШУВАННЯ ЗОБРАЖЕНЬ

В цьому розділі представлено розробку методів для автоматичного видалення заднього фону зображень і його ретушування, що використовується у веб-застосуванні.

Важливими етапами подальшого дослідження є збір інформації, аналіз наявних мереж для вирішення даних задач, вивчення їх структур, налаштування і порівняння їх результатів роботи відповідно до поставлених задач на навчальному і тестовому наборі даних.

3.1 Генеративно-змагальні мережі (GAN), структура та робота

3.1.1 Згорткові нейроні мережі

Згорткова нейронна мережа (CNN) [19] — це тип нейронної мережі, яка спеціалізується на обробці та класифікації зображень. По суті, мережа просто приймає значення пікселів зображення у векторній/матричній формі як вхідні дані, пропускає його через послідовність шарів і виводить результати в залежності від задачі.

Шари згорткових нейронних мереж зазвичай складаються з чотирьох типів шарів:

1) Згортковий шар фіксує шаблони в зображенні, пропускаючи його репрезентативну матрицю через набір фільтрів/ядер, які представляють різні візуальні особливості зображення. Ці фільтри ковзають по зображенню на основі заданого розміру вікна пікселів та кроків. Під час цих згорток кожен фільтр створює свою власну карту ознак. Кінцевим результатом цього шару є перетворення вихідного зображення, яке складається з усіх карт об’єктів, накладених один на одного.

Двовимірна згортка (2D згортка) - це досить проста операція: починаємо з ядра, що представляє себе матрицю вагів. Ядро "ковзає" над двовимірним зображенням, поелементно виконує операцію множення з тією частиною вхідних даних, над якою воно зараз знаходиться, а потім підсумовує всі отримані значення в один вихідний піксель. Ядро повторює це з кожною локацією, перед якою воно «ковзає», перетворюючи двовимірну матрицю на іншу двовимірну матрицю ознак.

2) Rectified Linear Unit Layer, або ReLU [20], є нелінійною функцією активації и визначається за формулою f(x) = max(x,0). Не змінюючи своєї форми, ReLU перетворює елементи виводу згорткового шару в діапазон від 0 до нескінченності, замінюючи будь-які негативні значення на 0.
3) Шар об’єднання (Pooling) виконує функцію зниження дискретизації вздовж просторових розмірів зображення, зменшуючи розмір представлення зображення. Зменшуючи розмір вхідного зображення, модель підвищує свою обчислювальну ефективність, зберігаючи при цьому більшість визначальних ознак зображень. Однак це також може призвести до перенавчання мережі.
4) Повнозв'язний шар. У той час як згортковий шар створює локальні з’єднання, де одному нейрону відповідає деяка кількість нейронів шару попереднього, у повністю з’єднаному шарі кожен вузол з’єднаний з усіма вузлами попереднього шару, що підвищує якість роботи, але при цьому значно збільшує кількість необхідних обчислень. Використовується в нижніх шарах мережей.

3.1.2 Генеративно-змагальні мережі (GAN)

Генеративні змагальні мережі, або скорочено GAN, є підходом до генеративного моделювання з використанням методів глибокого навчання, таких як згорткові нейронні мережі.

Генеративне моделювання передбачає автоматичне виявлення та вивчення закономірностей у вхідних даних таким чином, що модель може бути використана для генерування або виведення нових прикладів, які, ймовірно, можна було б взяти з вихідного набору даних.

GAN – це розумний спосіб навчання генеративної моделі, формулюючи проблему як проблему навчання за допомогою двох підмоделей: моделі генератора, яку ми навчаємо для створення нових згенерованих прикладів, і моделі дискримінатора, яка намагається класифікувати приклади як реальні або підробні зразки (згенеровані мережею). Дві моделі навчаються разом у змагальній грі з нульовою сумою, поки модель дискримінатора не буде визначати згенерований зразок як реальний приблизно в половині випадках, що означає, що модель генератора генерує дійсно правдоподібні приклади, і модель дискримінатора не може відрізнити їх від оригінальних зображень.

GAN є захоплюючою та швидко змінною сферою, яка забезпечує обіцянку генеративних моделей у своїй здатності генерувати реалістичні приклади в ряді проблемних областей, особливо в завданнях перекладу зображення в зображення, таких як переклад фотографій літа на зиму або дня. до ночі, а також при створенні фотореалістичних фотографій об’єктів, сцен і людей, які навіть люди не можуть розпізнати, чи є вони синтетично підробленими чи ні.

Модель генератора (рис 3.1) приймає випадковий вектор фіксованої довжини як вхідні дані та генерує вибірку в області. Початковий вектор задається випадковим чином із гауссового розподілу, і вектор використовується для початку генеративного процесу, постійно змінюючи свої значення під час навчання. Після навчання точки в цьому

багатовимірному векторному просторі відповідатимуть точкам у проблемній області, утворюючи стиснене представлення розподілу даних.

Цей векторний простір називають латентним простором або векторним простором, що складається з прихованих змінних. Приховані змінні — це ті змінні, які важливі для мережі, але не піддаються безпосередньому спостереженню.

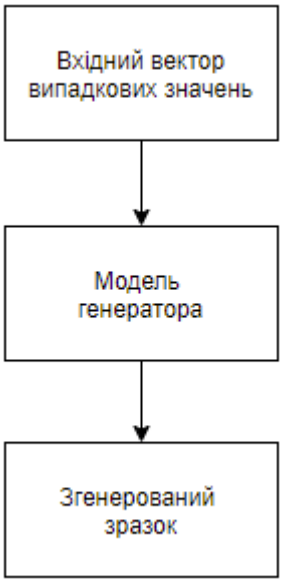

Рис 3.1 Робота моделі генератора

Модель дискримінатора отримує на вхід зразки як реальних, так і згенерованих зображень і передбачає бінарну мітку класу реального чи підробленого (згенерованого). Справжній приклад походить із навчального набору даних. Згенеровані приклади виводяться моделлю генератора. Дискримінатор є звичайною моделлю класифікації. Принцип роботи моделі дискримінатора представлений на рисунку 3.2. Після процесу навчання та тестування модель дискримінатора відкидається, оскільки дискримінатор є лише допоміжною ланкою для створення якісної моделі генератора.

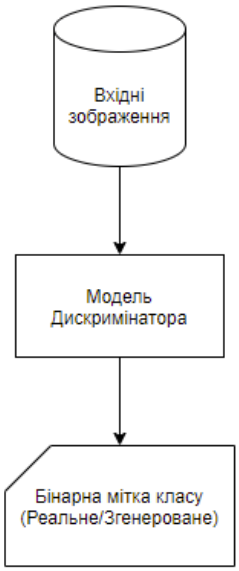

Рис 3.2 Робота моделі дискримінатора

Дві моделі, генератор і дискримінатор, навчаються разом. Генератор генерує набір зразків, які разом із реальними прикладами надаються дискримінатору та класифікуються ним як справжні чи підроблені (рис 3.3).

Потім дискримінатор оновлюється, щоб краще розрізняти справжні та підроблені зразки в наступному раунді, і, що важливо, генератор оновлюється залежно від того, наскільки добре згенеровані зразки визначаються дискримінатором.

Таким чином, обидві моделі конкурують одна з одною, вони є змагальними в сенсі теорії ігор [21] і грають у гру з нульовою сумою.

У цьому випадку нульова сума означає, що коли дискримінатор успішно ідентифікує справжні та підроблені зразки, він винагороджується, і відповідно не потрібно змінювати параметри моделі, тоді як генератор штрафується великими оновленнями параметрів моделі.

Крім того, коли дискримінатор визначає згенерований зразок як реальний, генератор винагороджується і не потрібно змінювати параметри його моделі, при цьому дискримінатор штрафується, а його параметри моделі оновлюються.

На певній межі генератор щоразу генерує ідеальні зразки з вхідного екземпляру, і дискримінатор не може розрізнити різницю і прогнозує «невпевненість» (наприклад, 50% для справжнього та підробленого) у кожному випадку. Це лише приклад ідеалізованого випадку; нам не потрібно доходити до цього моменту, щоб отримати корисну модель генератора.

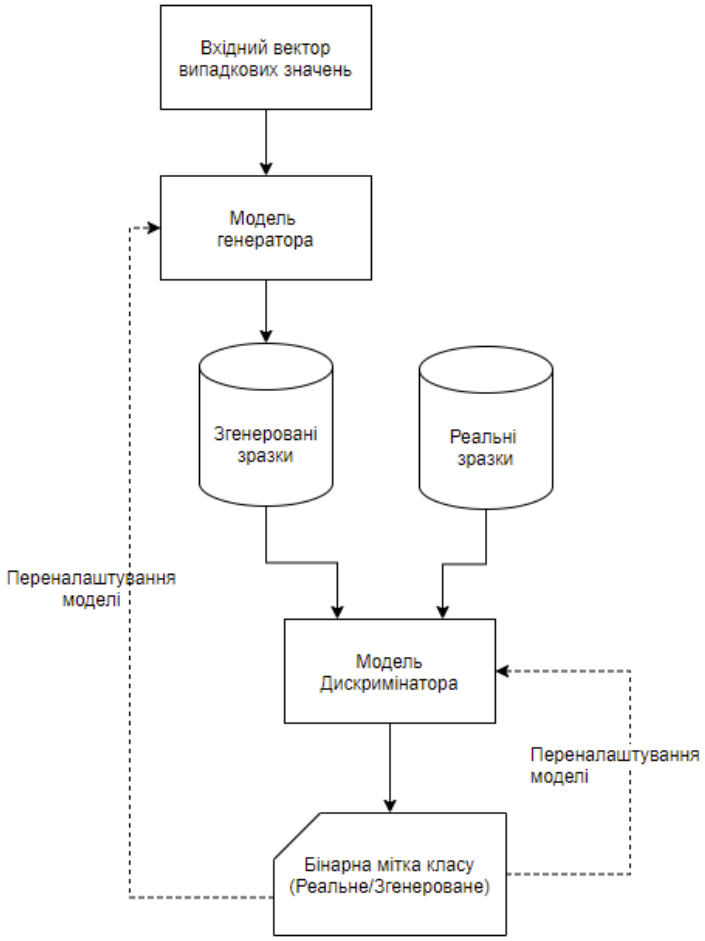

Рис 3.3 Етапи роботи генеративно-змагальних мереж

Важливим розширенням GAN є їх використання для умовного генерування виводу.

Генеративну модель можна навчити генерувати нові приклади з вхідної області, де вхід, випадковий вектор із деякого простору, надається (обумовлений) деякими додатковими вхідними даних. Додатковим введенням може бути значення класу, наприклад, чоловічий або жіночий при створенні фотографій людей, або цифра у випадку створення зображень рукописних цифр.

Генеративні змагальні мережі можна розширити до умовної моделі [22], якщо і генератор, і дискримінатор обумовлені деякою додатковою інформацією щодо зразків. Це може бути будь-яким видом допоміжної інформації, наприклад мітки класів або дані з інших модальностей.

Дискримінатор також є умовним, що означає, що йому надається як вхідне зображення, яке є справжнім або підробленим, так і додатковий вхід. У випадку умовного введення типу мітки класифікації дискримінатор очікував би, що вхід буде відповідним класом, у свою чергу навчаючи генератор генерувати приклади цього класу, щоб дискримінатор визнавав їх як оригінальні зображення. Таким чином, умовний GAN можна використовувати для створення прикладів із зразків заданого типу.

Якщо зробити ще один крок далі, моделі GAN можуть бути обумовлені прикладом, таким як зображення. Це дозволяє застосовувати GAN, наприклад для перекладу тексту в зображення або перекладу зображення в зображення. Це дозволяє використовувати деякі з найбільш вражаючих застосувань GAN, такі як передача стилю, розфарбовування фотографій, перетворення фотографій з літа на зиму або з дня на ніч тощо.

У разі умовних GAN для трансляції зображення в зображення, наприклад перетворення зображення дня в ніч, дискримінатор надає приклади реальних і згенерованих нічних фотографій, а також (зумовлені) реальні денні фотографії як вхідні дані. Генератор починає свою роботу із випадковим вектором із прихованого простору, а також реальними денними фотографіями в якості вхідних даних.

Одним із багатьох основних досягнень у використанні методів глибокого навчання в таких областях, як комп’ютерний зір, є техніка, яка називається збільшенням даних, або аугментація [23].

Збільшення даних призводить до кращої продуктивності моделей, як покращення навичок моделювання, так і надання ефекту регуляризації, зменшення помилки узагальнення. Він працює шляхом створення нових,

штучних, але правдоподібних прикладів із області вхідної проблеми, на якій навчається модель.

Методи є примітивними у випадку даних зображень, що включають кадрування, перевертання, масштабування та інші прості перетворення наявних зображень у наборі навчальних даних.

Успішне генеративне моделювання забезпечує альтернативний і потенційно більш специфічний підхід для збільшення даних. У складних галузях або галузях з обмеженою кількістю даних генеративне моделювання забезпечує шлях до більшої підготовки до моделювання. GAN досягли значного успіху в цьому випадку використання в таких областях, як глибоке навчання з підкріпленням.

Існує багато причин, чому GAN є цікавими, важливими та потребують подальшого вивчення. Ян Гудфеллоу викладає деякі з них у своїй основній доповіді на конференції 2016 року та пов’язаному з ними технічному звіті під назвою «Підручник NIPS 2016: генеративні змагальні мережі» [24].

Серед цих причин він підкреслює успішну здатність GAN моделювати дані великого розміру, обробляти відсутні дані, а також здатність GAN надавати мультимодальні результати або кілька правдоподібних відповідей. Можливо, найбільш переконливе застосування GAN — це умовні GAN для завдань, які вимагають створення нових прикладів. Тут Гудфеллоу вказує три основні приклади:

1) Супер-роздільна здатність зображення. Можливість генерувати версії вхідних зображень з високою роздільною здатністю.
2) Створення мистецтва. Здатність до чудових нових і художніх зображень, ескізів, живопису тощо.
3) Переклад зображення в зображення. Можливість перекладати фотографії з одного напряму в інший, наприклад день у ніч, літо на зиму тощо.

Можливо, найбільш вагома причина того, що GAN мережі широко вивчаються, розробляються та використовуються, полягає в їх успіху. GAN

змогли створювати фотографії настільки реалістичні, що люди не можуть сказати, що на них зображені об’єкти, сцени та люди, яких не існує в реальному житті.

3.2 Використання змагально-генеративних мереж для очищення заднього фону зображення

Після аналізу наявних структур GAN мереж для видалення заднього фону зображення були виявлені три основні, що могли використовуватися за основу створеної мережі. Для їх порівняння були обрані заявлені результати відповідно до офіційної документації мережі за метрикою MAE (Mean Absolute Error) на таких наборах даних як DUT-OMRON, HKU-IS і SOD, адже вони більш за все схожі за своїми екземплярами до того, на чому нам потрібно навчати мережу.

1) FCN – побудована [25], використовуючи повнозв’язні згорткові мережі з використанням структури блоків ResNet-50 або ResNet-101, залежно від необхідної глибини моделі. Загальні результати вражають, так значення MAE дорівнюють 0.056, 0.032, 0.113 для відповідних наборів даних. Але основним недоліком даної моделі є її надто великий розмір. Загальний розмір найменшої моделі складає 571 Мб. Отже ця модель точно не підійде для використання у браузері та подальшому у мобільному додатку.

2) AFNet [26], що побудована використовуючи блоки більш старої структури VGG-16. Розмір цієї мережі вже значно менший від попередньої, усього 143 Мб, що нас влаштовує. Але при цьому значення помилки зросло до 0.057, 0.036 і 0.119. Отже якість впала на усіх наборах даних відносно першої моделі.

3) U2-Net – [27] використовує блоки власної структури під назвою RSU. Розмір моделі складає 176 МБ, що є трохи більшим ніж мережа AFNet, але при цьому демонструє кращі результати не лише за них, а й за більш об’ємну мережу FCN – 0.054, 0.031, 0.108.

Після порівнянь даних змагально-генеративних мереж та їх результатів роботи на різних наборах даних було прийнято рішення, що основою мережі для вирішення проблеми очищення заднього фону зображення буде структура мережі U2-Net. Структура мережі має такі переваги:

1) Мережа здатна отримувати більше контекстної інформації з різних масштабів згортки завдяки суміші сприйнятливих полів різного розміру, що використовуються в ReSidual U-блоках (RSU).
2) Збільшує глибину всієї архітектури без істотного збільшення кількості обчислювальних витрат через операції об'єднання, що використовуються в цих блоках RSU. Ця архітектура дозволяє тренувати глибоку мережу з нуля без використання попередньо натренованих мереж із завдань класифікації зображень.

3.2.1 Залишкові U-подібні блоки

Як локальна, так і глобальна контекстна інформація є дуже важливою для виявлення помітних об'єктів на зображені та вирішення інших завдань сегментації. У сучасних дизайнах глибоких згорткових мереж, таких як VGG, ResNet, DenseNet і так далі, невеликі згорткові фільтри з розміром 1×1 або 3×3 є найбільш часто використовуваними компонентами для вилучення ознак. Оскільки вони вимагають менше місця для зберігання і є обчислювальною ефективними. На рисунку 3.4 можна побачити структуру згорткового RSU блока, що використовується в мережі U2-net.

Вихідні карти об’єктів неглибоких шарів містять лише локальні ознаки зображень, оскільки сприйнятливі фільтри мають розмір 1×1 або 3×3, вони є занадто малими, щоб охопити глобальну інформацію із зображення. Основна ідея полягає в тому, щоб збільшити розмір поля згортки, щоб отримати більше глобальної інформації на високому рівні зображення.

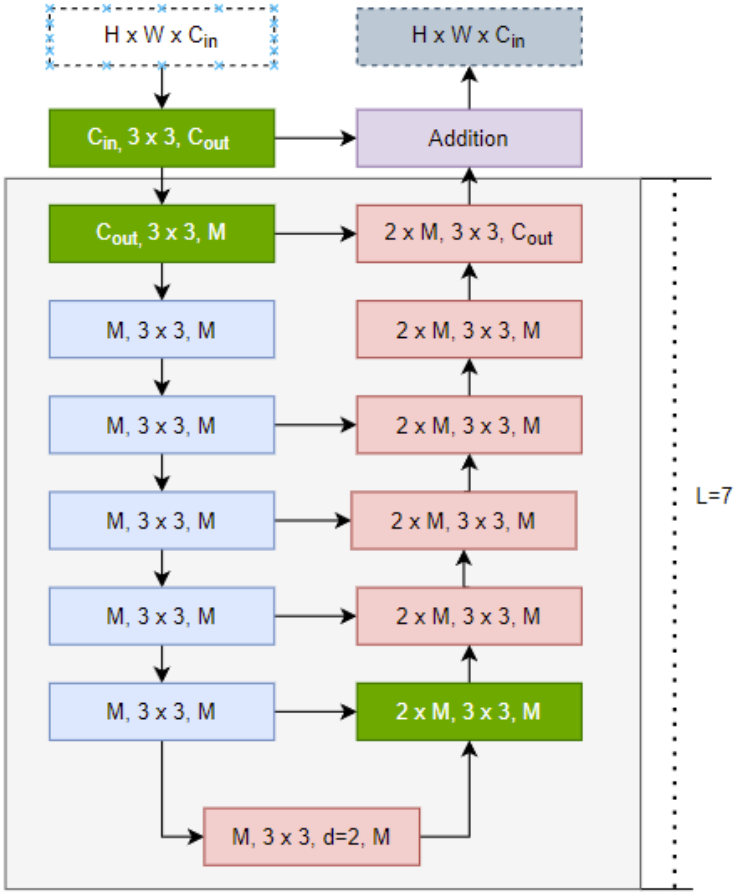

Рис 3.4 - Ілюстрація залишкового U-блок RSU

Однак проведення багаторазових збільшень згортки на вхідній карті об’єктів (особливо в ранній стадії) з оригінальною роздільною здатністю вимагає занадто багато обчислювальних ресурсів та пам'яті.

Тому в структурі U2-Net пропонується використовувати новий ReSidual Ublock (RSU) для захоплення внутрішньоетапних багатомасштабних функцій. Структура RSU-L (Cin, M, Cout) показана на рис. 3.4, де L це - загальна кількість шарів у мережі, Cin, Cout позначають вхідний і вихідний канали, а M позначає кількість фільтрів, що використовуються у внутрішніх шарах RSU. Отже, наша RSU в основному складається з трьох компонентів:

1) Вхідний шар згортки, який перетворює вхідні дані карти об'єктів (H×W×Cin) на проміжну карту F1(x) з каналом Cout. Це звичайний згортковий шар для виділення локальних ознак зображення.

2) U-Net як симетрична структура кодера-декодера або ж генератора-дискримінатора, тобто кількість блоків, що відповідають за поступове зменшення зображення (кодер) для отримання більш глобальних ознак дорівнює кількості блоків, що використовуються для розгортування зображення до початкових розмірів (декодер), з висотою L, яка приймає проміжну карту ознак $F_1(x)$ як вхідні дані та навчається витягувати та кодувати багатомасштабну контекстну інформацію $U(F_1(x))$. Збільшення значення L веде до глибшого залишкового U-блоку (RSU), буде більше операцій, більший діапазон згорток і відповідно будуть виявлені багатші локальні та глобальні особливості. Налаштування цього параметра дозволяє витягувати багатомасштабні об’єкти з вхідних карт об’єктів із довільною просторовою роздільною здатністю. Багатомасштабні об’єкти витягуються з карт об’єктів, що поступово зменшуються, і кодуються у карти об’єктів високої роздільної здатності шляхом прогресивного підвищення дискретизації, конкатенації та згортки. Цей процес дозволяє зменшити значення функції втрати, що викликано прямим підвищенням дискретизації з великими масштабами.
3) Залишкове з'єднання, яке об'єднує місцеві особливості та багатомасштабні ознаки шляхом підсумовування: $F_1(x) + U(F_1(x))$.

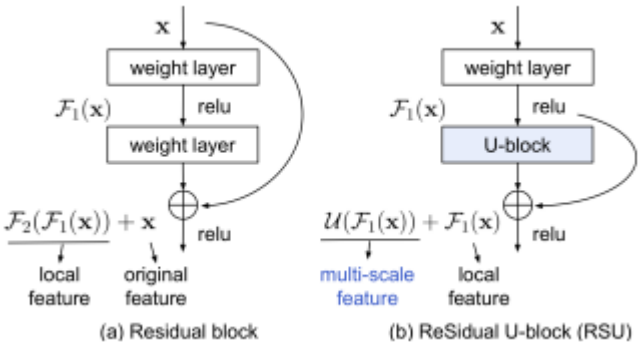

Рис 3.5 Відмінності залишкових блоків Residual Block та Residual U-Block - https://arxiv.org/pdf/2005.09007.pdf

Основна конструктивна відмінність RSU блоку від залишкового блоку відображена на рисунку 3.5 і полягає в тому, що RSU замінює звичайну

однопотокову згортку на структуру U-Net, і замінює оригінальний вихідний об’єкт локальним об’єктом, трансформованим за допомогою вагового шару: HRSU (x) = U(F_1(x)) + F_1(x), де U являє собою багатошарову U-структуру. Ця зміна дизайну дає можливість мережі витягувати функції кількох масштабів безпосередньо з кожного залишкового блоку. Зокрема, важливим є те, що накладні витрати на обчислення U-структури невеликі, оскільки більшість операцій застосовуються до карт об’єктів зі зниженою дискретизацією.

3.2.2 Структура мережі для очищення заднього фону

Укладання кількох RSU блоків для різних завдань розглядалися уже деякий час, ці методи зазвичай складаються з структур, подібних до RSU, використаних послідовно для побудови каскадних моделей, і може бути узагальнено як "(RSU * n)", де n - кількість повторюваних модулів RSU. Питання в тому, що обчислення та витрати на пам'ять збільшуються в n раз. При цьому слід пам’ятати, що внутрішня структура блоків може бути різною. Теоретично показник n може бути встановленим як довільне натуральне число для досягнення однорівневої структури або багаторівневої вкладеної U-структури. Але архітектури с занадто багатою кількістю вкладених рівнів будуть надто складними для реалізації та використання в реальних умовах.

Для вирішення нашої задачі було вирішено побудувати дві моделі однакової структури, але з різною кількість фільтрів у проміжних шарах RSU, а також значно зменшеним значенням вхідного і відповідно вихідного каналів, так для великої моделі значення вхідного каналу (роздільна здатність вхідного зображення) у нижньому шарі (RSU-4F) становить 1024, а для малої мережі усього 128. Це може призвести до втрати дрібномасштабної інформації, але направлене на зменшення загальної ваги мережі, бо модель планується використовувати в майбутньому у мобільному застосуванні. Загальна вага складає 157 Мегабайт, а малої усього 9.8 Мегабайт. Так обидві моделі містять по шість RSU блоків для кодера і відповідно п’ять для декодера зображення (рис 3.6). При зменшені кількості блоків починає втрачатися зчитуєма

інформація з зображення і відповідно зростає значення функції втрат і погіршується згенеровані зображення. В свою чергу збільшення кількості блоків не призводить до суттєвого збільшення розміру і несуттєвого зменшення значення функції втрат.

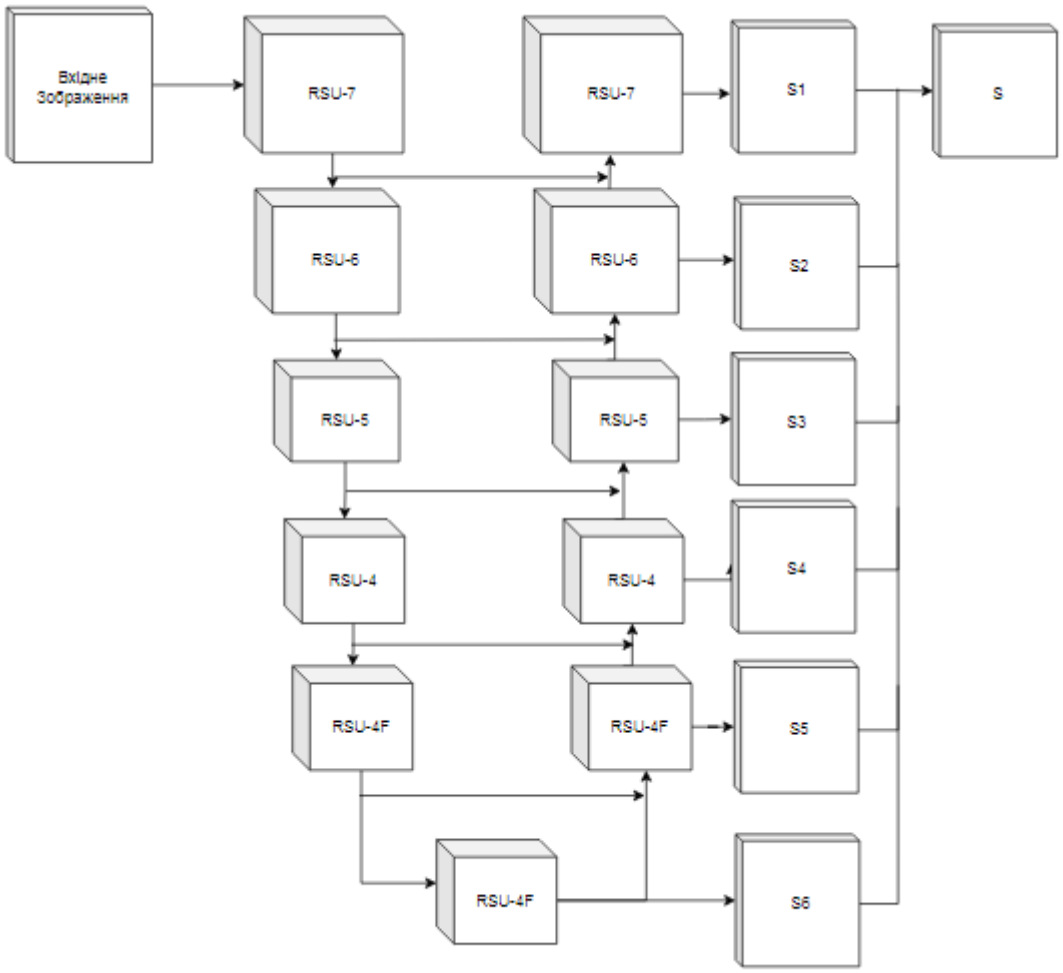

Рис 3.6 - Структура запропонованої мережі

Верхній рівень кодера починається з блоку RSU7, що складається з одинадцяти залишкових U-блоків, далі ідуть відповідно блоки RSU-6, RSU-5 і RSU-4. В свою чергу цифри - «7», «6», «5» і «4» позначають висоту (L) для блока RSU (рис 3.5). Для карт об’єктів з великою висотою та шириною ми використовуємо більше значення L для захоплення великомасштабної інформації. Роздільна здатність карт об’єктів для двох останніх блоків кодера відносно низька, тобто подальше зниження дискретизації цих карт призводить

до втрати корисного контексту зображення. Тому на цих етапах використовуються блоки RSU-4F, де літера «F» означає, що RSU є розширеною версією, в якій ми замінюємо операції об’єднання та підвищення дискретизації на розширені згортки. Це означає, що усі проміжні карти ознак RSU-4F мають однакову роздільну здатність з його вхідними картами об'єктів, не змінюється його розмір. Аналогічно до кодера, декодер має такі ж самі блоки RSU-7, RSU-6, RSU-5, RSU-4 і один блок RSU- 4F.

Мережа генерує шість виходів S6, S5, S4, S3, S2, S1 відповідно до кількості стадій, попередньо використавши шар згортки розміром 3 × 3 і сигмоїдної функції активації. Потім накладаються один на одний за допомогою функції конкатенації.

Підсумовуючи, дизайн мережі дозволяє мати глибоку архітектуру з великою кількістю багатомасштабних фільтрів та відносно низькими витратами на обчислення та затратами пам’яті. В додаток, мережа побудована лише на RSU блоці без використання будь-яких попередньо підготовлених і адаптованих блоків з класифікації зображень, мережа є гнучкою і простою в адаптації до різних робочих середовищ з незначною втратою у продуктивності.

3.2.3 Опис експериментів по очищенню заднього фону зображень

Як уже було сказано раніше, тренувальні дані складають із себе набор із пар зображень, де вхідне зображення є повноцінною фотографією, а вихідне є чорно-білою маскою для виділення об’єкту (рис 3.7). Зображення має містити чітко визначений центральний об’єкту, контур якого необхідно виявити.

Оскільки мережа генерує шість виходів, то функція втрат мережі визначається за формулою 3.1:

$$\boldsymbol{L = \sum_{m=1}^{M} w_{side}^{(m)} l_{side}^{(m)} + w_{fuse} l_{fuse}} \qquad (3.1)$$

де $l^{(m)}_{side}$ - функція втрат бічної вихідної карти $S^{(m)}_{side}$, а l_{fuse} – це втрата вихідного кінцевого результату; карти S. w_{side} і w_{fuse} — це ваги кожної функції втрат відповідно, M – кількість бічних вихідних карт.

Рисунок 3.7 Пара навчальних зображень

Для кожної вихідної карти ми використовуємо стандартну бінарну кросентропію для обчислення втрат. В процесі навчання мережа намагається мінімізувати загальні втрати L.

Оптимізатор був обраний – Adam [28], де перші 100 епох навчання відбувалися із значенням параметра “learning rate”, що дорівнює 0.0001, а після продовжили навчання зі значенням у десять разів нижче, для більш тонкого налаштування мережі.

Результати роботи мережі по закінченню циклу з трьохсот епох: для великої моделі (розмір 157 Мегабайт) значення втрат – 0.009145 для навчальної вибірки і 0.009196 для тестової, для малої мережі U2-Net (9.8 МБ) – відповідно 0.011567 на навчальному наборі даних і 0.012625 на тестовому.

3.3 Мережа для ретушування зображень

Запропоновано умовно змагальну структуру для створення рекламних відретушованих зображень високої роздільної здатності із фотографій

невисокої якості, в тому числі, створених за допомогою камери смартфону. В цьому розділі спочатку розглянемо метод pix2pix, що буде використана за основу розробленої мережі. А потім буде описано, як було підвищено фотореалізм і роздільну здатність зображень, що отримаємо в результаті за допомогою покращеної цільової функції втрат та структури мережі.

3.3.1 Метод pix2pix

Метод pix2pix є умовною структурою GAN для трансляції зображення в зображення. Він складається з генератора G і дискримінатора D. Для нашого завдання мета генератора G перекладати карти семантичних міток для створення реалістично виглядаючих зображень, тоді як дискримінатор D прагне відрізняти реальні зображення від згенерованих. Навчальний набір даних задається як набір пар відповідних зображень, оригінальних і відретушованих. Умовні GAN мають на меті модель умовного розподілу реальних зображень з урахуванням карти семантичних міток.

Метод pix2pix використовує U-Net, що була розглянута в попередньому розділі, як генератор, і глибоку згорткову мережа, що постійно змінює власні ваги при навчанні, як дискримінатор. Вхідні дані для дискримінатора є поканальною конкатенацією карти семантичної мітки та відповідного зображення. Разом з тим мережа дозволяє генерувати зображення з роздільною здатністю розмірами до 1024 × 512.

3.3.2 Модель генератора

Генератор мережі розкладається на дві підмережі: G1 і G2. Ми називаємо G1 глобальною мережею генераторів і G2 - локальним підсилювачем мережі. Тоді генератор задається кортежом G = {G1, G2}, як показано на рисунку 3.8. Глобальна генераторна мережа працює з роздільною здатністю 1024 × 512, а локальна мережа виводить зображення з роздільною здатністю, яка є у 4 рази більшою за вихідний розмір попереднього шару (2× уздовж кожного виміру зображення).

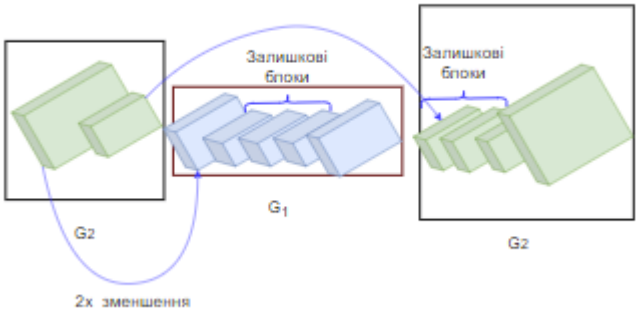

Рис 3.8 Архітектура генератора

Для синтезу зображень на ще вищому рівні можна використовувати додаткові локальні мережі покращення. Наприклад, вихідне зображення генератора G = {G1, G2} становить 2048 × 1024, а вихідне зображення має роздільну здатність G = {G1, G2, G3}, що становить 4096 × 2048. Глобальний генератор побудований на запропонованій архітектурі від Johnson [29], що виявився успішним для передачі нейронного стилю на зображеннях розміром до 512 × 512. Він складається з 3 компонентів:

1) згортковий блок - $G^{(F)}_1$

2) набір залишкових блоків - $G^{(R)}_1$

3) транспонований згортковий блок $G^{(B)}_1$

Семантична карта роздільної здатності 1024×512 передається через 3 компоненти послідовно, щоб вивести зображення з такою самою роздільною здатністю 1024 × 512. Локальна мережа генератора також складається з 3 компонентів: згорткового блоку $G^{(F)}_2$, набору залишкових блоків $G^{(R)}_2$ і транспонованого згорткового блоку $G^{(B)}_2$. Роздільна здатність карти вхідної мітки G2 дорівнює відповідно в два рази більше відносно кожного з вимірів - 2048 × 1024. На відміну від глобальної мережі генераторів, вхід до залишкового блоку $G^{(R)}_2$ є поелементною сумою з двох карт об’єктів: вихідна карта об’єктів $G^{(F)}_2$, і остання карта блоку глобального генератора мережа $G^{(B)}_1$. Це допомагає інтегрувати глобальну інформацію від G1 до G2.

Під час навчання ми спочатку тренуємо глобальний генератор і вже потім починаємо навчання локального підсилювача в порядку їх роздільної

здатності. Потім ми проводимо тонке налаштування всіх мереж разом. Ми використовуємо цю конструкцію генератора для ефективного агрегування глобальної та локальної інформації для синтезу зображення. Зауважимо, що такий конвеєр із багатьма шарами з різною роздільною здатністю є усталеною практикою в комп'ютерному баченні.

3.3.3 Структура дискримінатора мережі

Синтезування багатомасштабних зображень високої роздільної здатності створює значну проблему для дискримінатора GAN. Щоб розрізняти реальні зображення високої роздільної здатності та синтезовані зображення, дискримінатор повинен мати велике поле сприйняття. Для цього знадобиться або більш глибока мережа, або більші згорткові ядра.

Обидва варіанти збільшать пропускну здатність мережі та потенційно можуть призвести до перенавчання. Тобто мережа надто підлаштовується під навчальні зображення і показує погані результати на тестових, яких мережа, ще ніколи не бачила. Крім того, обидва варіанти вимагають більшого обсягу пам'яті для навчання, яке вже є дефіцитним ресурсом для створення зображень високої роздільної здатності. Для вирішення цієї проблеми ми пропонуємо використовувати декілька багатомасштабних дискримінаторів. Ми використовуємо 3 дискримінатори, які мають ідентичну структуру мережі, але працюють у різних масштабах зображення. Ми будемо називати дискримінатори D1, D2 і D3.

Зокрема, ми зменшуємо вибірку реальних і синтезованих зображень з високою роздільною здатністю в 2 і 4 рази, щоб створити піраміду зображень, що складається з трьох масштабів. Дискримінатори D1, D2 і D3 відповідно навчаються розрізняти реальні та синтезовані зображення на трьох різних масштабах зображення відповідно. Хоча дискримінатори мають ідентичну архітектуру, той, який діє у найбільш грубому масштабі, має найбільше рецептивне поле. Цей дискримінатор має більш глобальний погляд на

зображення і може спрямовувати генератор для створення глобально узгоджених зображень. З іншого боку, дискримінатор у найменшому масштабі заохочує генератор для отримання більш дрібних деталей. Це також робить навчання генератора від грубого до точного простіше, оскільки розширення моделі з низькою роздільною здатністю до більш високої роздільної здатності вимагає лише додавання дискримінатора найвищого рівні, а не його повне перенавчання з нуля. Без багатомасштабних дискримінаторів ми спостерігаємо, що багато повторюваних візерунків часто з'являється в створеному зображення. Для дискримінаторів тоді проблема навчання стає багатозадачною і визначається за формулою 3.2.

$$\min(G)\max(D_1, D_2, D_3) \sum_{k=1,2,3} L_{GAN}(G, D_k) \quad (3.2)$$

Використання кількох дискримінаторів GAN в одному масштабі зображення було запропоновано в безумовних GAN. Додавання глобального класифікатора зображень до умовних GAN дозволяє синтезувати глобально більш цілісний вміст для зображення. Для цього ми розширюємо структуру до кількох дискримінаторів у різних масштабах зображення для моделювання зображень з високою роздільною здатністю.

3.3.4 Покращуємо якість мережі за допомогою втрат

Ми покращуємо втрати GAN шляхом включення втрати ознак на основі дискримінатору. Ця втрата стабілізує тренування, так як генератор повинен виробляти природне зображення в декількох масштабах. Зокрема, ми витягуємо ознаки з кількох шарів дискримінатора і зіставляємо ці проміжні уявлення реального та синтезованого зображень. Для легкості представлення позначимо і-шаровий екстрактор ознак дискримінатора D_k як $D^{(i)}_k$ (від входу до і-го шару D_k). Потім обчислюється втрата збігу ознак $L_{FM}(G, D_k)$ за формулою 3.3

$$L_{FM}(G, D_k) = E_{(a,x)} \sum_{i=0}^{T} \frac{1}{N} \left(D_k^{(i)}(s, x) - D_k^{(i)}\big(s, G(s)\big)\right) \quad (3.3)$$

де T - загальна кількість шарів, а Ni позначає кількість елементів у кожному шарі. Втрата відповідності ознак дискримінатора GAN пов'язана з втратою сприйняття, яка, як було показано, є корисною для роботи з дуже великими зображеннями та передачі стилю зображення. Відповідні втрати та втрати сприйняття можна спільно використовувати для подальшого вдосконалення процесу виконання дискримінатора.

3.3.5 Результати навчання і тестування

Мережа навчалася на наборі даних, що являють собою пари зображень (рис 3.9), де вхідне зображення є звичайною фотографією, а вихідне цим саме відретушованим екземпляром. Набір даних містить у собі 20000 відповідних пар, 18 000 були використані для навчання і 2000 для тестування.

Рисунок 3.9 Пара навчальних зображень

Оптимізатор був обраний – Adam, на відміну від U2-Net найкращі результати були при сталому значенні параметра “learning rate” - 0.0001. Результати роботи мережі по закінченню циклу з 210 епох: значення втрат для навчальної вибірки – 0.02678 і 0.02713 для тестової.

3.4 Зменшення розміру генеративно-змагальних мереж

Оскільки мережі плануються до використання у веб-застосуванні і у подальшому мають у планах використання в мобільній версії додатку, було прийнято рішення спробувати зменшити кінцевий розмір ваг нейронних мереж для підвищення продуктивності роботи і зменшення загальної ваги додатку. При цьому необхідно максимально зберегти рівень втрат, що має повноцінна версія генеративно-змагальних мереж. До розгляду було обрано дві основні методики, такі як: квантування [30] і дистиляція знань [31].

3.4.1 Квантування

Глибоке навчання для завдань класифікації включає навчання параметрів нейронної мережі таким чином, щоб алгоритм навчився розрізняти класи об’єктів. Це досягається шляхом подачі багатьох зображень мічених даних в нейронну мережу, одночасно оновлюючи параметри для підвищення продуктивності плавної цільової функції. Недоліком є те, що використовується велика кількість параметрів у порівнянні з більш традиційними алгоритмами.

Таким чином, квантування стає методом для приведення нейронної мережі до розумного розміру, а також досягнення високої точності. Це особливо важливо для програм на пристрої, де обсяг пам’яті та кількість обчислень є обмеженими. Квантування для глибокого навчання — це процес апроксимації нейронної мережі, яка використовує числа з плаваючою комою, нейронною мережею чисел з низькою бітовою шириною. Це різко зменшує як вимоги до пам’яті, так і обчислювальні витрати на використання нейронних мереж.

Фундаментальна ідея квантування полягає в тому, що якщо ми перетворюємо ваги та вхідні дані в цілі типи, ми споживаємо менше пам’яті, а на певному обладнанні обчислення відбуваються швидше. Однак є компроміс:

з квантуванням ми можемо втратити значну точність. Тому його необхідно використовувати дуже обережно.

Неможливо просто зберігати в пам’яті числа, лише одиниці та нулі. Отже, щоб правильно зберігати числа та використовувати їх для обчислень, вони повинні бути закодовані. Є два основних представлення: цілі числа і числа з плаваючою комою.

Можна сказати, що використання int8 зазвичай швидше, ніж float32. Однак float32 використовується за замовчуванням для навчання та виводу для нейронних мереж.

Ідея в принципі дуже проста. Припустимо, що є шар з виходами в діапазоні [-a, a), де a — будь-яке дійсне число. Спочатку масштабується вихід до [-128, 128), а потім просто округлюється вниз. Отже, квантування в основному відбувається операційно. Перехід від float32 до int8 не єдиний варіант, є й інші, наприклад, від float32 до float16. Їх також можна комбінувати. Наприклад, можна квантувати множення матриці до int8, а активації — до float16.

Квантування є наближенням. Загалом, чим ближче наближення, тим менше зниження продуктивності можна очікувати. Якщо квантувати усі зміні до float16, це скоротить використання пам’ять вдвічі і, ймовірно, не втратить у точності, але насправді це не призведе до явного прискорення. З іншого боку, квантування за допомогою int8 може призвести до набагато меншого використання пам’яті, але продуктивність, ймовірно, буде гіршою. В екстремальних сценаріях це навіть не спрацює, і може знадобитися навчання з урахуванням квантування.

На практиці існує два основних способи квантування.

1) Після навчання: спочатку відбувається тренування моделі, використовуючи ваги та вхідні дані float32, а уже потім відбувається квантування ваг мережі. Його головна перевага в тому, що це спосіб є простим у застосуванні. Однак значних недоліком є те, що це може призвести до втрати точності.

2) Тренування з урахуванням квантування: процес квантування відбувається під час тренування. Тут навіть градієнти розраховуються для квантованих ваг. Застосовуючи квантування до типу int8 має найкращий результат, але воно є більш складним, ніж попередній варіант.

3.4.2 Дистиляція знань

Нейронні моделі в останні роки були успішними майже в усіх областях, включаючи надзвичайно складні постановки задач. Однак ці моделі мають величезні розміри, з мільйонами (і мільярдами) параметрів, і тому не можуть бути розгорнуті на периферійних пристроях.

Дистиляція знань відноситься до ідеї стиснення моделі шляхом навчання меншої мережі, що саме робити (крок за кроком), використовуючи більшу вже навчену мережу. «М’які мітки» відносяться до вихідних карт функцій більшої мережі після кожного шару згортки. Меншу мережу потім навчають вивчати точну поведінку більшої мережі, намагаючись відтворити її виходи на кожному рівні (а не лише на кінцевому рівні втрат).

Очевидно, що в більш складних моделях теоретичний простір пошуку більше, ніж у меншої мережі. Однак, якщо ми припустимо, що такої ж (або навіть подібної) конвергенції можна досягти за допомогою меншої мережі, тоді простір можливих розв’язків мережі вчителів повинен перекриватися з простором розв’язків мережі учнів.

На жаль, це само по собі не гарантує зближення для студентської мережі в тому самому місці. Мережа учнів може мати конвергенцію, яка може значно відрізнятися від мережі викладачів. Однак, якщо студентська мережа керується поведінкою мережі викладачів (яка вже здійснила пошук у більшому просторі рішень), очікується, що її простір конвергенції перекриватиметься з вихідним простором конвергенції мережі викладачів.

Послідовність роботи:

1) Навчання мережі вчителів: дуже складну мережу викладачів спочатку навчають окремо, використовуючи повний набір даних. Цей крок вимагає високої обчислювальної продуктивності, тому його можна виконувати лише в автономному режимі (на високопродуктивних графічних процесорах).
2) Встановлення відповідності: під час проектування учнівської мережі необхідно встановити відповідність між проміжними результатами учнівської мережі та результатами мережі викладачів. Ця відповідність може передбачати безпосередню передачу виводу рівня з мережі викладача в мережу студента або виконання деякого зближення даних перед передачею їх в мережу учнів.
3) Пряма передача через мережу викладача: передача даних через мережу викладача, щоб отримати всі проміжні результати, а потім застосувати до них розширення даних.
4) Зворотне розповсюдження через студентську мережу: тепер використовуються вихідні дані з мережі викладачів і значення помилки зворотного поширення в мережі учнів, щоб мережа учнів могла навчитися копіювати поведінку мережі викладачів.

3.4.3 Результати зменшення розміру мереж

Для обох мереж були проведені процедури, описані вище. Так для великої мережі (Big), що відповідає за видалення заднього фону зображення, було виконало дистиляцію знань до малої версії цієї мережі (Small), а також квантування після навчання. Для мережі Small було вирішено не проводити зменшення, адже загальний розмір є достатньо низьким для використання в мобільних застосуваннях, а отже подальше зменшення не має сенсу.

Для мережі ретушування і регулювання зображень для створення зображень рекламного виду були проведені як процедура квантування після

навчання, так і дистиляція знань з двома різними показниками параметра T, що визначає силу зв’язку між мережею студента і вчителя.

Виконавши необхідні експерименти для зменшення кінцевого розміру моделей, були отримані наступні результати, що представленні в таблиці 3.1 для мережі для очищення заднього фону зображення і в таблиці 3.2 для мережі для ретушування зображень.

Таблиця 3.1 - Результати квантування і дистиляції знань для мережі для очищення заднього фону

Мережа	Розмір, Мб	Рівень втрат
U2-Net Big	157 Мб	0.009145
U2-Net Small	9.8 Мб	0.011567
U2-Net Big з використанням квантизації після навчання	41.4 Мб	0.009451
Дистиляція знань з U2-Net Big з параметром T = 2	9.7 Мб	0.01676
Дистиляція знань з U2-Net Big з параметром T = 7	9.7 Мб	0.01789

Найкращим варіантом серед зазначених (табл. 3.1) було обрано використання мережі U2-Net Big з квантуванням після навчання, адже вона зменшує загальний розмір мережі більш ніж в 4 рази відносно великої моделі U2-Net, при цьому майже не втрачаючи у якості роботи (значення рівня втрат є досить близьким).

Таблиця 3.2 - Результати квантування і дистиляції знань для мережі для ретушування зображень

Мережа	Розмір, Мб	Рівень втрат
pix2pix	31.8 Мб	0.02678
pix2pix з використанням квантизації після навчання	6.4 Мб	0.03589
Дистиляція знань з параметром T = 2	12.1 Мб	0.033801
Дистиляція знань з параметром T = 7	12.1 Мб	0.03404

Для мережі з ретушуванням зображення було обрано зашити модель pix2pix без усякого загального зменшення, адже при використані методик зменшення загального розміру доволі сильно виріс рівень втрат.

4 ПРОЕКТУВАННЯ СИСТЕМИ

Важливим етапом розробки програмних систем є передпроектний аналіз. Його ключовими завданнями є виявлення різних видів вимог, які має задовольняти розроблювальна програмна система. Формулювання вимог дозволяє створити закінчений опис очікуваної поведінки системи, яку потрібно розробити, та список повністю сформульованих функціональних і нефункціональних вимог до неї. Це дає можливість більш точної оцінки етапів та планів розробки, а також термінів реалізації проекту.

Специфікація вимог програмного забезпечення згідно до IEEE 830 (Software requirements specification) [32] – структурований набір вимог до поведінки програмної системи, що розроблюється. Включає в себе користувальницькі сценарії, які описують всі варіанти поведінки програмної системи під час її взаємодії з користувачами.

Призначені для користувача сценарії є засобом представлення функціональних вимог. Окрім сценаріїв для користувача, специфікація також містить нефункціональні вимоги, які накладають обмеження на дизайн або реалізацію.

4.1 Визначення вимог користувачів

Щоб описати вимоги користувачів до системи, використовується діаграма варіантів використання (рис 4.1). Далі йде відповідний словесний опис усіх прецедентів, що включає у себе опис основного сценарію прецедента, а також його альтернативні варіанти.

Рис 4.1 - Діаграма варіантів використання

Система повинна надавати можливість користувачеві створити новий аккаунт або увійти до вже існуючого. Далі завантажити зображення і провести його редагування: видалення заднього фону, авторетушування, налаштування параметрів і збереження кінцевого результату роботи.

Далі в даному пункті наводиться опис сценаріїв, які були зображені на діаграмі варіантів використання для більш детального їх розгляду, побудови основного і альтернативних сценаріїв.

Варіант використання «Вхід в систему»

Основна діюча особа: користувач.

Передумова: користувач попередньо створив аккаунт.

Гарантії успіху: користувач успішно увійшов до системи.

Основний сценарій:

1. Користувач звертається до системи.
2. Система виводить форму для входу.

3. Користувач вводить логін і пароль.
4. Система підтверджує, що дані введені вірно.
5. Система впускає користувача.

Альтернативний сценарій:

4а. Система не підтверджує, що дані введені вірно:

4а1. Система виводить вікно про те, що введені дані невірні.

4а2. Повернення до кроку 3.

Варіант використання «Завантаження зображення»

Основна діюча особа: користувач.

Передумова: користувач попередньо увійшов в систему.

Гарантії успіху: користувач успішно завантажив зображення для редагування.

Основний сценарій:

1. Користувач звертається до системи і натискає «Завантажити».
2. Система відкриває вікно для завантаження файлів.
3. Користувач вибирає на локальному середовищі зображення, яке прагне завантажити.
4. Система підтверджує, що завантажувальний файл є дійсно зображенням і з розширенням *png або *jpg.
5. Система завантажує зображення для подальшої роботи з ним.

Альтернативний сценарій:

4а. Система не підтверджує, що вибрано коректне зображення:

4а1. Система виводить вікно про помилку під час завантаження.

4а2. Повернення до кроку 3.

Варіант використання «Видалення заднього фону зображення»

Основна діюча особа: користувач.

Передумова: користувач попередньо увійшов в систему.

Гарантії успіху: користувач успішно видалив фон із зображення.

Основний сценарій:

1. Користувач звертається до системи і натискає «Очистити фон».
2. Система підтверджує наявність завантаженого зображення.

3. Система очищає фон зображення і показує користувачу відредаговане зображення.

Альтернативний сценарій:

2а. Система не підтверджує наявність завантаженого зображення:

2а1. Система виводить вікно про те, що зображення ще не завантажено.

2а2. Повернення до варіанту використання «Завантаження зображення».

Варіант використання «Ретушування зображення для надання йому рекламного виду»

Основна діюча особа: користувач.

Передумова: користувач попередньо увійшов в систему.

Гарантії успіху: користувач успішно провів ретушування зображення.

Основний сценарій:

1. Користувач звертається до системи і натискає «Автоматичне ретушування».
2. Система підтверджує наявність завантаженого зображення.
3. Система проводить ретушування вхідного зображення і виводить відповідний результат

Альтернативний сценарій:

2а. Система не підтверджує наявність завантаженого зображення:

2а1. Система виводить вікно про те, що зображення ще не завантажено.

2а2. Повернення до варіанту використання «Завантаження зображення».

Варіант використання «Налаштування параметрів зображення»

Основна діюча особа: користувач.

Передумова: користувач попередньо увійшов в систему.

Гарантії успіху: користувач успішно налаштував параметри зображення.

Основний сценарій:

1. Користувач звертається до системи і переходить до вкладки «Параметри зображення».
2. Система відкриває необхідну форму.

3. Користувач вводить необхідні значення для кожного параметра або вибирає варіант «Автоматично».
4. Система підтверджує наявність завантаженого зображення.
5. Система підтверджує коректність внесених параметрів.
6. Система виводить відредаговане зображення.

Альтернативний сценарій:

4а. Система не підтверджує наявність завантаженого зображення:

4а1. Система виводить вікно про те, що зображення ще не завантажено.

4а2. Повернення до варіанту використання «Завантаження зображення».

5а. Система не підтверджує коректність внесених параметрів.

5а1. Система виводить вікно про некоректні значення параметрів.

5а2. Повернення до кроку 3.

Варіант використання «Збереження результату»

Основна діюча особа: користувач.

Передумова: користувач попередньо увійшов в систему і .

Гарантії успіху: користувач успішно зберіг відредаговане зображення.

Основний сценарій:

1. Користувач звертається до системи і натискає «Зберегти результат».
2. Система підтверджує наявність відредагованого зображення.
3. Система відкриває форму для збереження відредагованого та/або відретушованого зображення.
4. Користувач вводить найменування файлу для збереження.
5. Система зберігає файл.

Альтернативний сценарій:

2а. Система не підтверджує наявність відредагованого зображення:

2а1. Система виводить вікно про те, що відредагованого зображення не має.

2а2. Повернення до варіантів використання «Налаштування параметрів зображення», «Видалення заднього фону зображення», «Ретушування зображення».

4.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якість системи за певними критеріями. Під час формування нефункціональних вимог були визначені такі сценарії якості для розроблювальної системи:

Сценарій «Вхід в систему»:

- Надійність: модуль входу до системи має справно працювати в 99 випадках із 100.
- Продуктивність: коли користувач заходить до системи, час авторизації не має перевищувати 10 секунд.
- Безпека: максимальна кількість спроб доступу з введенням неправильного пароля - 5.
- Зручність використання: для входу у систему користувачу достатньо логіна і пароля.

Сценарій «Завантаження зображення»:

- Надійність: коли користувач завантажує зображення, система повинна у 97 випадках із 100 завантажити його.
- Продуктивність: коли користувач завантажує зображення, час завантаження не має перевищувати 3 хвилини.
- Безпека: максимальний розмір завантаженого зображення не перевищує 35 Мегабайт.
- Зручність використання: коли користувач бажає завантажити зображення, ця процедура повинна зайняти не більше 3-х кліків.

Сценарій «Видалення заднього фону зображення»:

- Надійність: коли користувач обирає використання видалення заднього фону, система повинна у 96 випадках із 100 відобразити результат.
- Продуктивність: час, який витрачається на видалення фону не має перевищувати 2 хвилини.
- Безпека: процедура недоступна під час завантаження зображення.

- Зручність використання: процедура повинна займати не більше 2-х кліків.

Сценарій «Ретушування зображення для надання йому рекламного виду»:

- Надійність: коли користувач обирає використання видалення заднього фону, система повинна у 98 відсотках випадків провести дану процедуру.
- Продуктивність: час, який витрачається на ретушування зображення не має перевищувати 3 хвилини
- Безпека: процедура недоступна під час завантаження зображення або виконання очистки заднього фону.
- Зручність використання: коли користувач бажає провести ретушування зображення, після виконання система виводить отриманий результат.

Сценарій «Налаштування параметрів зображення»:

- Надійність: коли користувач обирає використання видалення заднього фону, система повинна у 99 відсотках випадків провести дану процедуру.
- Продуктивність: час, на роботу кожного з параметрів не має перевищувати 1 хвилини.
- Безпека: максимальна кількість параметрів, що одночасно можна налаштувати – 3.
- Зручність використання: після завершення роботи процедури система показає користувачу отриманий результат.

Сценарій «Збереження результату»:

- Надійність: результат повинен зберігатися у локальному середовищі з відповідним розширенням у 99 випадках із 100.
- Продуктивність: час на зберегання зображення будь-якого розміру не має перевищувати 2 хвилини.
- Безпека: процедура не доступна під час виконання процедур з редагування чи ретушування зображення.
- Зручність використання: після завершення роботи процедури система демонструє вікно про успішне збереження зображення.

4.3 Проектування концептуальних класів системи

На основі діаграми варіантів використання було розроблено діаграму концептуальних класів. Дана діаграма зображена на рисунку 4.2.

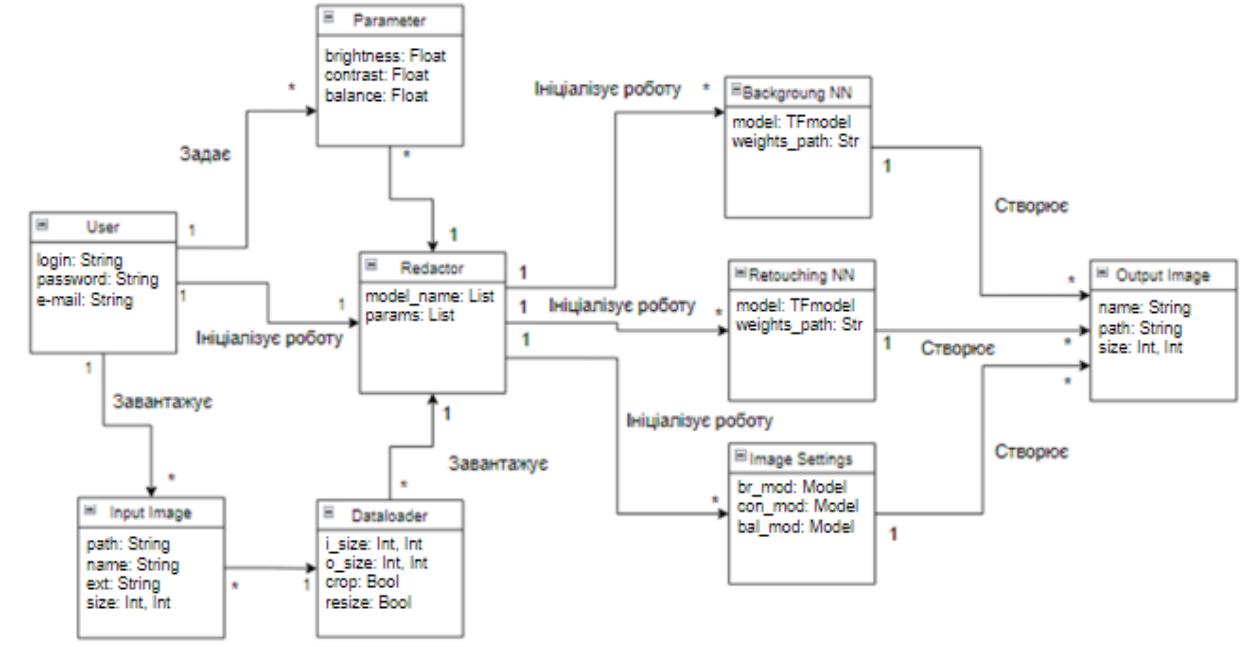

Рис 4.2 – Діаграма концептуальних класів

Клас користувач, на діаграмі «User», завантажує зображення «Input Image», яке прагне відредагувати і ініціалізує початок роботи фоторедактора. Користувач може також попередньо встановити параметри яскравості, контрастності і балансу кольору за власним побажанням або ж залишити їх в автоматичному режимі. Завантажене зображення спочатку перед тим як потрапити до редактора проходить клас «Dataloader», що перетворює відповідне зображення на таке, що приймає на вхід мережа редактора. Редактор складається з трьох частин – модулю, що відповідає за очищення заднього фону зображення, ретушування зображення для надання йому рекламного виду і налаштувань параметрів зображень.

Кожен з цих модулів в кінці власної роботи створює вихідне відредаговане зображення, що відображається користувачу, і може бути збереженим, якщо у цьому є необхідність.

На основі діаграми концептуальних класів була створені діаграми програмних класів, що відображені в наступному підрозділі.

4.4 Діаграма програмних класів

Діаграму програмних класів (рис. 4.3) було побудовано використовуючи архітектуру «Модель-Вид-Контролер» (*Model-View-Controller – MVC*).

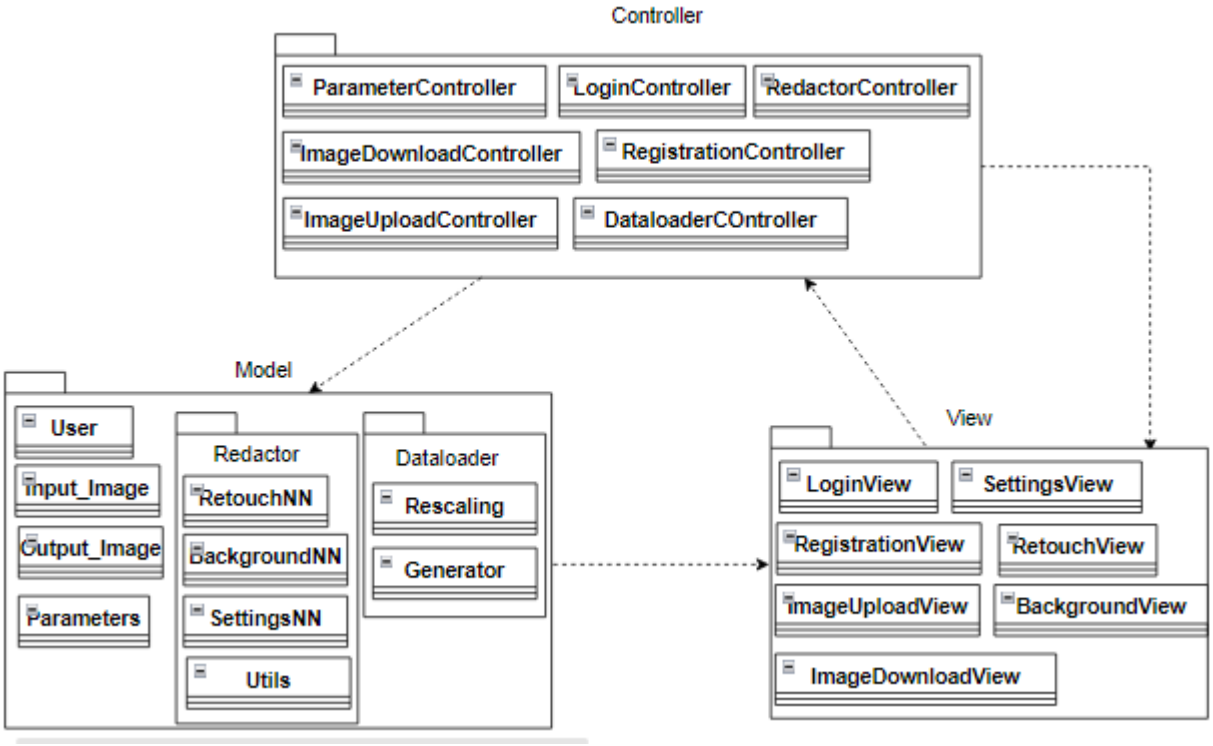

Рис 4.3 – Діаграма пакетів

Оскільки програмний продукт передбачає взаємодію користувача з графічним інтерфейсом системи, то для зручного застосування було прийнято рішення винести усе пов’язане з побудовою графічної частини системи до окремого пакету «View» (рис. 4.4)

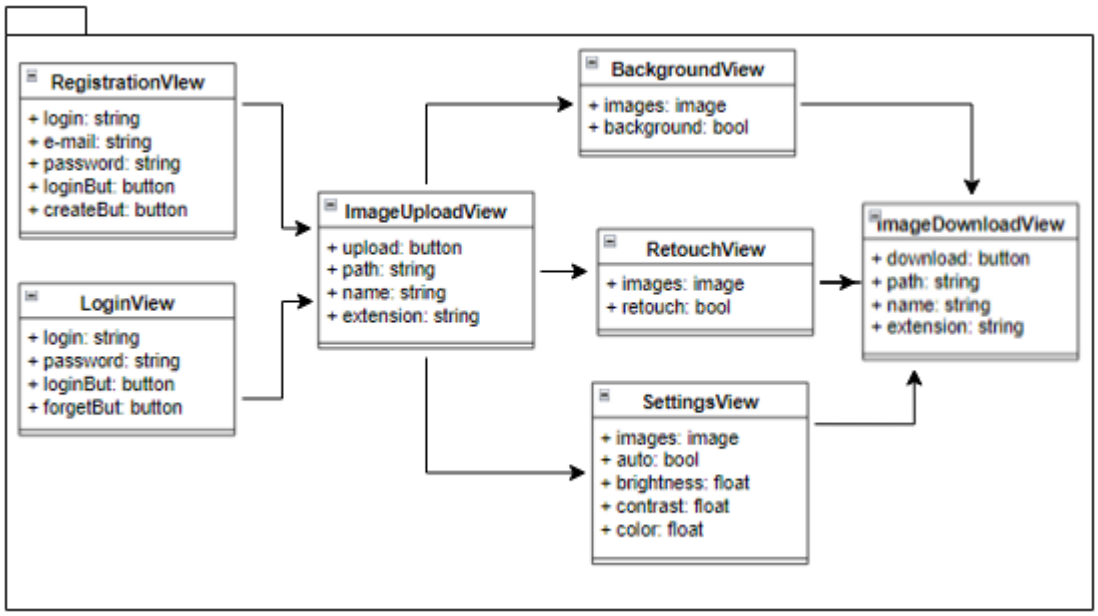

Рис 4.4 – Діаграма класів. Пакет «View»

Основною метою пакету «View» є виведення графічного інтерфейсу для зручної роботи користувача з програмною системою. Так пакет містить класи «ImageUploadView», «ImageDownloadView», «RetouchView», «LoginView», «BackgroundView», «SettingsView», «RegistrationView».

Кожен з класів відповідає за графічне виведення інформації для відповідної вкладки програмної системи. «LoginView» і «RegistrationView» відповідають за форми входу до системи і реєстрації, відображають необхідні поля для заповнення. Класи «BackgroundView», «SettingsView» і «RetouchView» відповідають за інтерфейс вкладок для різних видів редагування вже завантаженого зображення.

Наступним на розгляд потравляє пакет «Model» (рис 4.5). Пакет є ядром системи і відповідає за найрізноманітніші операції, що відбуваються у неї всередині.

Пакет містить у собі класи «User», «Input_Image», «Output_Image», «Parameter», а інші класи були виділені до двох пакетів «Redactor» і «Dataloader». Їх діаграми класів зображені на рисунках 4.6 і 4.7 відповідно.

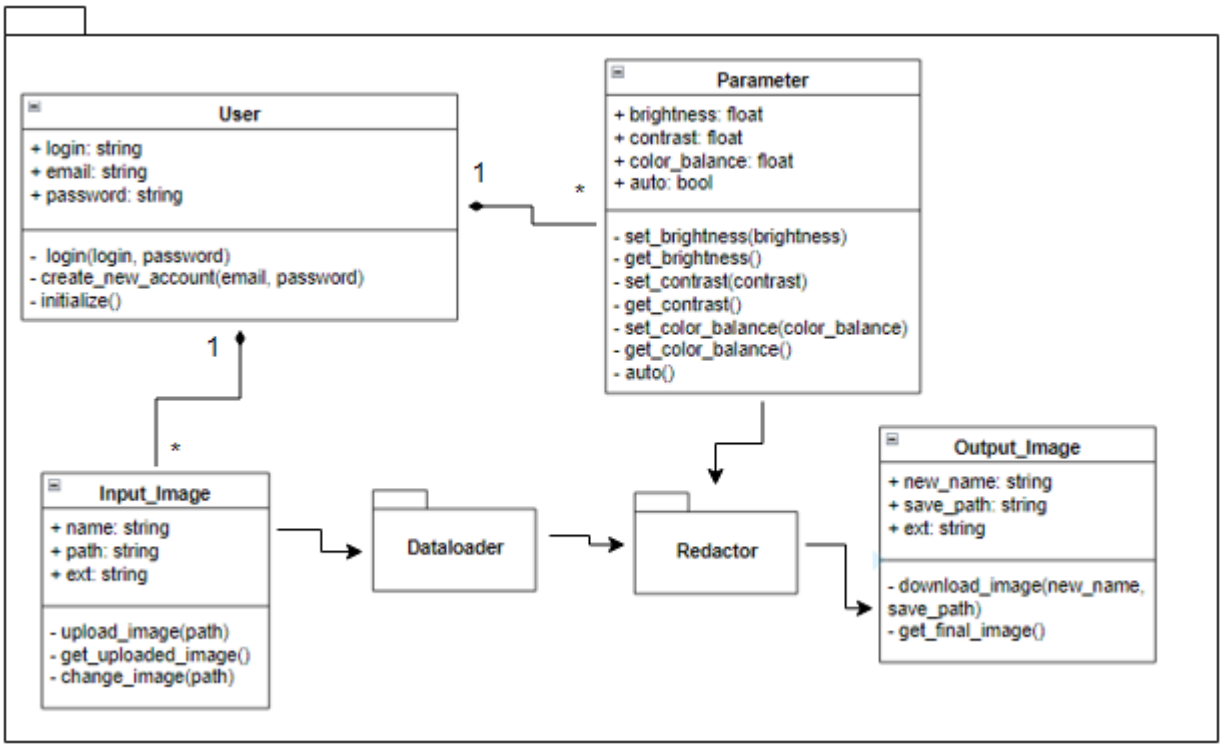

Рис 4.5 – Діаграма класів. Пакет «Model»

Клас «User» зберігає в собі інформацію про користувача системи, а саме його логін, пароль, електрону пошту, а також надає доступ користувачу до ініціалізації роботи з редактором. «Input_Image» і «Output_Image» служать для роботи із зображенням, яке завантажив користувач, і фінальною версією зображення, що вийшло в результаті редагування. Клас «Parameter» в свою чергу несе відповідальність за налаштування параметрів зображення, користувач має можливість задати їх вручну або просто вибрати автоматичний варіант редагування.

Пакет «Redactor» (рис 4.6) відповідає безпосередньо за редагування, ретушування та видалення заднього фону зображення. Кожен клас відповідає за свою мережу, призначену до відповідних цілей і містить параметри мережі, збережені ваги і функцію втрат, що дозволяє виконувати функції редагування зображення. Клас «Utils» призначений для додаткової роботи з зображенням перед наданням його до входу мережі або користувальницького інтерфейсу. Наприклад, перетворення кольорового зображення в його чорно-білу або сіру версію.

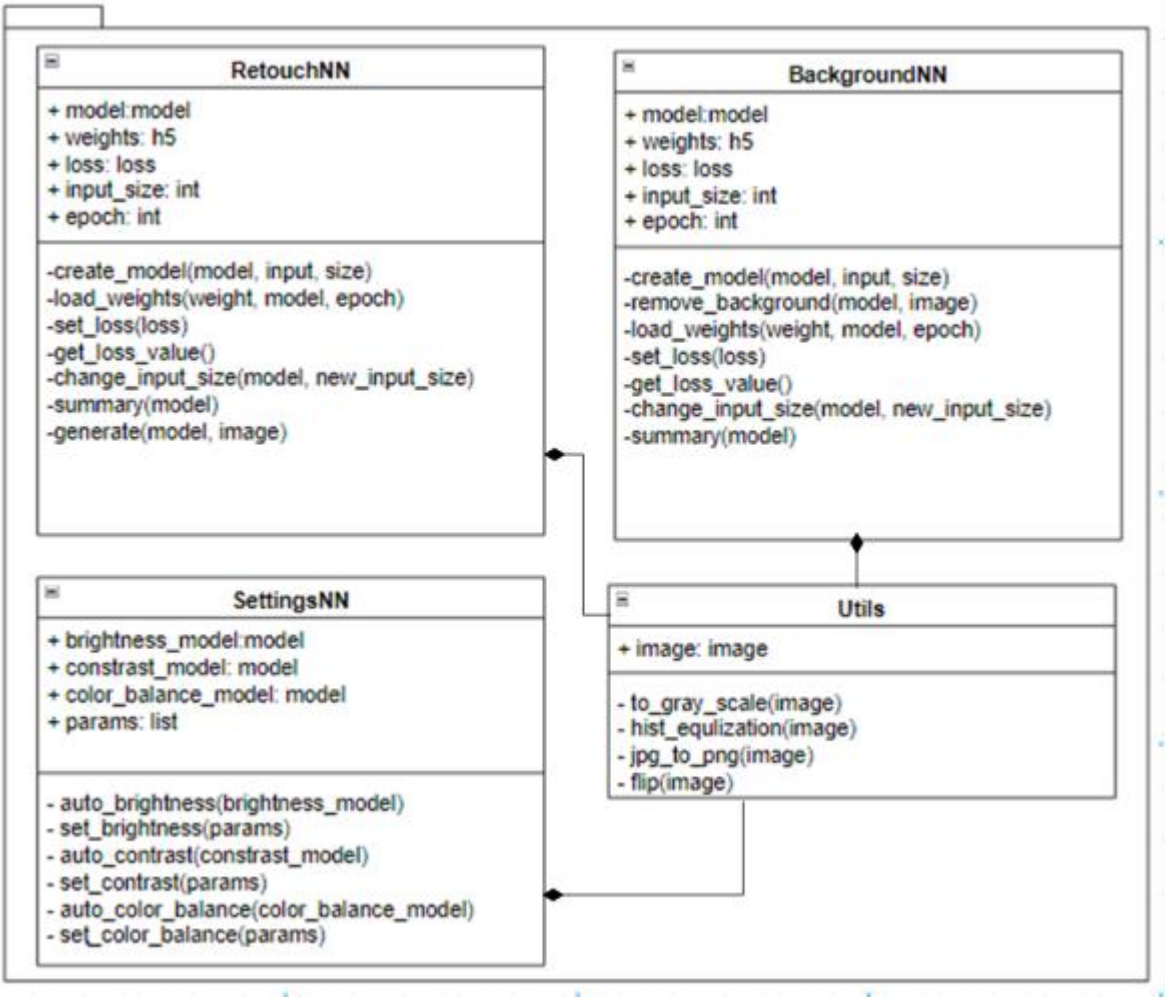

Рис 4.6 – Діаграма класів. Пакет «Redactor»

Пакет «Dataloader» (рис 4.7) створений для перетворення завантаженої фотографії в зображення, яке необхідно подати на вхід навчанної мережі. Оскільки різні мережі потребують різних вхідних зображень, пакет «Dataloader» перетворює їх враховуючи необхідний для мережі розмір, кількість каналів кольору, тощо.

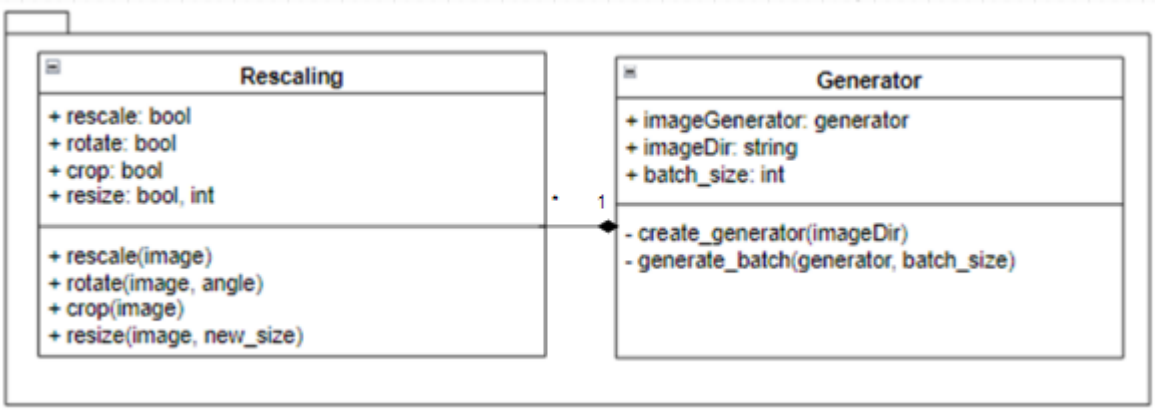

Рис 4.7 – Діаграма класів. Пакет «Dataloader»

4.5 Проектування інтерфейсу користувача

У даному розділі зображені макети, що відображають графічний інтерфейс користувача. Вони були розроблені перед початком реалізації розробки програмної системи в цілому і дозволяють визначити додаткові вимоги і обмеження ще до старту розробки.

На рисунку 4.8 зображено стартовий екран системи. Його призначення полягає у тому, щоб користувач міг увійти в систему під власним ім’ям або створити нового користувача.

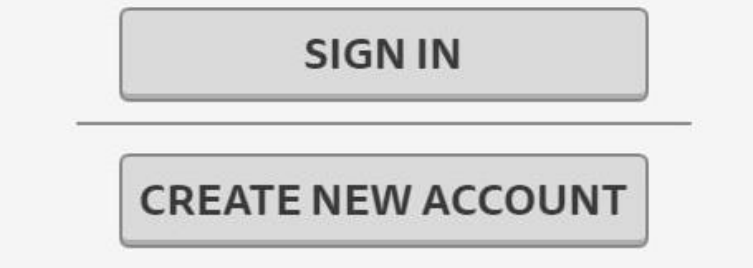

Рис 4.8 – Стартовий екран системи

Вікно входу до системи (рис 4.9) містить поля для введення логін і паролю користувача, а також кнопку «Go» для входу і «forget password», якщо користувач забув власний пароль

На рисунку 4.10 зображений макет для сторінки з реєстрацією нового користувача. Містить поля – логін, пароль і електронна пошта. Усі поля є обов’язковими для заповнення.

Рис 4.9 – Макет екрану входу до системи

Рис 4.10 – Макет екрану реєстрації

Наступним слідує макет екрану для завантаження зображень (рис 4.11). Має мінімалістичну структуру: три кнопки, що відповідають за завантаження, видалення і редагування зображення, і рамка, куди буде виведене завантажене зображення.

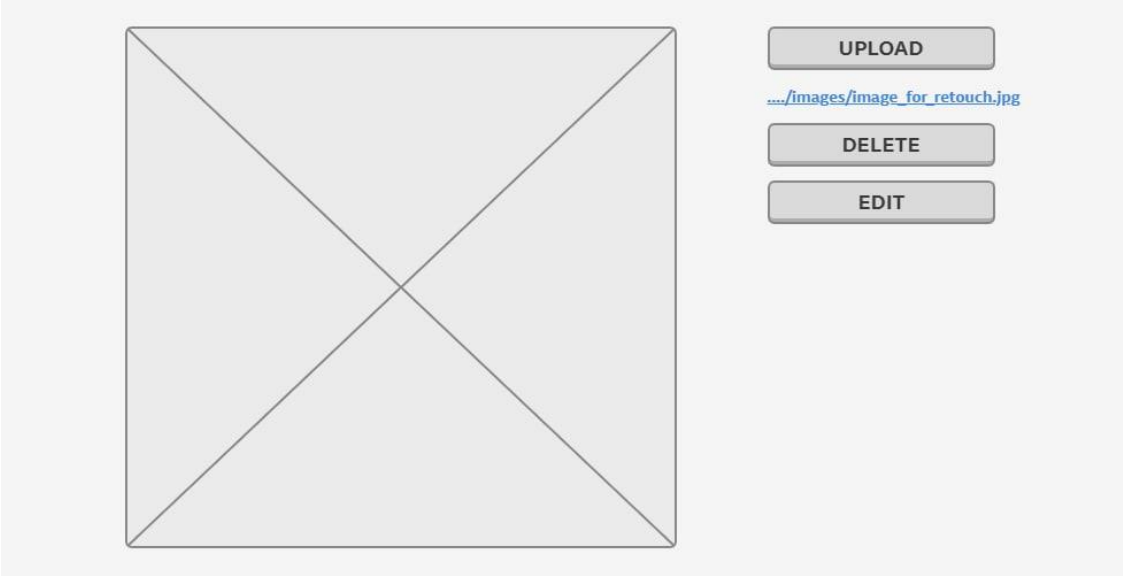

Рис 4.11 – Макет екрану завантаження зображення

І нарешті макет для екрану редагування зображень (рис 4.12). Містить два зображення: перше завантажене, а друге – результат редагування. Зліва розташована панель з інструментами для редагування.

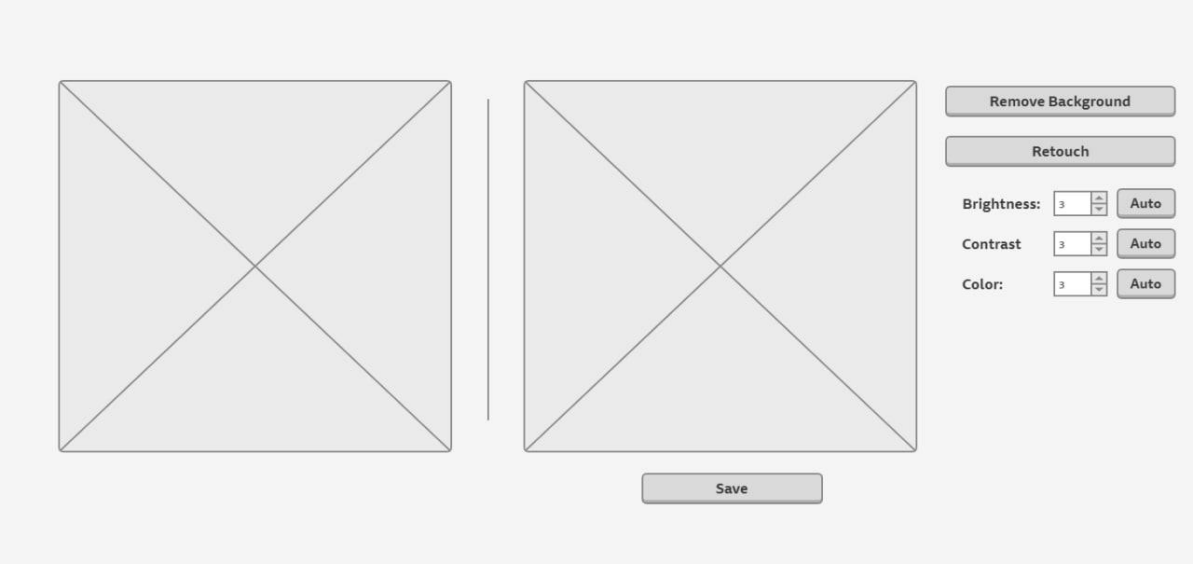

Рис 4.12 – Макет екрану редагування

5 РЕАЛІЗАЦІЯ ТА ВИПРОБУВАННЯ СИСТЕМИ

5.1 Опис програмних технологій

Для розробки програмної систему використовувалась середа Google Colab, що дозволяє використовувати хмарні ресурси для навчання важких нейронних мережей. У якості мови програмування для створення, навчання і тестування мережей був використаний Python. Потім, використовуючи технології TensorflowJS, мережі були імпортовані до середовища Java Script, у якому і було розроблено повноцінну програмну систему. Для розробки системи був також використаний швидкий редактор коду Visual Studio Code.

Python - високорівнева мова програмування з динамічною строгою типізацією та автоматичним управлінням пам’яті. Орієнтована на підвищення продуктивності розробника та читабельності коду.

JavaScript – мова програмування, що зазвичай використовується як вбудована мова для програмного забезпечення доступу до об'єктів додатків. Найширше застосування знаходить у браузерах як мову сценаріїв для надання інтерактивності веб-сторінок.

TensorFlow – це відкрита програмна бібліотека для машиного навчання, що була розроблена компанією Google для вирішення задач побудови, навчання і тестування нейронних мереж та інших методів. Основний API для роботи з бібліотекою реалізован для мови програмування Python. Має підрозділ TensorFlowJS для використання з мовою програмування JavaScript.

Під час розробки та тестування програмної системи був використаний ПК із операційною системою Windows 10.

5.2 Інструкція з використання

Початковий екран програмної системи (рис 5.1) містить усього дві кнопки – «Sign In» і «Create new account». Перша з них відповідає за вхід користувача, якщо він уже попередньо створив аккаунт. Друга відповідає безпосередньо за реєстрацію користувача (рис 5.2). Для її проходження

необхідно заповнити поля «Login», «Password», «E-mail» і натиснути кнопку «Create».

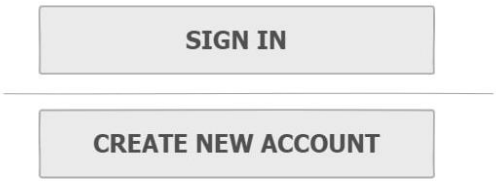

Рисунок 5.1 – Початковий екран системи

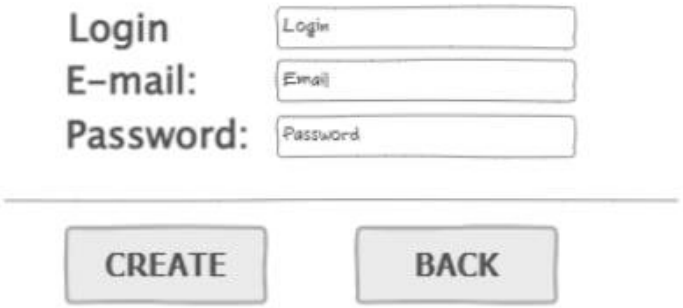

Рисунок 5.2 – Екран реєстрації нового користувача

Отже після проходження реєстрації користувач входить до програмної системи (рис 5.3) використовуючи власний логін і пароль.

Рисунок 5.3 – Екран входу до системи

Після входу користувачу відображається екран з завантаження зображення (рис 5.4). Після натискання кнопки «Upload» відкривається вікно з вибором зображення для завантаження (рис 5.5), і після завантажене

зображення демонструється в лівій частині екрану. Зображення повинні мати розширення jpg або png. Кнопка «Delete» дозволяє видалити зображення і потім завантажити інше. А кнопка «Edit» служить для переходу до редагування. Вона є неактивною поки відсутнє певне завантажене зображення.

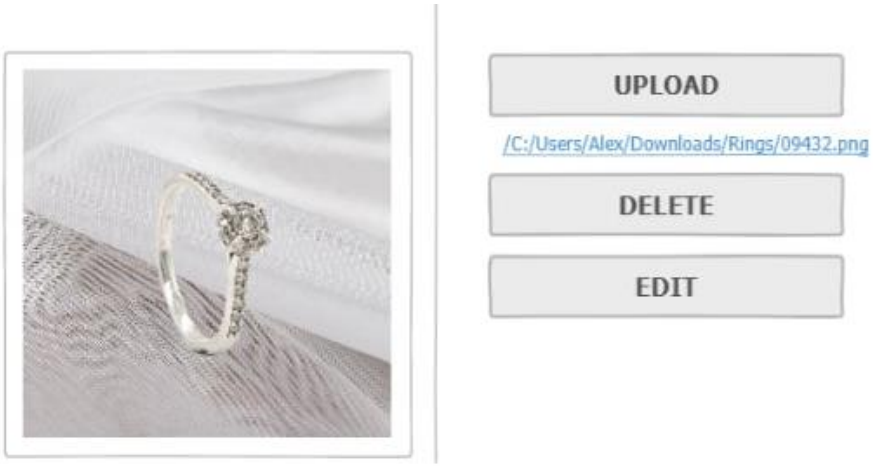

Рисунок 5.4 – Екран завантаження зображення

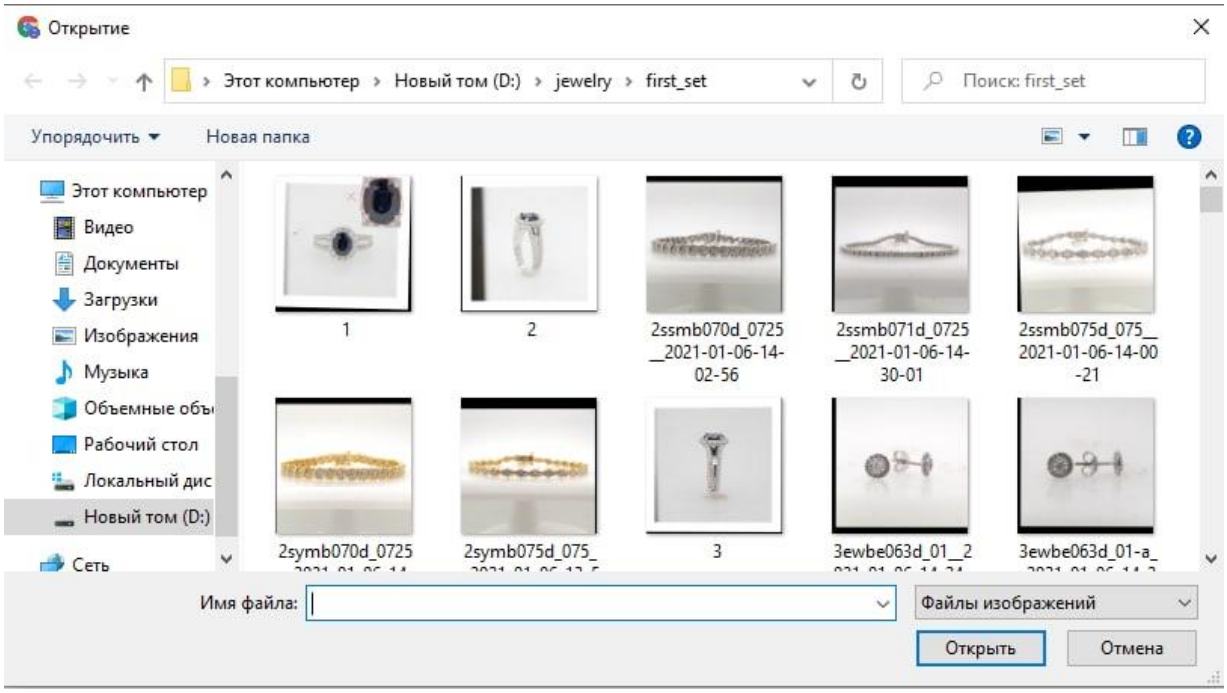

Рисунок 5.5 – Вікно вибору зображення

Після того як користувач натискає кнопку «Edit» система переходить до екрану редактора зображень. При натискані на кнопку «Background remove» відбувається очищення заднього фону зображення, і результат

відображається зправа від початкового зображення (рис 5.6). Після використання кнопка стає неактивною.

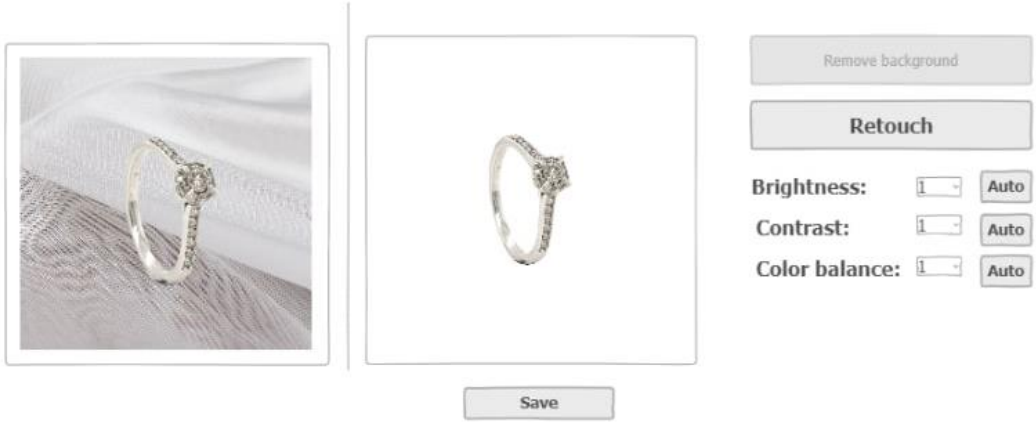

Рисунок 5.6 – Видалення заднього фону зображення

Далі можна провести автоматичне ретушування зображення для надання об'єкту рекламного виду, натиснувши кнопку «Retouch». Результат роботи зображений на рисунку 5.7. Або ж налаштувати параметри зображення такі як яскравість, контрастність, баланс кольору, або вибрати автоматичний режим їх налаштування, використовуючи кнопку «Auto».

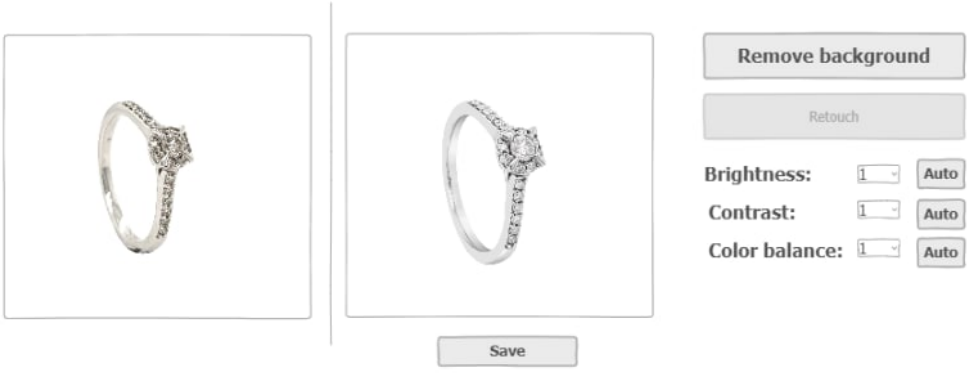

Рисунок 5.7 – Екран після автоматичного ретушування зображення

5.3 Функціональне тестування системи

Тестування служить для перевірки заявлених у системі вимог і дійсно реалізованих функціональних рішень для їх виконання. Тестування системи проводилося методом «чорного ящика», що дозволяє протестувати роботу системних характеристик системи, але в той же час не поглиблюючись до їх внутрішньої структури. Даний підхід дозволяє розробнику перевірити реакцію систему на певні дії користувача безпосередньо під час його роботи з програмним продуктом.

Для тестування були обрані тести, базуючись на діаграмі варіантів використання, що була представлена раніше. Були побудовані відповідні тест-кейси.

Найменування теста: Вхід в систему

Передумова: Користувач завантажив програмний продукт

Результат: Тест-кейс (табл. 5.1)

Таблиця 5.1 – Тест-кейс входу до системи

Крок	**Опис**	**Очікуваний результат**	**Статус**
Крок 1	Користувач знаходиться на початковому екрані	Відображається дві кнопки «Sign In» і «Create New Account»	Успішно
Крок 2	Користувач натискає кнопку «Sign In»	Відкрилась форма входу з полями – «login» і «password», а також кнопки «Sign In» і «Forget password»	Успішно
Крок 3	Користувач вводить власний логін у відповідне поле	Система відображає введений логін в поле «login»	Успішно

Продовження таблиці 5.1

Крок	Опис	Очікуваний результат	Статус
Крок 4	Користувач вводить власний пароль у відповідне поле	Система відображає введений пароль в поле «password»	Успішно
Крок 5	Користувач натискає кнопку «Sign In»	Система переходить до вкладки завантаження зображення	Успішно

Найменування теста: Створення нового аккаунту

Передумова: Користувач завантажив програмний продукт

Результат: Тест-кейс (табл. 5.2)

Таблиця 5.2 – Тест-кейс створення нового аккаунту

Крок	Опис	Очікуваний результат	Статус
Крок 1	Користувач знаходиться на початковому екрані	Відображається дві кнопки «Sign In» і «Create New Account»	Успішно
Крок 2	Користувач натискає кнопку «Create New Account»	Відкрилась форма реєстрації з полями – «login», «e-mail» і «password», а також кнопка «Create»	Успішно
Крок 3	Користувач вводить логін, який прагне використовувати у подальшому	Система відображає введений користувачем логін в поле «login»	Успішно
Крок 4	Користувач вводить новий пароль для реєстрації	Система відображає введений користувачем пароль в поле «password»	Успішно

Продовження таблиці 5.2

Крок	Опис	Очікуваний результат	Статус
Крок 5	Користувач вводить власну електрону адресу	Система відображає введену електрону адресу в поле «e-mail»	Успішно
Крок 6	Користувач натискає кнопку «Create»	Система відображає вікно про успішну реєстрацію і переходить до вікна входу	Успішно

Найменування теста: Завантаження зображення

Передумова: Користувач увійшов до системи

Результат: Тест-кейс (табл. 5.3)

Таблиця 5.3 – Тест-кейс завантаження зображення

Крок	Опис	Очікуваний результат	Статус
Крок 1	Користувач знаходиться на екрані завантаження	Відображається дві кнопки «Upload» і «Remove» (неактивна) і рамка для зображення	Успішно
Крок 2	Користувач натискає кнопку «Upload»	Відкрилась форма вибору файлу з локального середовища	Успішно
Крок 3	Користувач вибирає необхідне зображення і натискає «Завантажити»	Система завантажує зображення і відображає його в необхідній рамці	Успішно

Найменування теста: Видалення заднього фону зображення

Передумова: Користувач увійшов до системи і завантажив зображення

Результат: Тест-кейс (табл. 5.4)

Таблиця 5.4 – Тест-кейс видалення заднього фону

Крок	**Опис**	**Очікуваний результат**	**Статус**
Крок 1	Користувач знаходиться на екрані завантаження	Відображається три кнопки «Upload» (неактивна), «Remove» і «Edit», а також рамка із завантаженим зображенням	Успішно
Крок 2	Користувач натискає кнопку «Edit»	Система переходить до вкладки «Редагування». Відображаються дві рамки з зображенням. Перша для завантаженого зображення, друга для редагованого.	Успішно
Крок 3	Користувач вибирає пункт «Background remove»	Система виводить у другій рамці зображення після очищення фону	Успішно

Найменування теста: Ретушування зображення

Передумова: Користувач увійшов до системи і завантажив зображення

Результат: Тест-кейс (табл. 5.5)

Таблиця 5.5 – Тест-кейс ретушування зображення

Крок	**Опис**	**Очікуваний результат**	**Статус**
Крок 1	Користувач знаходиться на екрані завантаження	Відображається три кнопки «Upload» (неактивна), «Remove» і «Edit», а також рамка із завантаженим зображенням	Успішно

Продовження таблиці 5.5

Крок	Опис	Очікуваний результат	Статус
Крок 2	Користувач натискає кнопку «Edit»	Система переходить до вкладки «Редагування». Відображаються дві рамки з зображенням. Перша для завантаженого зображення, друга для редагованого.	Успішно
Крок 3	Користувач вибирає вибирає пункт «Auto retouch»	Система виводить у другій рамці зображення після автоматичного ретушування	Успішно

Найменування теста: Налаштування параметрів зображення

Передумова: Користувач увійшов до системи і завантажив зображення

Результат: Тест-кейс (табл. 5.6)

Таблиця 5.6 – Тест-кейс налаштування параметрів зображення

Крок	Опис	Очікуваний результат	Статус
Крок 1	Користувач знаходиться на екрані завантаження	Відображається три кнопки «Upload» (неактивна), «Remove» і «Edit», а також рамка із завантаженим зображенням	Успішно
Крок 2	Користувач натискає кнопку «Edit»	Система переходить до вкладки «Редагування». Відображаються дві рамки з зображенням. Перша для завантаженого зображення, друга для редагованого.	Успішно

Продовження таблиці 5.6

Крок	Опис	Очікуваний результат	Статус
Крок 3	Користувач вводить значення для параметра «Brightness»	Система відображає значення в відповідному полі і змінює зображення	Успішно
Крок 4	Користувач вводить значення для параметра «Contrast»	Система відображає значення в відповідному полі і змінює зображення	Успішно
Крок 5	Користувач вибирає пункт автоматично біля параметра «Color balance»	Система змінює зображення в автоматичному режимі регулювання кольорів	Успішно

Найменування теста: Збереження зображення

Передумова: Користувач увійшов до системи і завантажив зображення

Результат: Тест-кейс (табл. 5.7)

Таблиця 5.7 – Тест-кейс збереження зображення

Крок	Опис	Очікуваний результат	Статус
Крок 1	Користувач знаходиться на екрані редагування	Відображається дві рамки з зображеннями, панель редагування і кнопка «Save»	Успішно
Крок 2	Користувач натискає кнопку «Save»	Система відображає вікно для збереження зображення до локального середовища	Успішно
Крок 3	Користувач обирає місце положення файлу, його ім’я і натискає «Зберегти»	Система зберіє зображення і виводить вікно про успішне збереження файлу.	Успішно

5.4 Випробування системи і аналіз результатів

Основною метою дослідження було підвищення якості автоматичного очищення заднього фону і ретушування зображення для надання йому рекламного вигляду. У попередніх розділах був проведений попередній аналіз і була розроблена архітектура змагально-генеративних мереж. Для повноцінної оцінки якості розробленої системи були проведені порівняння розробленої системи з аналогами.

Першим проведемо тестування щодо мережі, що відповідає за очищення заднього фону зображення (табл. 5.8) за метрикою MAE (Mean Absolute Error) на тестовому наборі даних. Наступні набори даних були обрані для тестування:

1) Власний - аналогічний, що й був для навчання мережі. У якості вхідного зображення – фото з чітко виділеним головним об’єктом. А у якості виходу маска фото що виділяє центральний об’єкт від фону зображення.
2) Набір даних SOD
3) Набір даних HKU-IS

Таблиця 5.8 – Результати випробування для мереж з очищення заднього фону зображення

Мережа\Набір	Власний набір	HKU-IS	SOD
Розроблена мережа	0.054	0.032	0.112
AFNet	0.086	0.036	0.119
U2-Net	0.071	0.031	0.108
FCN	0.078	0.032	0.113

Отже, з отриманої таблиці результаті помітно, що розроблена мережа значно краще справляється з власним набором даних, що є основною

метрикою, адже планується, що саме такі зображення будуть використовувати користувачі програмної системи. Розроблена мережа відповідно в середньому на 59%, 31% і 44% має менше значення помилки на тестовому наборі даних за свої конкурентів.

Такий результат пояснюється тим, що мережа при розробці пристосувалася під задані умови. Але слід також помітити, що результати мережі на інших наборах даних не набагато гірші за результати аналогів.

Далі проведемо відповідне тестування і для другої мережі, що відповідає за ретушування зображення (табл. 5.9). Для тестування візьмемо власний набір даних, набір з пар початкова фотографія об’єкта та відретушована версія зображення, і MonetPhoto [33], набір що містить перемальовані зображення в стилі відомого художника Клода Моне. Описання функції втрат знаходиться у розділі 3. Навчання для кожної з мережей проходило по 200 епох. Результати порівняння відображені у таблиці 5.9.

Таблиця 5.9 – Результати випробування для мереж з ретушування зображень

	Власний набір (трену-вання)	Власний набір (тесту-вання)	MonetPhoto (тренуван-ня)	MonetPhoto (тестуван-ня)
Розроб-лена мережа	0.02678	0.02713	0.0548	0.0539
pix2pix	0.03567	0.037172	0.0467	0.04701
CycleGan	0.09109	0.09012	0.0325	0.03411

Отримані результати свідчать, що розроблювальна модель значно краще адаптована до власного набору даних, даних які плануються до використання

у програмній системі. Так значення рівня втрат в середньому при навчані і тестувані менше на 33,2%, ніж у мережі pix2pix, і більш ніж в три рази краще ніж у CycleGan. При цьому необхідно відмітити, що результати роботи на іншому наборі даних гірші за конкурентів, це свідчить, що розроблена мережа є добре підлаштованою під необхідний набір зображень.

Для підвищення зрозумілості і повноцінності отриманих результатів тестування можна також використати конформні предиктори [34] [35].

ВИСНОВКИ

В ході виконання кваліфікаційної роботи були виконані наступні завдання.

Проведено вибір наявних згорткових мереж та мереж GAN, а також методів для вирішення таких задач як видалення фону зображень та їх ретушування, порівняно їх загальну структуру і структуру окремих блоків, виконано тестування для визначено кращих, в результаті кращі були взяті для подальшої доработки. За основу мережі з очищення заднього фону було узято RSU блоки від U2-Net, а для автоматичного ретушування був обраний метод pix2pix, його генератор складається з RSU блоків, а дискримінатор розділений на три частини для роботи з зображеннями різних масштабів. Було проведено налаштування обраних мереж для вирішення поставлених задач, а також зменшення їх розміру за допомогою методів квантизації та дистиляції знань.

В результаті було підвищено якість автоматичного ретушування зображеного об’єкта в середньому на 33,2% відносно мережі pix2pix і більш ніж в три рази відносно CycleGan. Для мережі з очищення заднього фону зменшено значення помилки на тестовому наборі даних на 59%, 31% і 44% відносно мережей аналогів.

Реалізовано веб-застосування для автоматичного очищення заднього фону і ретушування для надавання зображенням, що представляють товари, вражаючий вид для привернення додаткової уваги з боку потенційних покупців.

Розроблена програмна система працює у браузері, а тому може бути використана на будь-яких пристроях, що підтримують роботу відповідних браузерів. Під час розробки використані сучасні засоби розробки, зокрема Python, JavaScript, TensorFlow, PyTorch.

В майбутньому планується випустити мобільну версію програмної системи.

СПИСОК ЛІТЕРАТУРИ

1. Arun Solanki, Anand Nayyar, Mohd Naved, Generative Adversarial Networks for Image-to-Image Translation / USA: Wordware Publishing, 2021 / ISBN 9780128236130.
2. Google Maps App [Електронний ресурс] – Режим доступу до ресурсу: URL: https://play.google.com/store/apps/details?id=com.google – Загол. з екрану.
3. MVC [Електронний ресурс] – Режим доступу до ресурсу: URL: https://web-creator.ru/articles/mvc – Загол. з екрану.
4. Google Play [Електронний ресурс] – Режим доступу до ресурсу: URL: https://play.google.com/ – Загол. з екрану.
5. App store [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.apple.com/ua/app-store/ – Загол. з екрану.
6. Conrad Chavez, Andrew Faulkner, Adobe Photoshop Classroom in a Book / New York, 2021 / ISBN-13: 978-0136904731
7. Saimon Haikein, Neural Networks Full Course / Madrid, 2019 / ISBN: 978-5-907144-22-4
8. Sebastian Rashka, Python и машинное обучение. Машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow / Packt, 2020
9. Super-resolution [Електронний ресурс] – Режим доступу до ресурсу: URL: https://github.com/idealo/image-super-resolution – Загол. з екрану.
10. Deep Photo Enhancer [Електронний ресурс] – Режим доступу до ресурсу: URL: https://github.com/nothinglo/Deep-Photo-Enhancer – Загол. з екрану.

11. Fotor [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.fotor.com/ru/ – Загол. з екрану.
12. CycleGAN [Електронний ресурс] – Режим доступу до ресурсу: URL: https://junyanz.github.io/CycleGAN/ – Загол. з екрану.
13. pix2pix [Електронний ресурс] – Режим доступу до ресурсу: URL: https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-keras/ – Загол. з екрану.
14. Мельник, Програмування веб-застосувань (фронт-енд та бек-енд) / Видавництво Львівської політехніки, 2019/ ISBN: 978-966-941-195-2
15. Mark Lutz, Learning Python / O`Reilly, 2017 / ISBN: 978-617-7812-51-6
16. Pramod Singh, Avinash Manure, Learn TensorFlow 2.0 Implement Machine Learning and Deep Learning Models with Python / Apress, 2018/ ISBN-13: 978-1484255575
17. Torch [Електронний ресурс] – Режим доступу до ресурсу: URL: https://pytorch.org/ – Загол. з екрану.
18. David Flanagan, JavaScript Full Course / O`reilly, 2019 / ISBN: 978-617-7874-22-4
19. Lili Mou, Zhi Jin, Tree-Based Convolution Neural Networks Principles and Application, Springer, 2019 / ISBN: 978-981-13-1870-2
20. ReLu [Електронний ресурс] – Режим доступу до ресурсу: URL: https://uk.wikipedia.org/wiki/ReLU – Загол. з екрану.
21. Diskit, Neilbuff, The Art of Strategy: A Game Theorist's Guide to Success in Business and Life / Springer, 2016 / ISBN: 978-5-00117-174-4
22. Conditional GAN [Електронний ресурс] – Режим доступу до ресурсу: URL: https://machinelearningmastery.com/how-to-develop-a-

conditional-generative-adversarial-network-from-scratch/ – Загол. з екрану.

23. Data Augmentation [Електронний ресурс] – Режим доступу до ресурсу: URL: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data/ – Загол. з екрану.
24. Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks / O'reilly, 2016
25. FCN [Електронний ресурс] – Режим доступу до ресурсу: URL: https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1?gi=741688db1cc4 – Загол. з екрану.
26. AFNet [Електронний ресурс] – Режим доступу до ресурсу: URL: https://kompetenznetz-vorhofflimmern.de/en – Загол. з екрану.
27. U2-Net [Електронний ресурс] – Режим доступу до ресурсу: URL: https://github.com/xuebinqin/U-2-Net – Загол. з екрану.
28. Adam [Електронний ресурс] – Режим доступу до ресурсу: URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ – Загол. з екрану.
29. Johnson structure [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.researchgate.net/figure/Architecture-of-Generator-and-Discriminator-Network-with-corresponding-number-of-feature_fig2_308152499 – Загол. з екрану.
30. Ruvinskaya V. M., Timkov Y. Y. Deep Learning Technology for Videoframe Processing in Face Segmentation on Mobile Devices. Herald of Advanced Information Technology. 2021; Vol.4 No.2: 185–194. DOI: https://doi.org/10.15276/hait.02.2021.7.
31. Knowledge Distillation [Електронний ресурс] – Режим доступу до ресурсу: URL:

https://intellabs.github.io/distiller/knowledge_distillation.html – Загол. з екрану

32. IEEE 830 (Software requirements specification) [Електронний ресурс] – Режим доступу до ресурсу: URL: https://standards.ieee.org/standard/830-1998.html– Загол. з екрану
33. Monet Photo [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.kaggle.com/amyjang/monet-cyclegan-tutorial – Загол. з екрану.
34. Рувінська В. М., Натальчишин О. В. kNN класифікатор з конформними предикторами / Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів "Інтелектуальні інформаційні системи", 28–31 січня 2020. - С. 40-42.
35. Ruvinskaya Victoria, Shevchuk Igor, Michaluk Nikolai. Models Based on Conformal Predictors for Diagnostic System in Medicine / Applied Aspects of Information Technology, No. 02 (03), 2019, - pp 49-59. – Access mode: https://aait.opu.ua/?fetch=articles&with=book&id=3.