

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ

SOFTWARE FOR IMAGE PROCESSING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ІНДЕНКО М.С.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛЬЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ІНДЕНКО Марини Сергіївни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для опрацювання зображень.
Software for Image Processing

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Марина ІНДЕНКО
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Михайло КОЦУР
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
94 с., 3 табл., 32 рис., 3 дод., 19 джерел.

PYCHARM, PYTHON, ГАЛЕРЕЯ, ЗОБРАЖЕННЯ, МОВА
ПРОГРАМУВАННЯ, ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ, СТИЛЬ.

Об'єкт дослідження – процес розробки програмного забезпечення для
опрацювання зображень.

Предмет дослідження – програмні засоби для опрацювання зображень.

Мета роботи – розробка програмного забезпечення для опрацювання та
трансформації зображень.

Матеріали, методи та технічні засоби: мова програмування Python,
середовище розробки PyCharm.

Результати. Розроблено проєктні рішення для створення програмного
забезпечення для опрацювання та трансформації зображень. Створено
програмне забезпечення для опрацювання та трансформації зображень.
Проведено тестування розробленого програмного забезпечення для
опрацювання та трансформації зображень.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення
для опрацювання та трансформації зображень за допомогою мови
програмування Python та середовища розробки PyCharm.

Галузь використання – опрацювання зображень.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 3 tables, 32 figures, 3 appendixes, 19 sources.

PYCHARM, PYTHON, GALLERY, IMAGES, PROGRAMMING LANGUAGE, IMAGE PROCESSING, STYLE.

The object of research is the process of developing software for image processing.

The subject of the research is software for image processing.

The purpose of this work is to develop the software for image processing and transformation.

Materials, methods and technical tools: Python programming language, PyCharm development environment.

Results. Project solutions for the creation of software for the image processing and transformation has been designed. The software for the image processing and transformation has been developed. The developed software for the image processing and transformation has been tested.

Conclusions. The goal of the work has been achieved. The software tools for the image processing and transformation using the Python programming language and the PyCharm development environment have been developed.

Scope of use – image processing.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для опрацювання та трансформації зображень	11
1.2 Аналіз програмних засобів для опрацювання та трансформації зображень	23
1.3 Висновки за розділом 1	32
2 Матеріали і методи	34
2.1 Вибір мови програмування.....	34
2.2 Вибір середовища розробки для створення програмного забезпечення для опрацювання та трансформації зображень	37
2.3 Висновки за розділом 2	39
3 Опис програми	41
3.1 Структура програмного забезпечення для опрацювання та трансформації зображень	41
3.2 Функціонування програмного забезпечення для опрацювання та трансформації зображень	44
3.3 Використання нейромережових моделей VGG для опрацювання зображень	46
3.4 Проєктування інтерфейсу програмного забезпечення для опрацювання та трансформації зображень	52
3.5 Висновки за розділом 3	54
4 Експлуатація, тестування та експериментальне дослідження програми.....	56
4.1 Призначення й умови застосування програми	56
4.2 Характеристики програмного забезпечення для опрацювання та трансформації зображень	57
4.3 Інструкція по експлуатації програми.....	59

4.3.1	Звернення до програми.....	59
4.3.2	Вхідні й вихідні дані.....	60
4.3.3	Повідомлення.....	60
4.4	Виконання програмного забезпечення для опрацювання та трансформації зображень	60
4.5	Тестування програмного забезпечення для опрацювання та трансформації зображень	64
4.6	Висновки за розділом 4	65
	Висновки.....	66
	Перелік джерел посилань.....	69
	Додаток А Технічне завдання.....	71
	Додаток Б Фрагмент тексту програми	76
	Додаток В Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

VGG – Visual Geometry Group;

БД – база даних;

ОЗ – обробка зображень;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для опрацювання зображень сьогодні зумовлена кількома ключовими факторами. У сучасному світі цифрових технологій візуальний контент є домінуючим: велика кількість інформації в Інтернеті представлена у вигляді фото, відео та графіки. Це створює значний попит на ефективні програмні інструменти для швидкої, якісної та креативної обробки зображень, особливо у сфері соціальних мереж, маркетингу та реклами [1], [2].

Програмне забезпечення для роботи із зображеннями є важливим у багатьох галузях. У медицині воно використовується для аналізу результатів обстежень, у промисловості застосовується для контролю якості продукції за допомогою комп'ютерного зору, у науці програми допомагають обробляти супутникові та мікроскопічні зображення. Також вони потрібні у сфері безпеки для розпізнавання облич чи номерних знаків, а в мистецтві та дизайні для редагування фотографій, створення графіки та 3D-моделювання [3], [4].

Сучасне програмне забезпечення для опрацювання зображень все активніше інтегрує штучний інтелект. Алгоритми машинного навчання та нейронні мережі дозволяють автоматично коригувати фото, застосовувати фільтри, відновлювати пошкоджені зображення, розпізнавати об'єкти, виділяти контури й покращувати якість фото. Це відкриває нові можливості використання таких програм у найрізноманітніших сферах [5], [6].

Водночас спостерігається тенденція до розвитку мобільних застосунків, що робить потужні інструменти доступними на мобільних телефонах і планшетах [6], [7].

Проте деякі програмні засоби для опрацювання зображень характеризуються високою вартістю ліцензій, що робить їх недоступними для багатьох користувачів або малих компаній. Безкоштовні або дешевші альтернативи часто обмежені за функціоналом, що не дозволяє реалізувати складні або спеціалізовані завдання. Ще однією проблемою є складність у

використанні, оскільки багато професійних програм мають складний інтерфейс і вимагають часу для навчання, що створює бар'єр для початківців, що змушує інвестувати ресурси у підготовку користувачів або залучення спеціалістів. Також варто зазначити, що деяке програмне забезпечення для опрацювання та трансформації зображень потребує високих обчислювальних ресурсів: потужного процесора, великого обсягу оперативної пам'яті та відеокарти. Через це старі або малопотужні комп'ютери не здатні забезпечити комфортну роботу з такими програмами, що вимагає додаткових витрат на оновлення техніки. Також у разі роботи з великими зображеннями чи складними проєктами виникає ризик втрати даних через збої програм або несумісність форматів файлів між різними програмами, що ускладнює обмін інформацією між платформами [1]-[7].

Тому актуальною є розробка програмного забезпечення для опрацювання та трансформації зображень. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для опрацювання та трансформації зображень.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для опрацювання та трансформації зображень;
- здійснити проєктування програмного забезпечення для опрацювання та трансформації зображень;
- створити програмне забезпечення для опрацювання та трансформації зображень;
- виконати тестування розробленого програмного забезпечення для опрацювання та трансформації зображень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для опрацювання та трансформації зображень

Опрацювання зображень — це процес, за допомогою якого цифрові зображення трансформуються, аналізуються або вдосконалюються шляхом застосування різних алгоритмів і технік за допомогою комп'ютерних технологій. Основна мета такого опрацювання полягає в отриманні більш інформативного або якіснішого результату, що може полегшити подальше використання зображення в інших завданнях, зокрема у програмних системах розпізнавання, медичній діагностиці чи автоматизованих системах контролю якості продукції [1], [2].

Коли цифрове зображення завантажується в комп'ютерну систему, воно репрезентується у вигляді матриці, кожен елемент якої відповідає окремому пікселю, що має певну величину інтенсивності. Така інтенсивність відображає яскравість конкретного пікселя й задається числовим значенням. У звичних для використання зображеннях інтенсивність кодується за допомогою восьми бітів, що дозволяє кожному пікселю набувати одного зі значень у межах від мінімального до максимального [1], [2].

У цифровій обробці зображень застосовується програмне забезпечення, яке дозволяє удосконалювати візуальні матеріали за допомогою комп'ютерних технологій. Таке опрацювання виконується для покращення вигляду зображень або для виокремлення з них важливої інформації. Застосування подібних програм охоплює різні галузі: від обробки фотографій до складних завдань у сфері машинного навчання, де попередня підготовка зображень допомагає підвищити ефективність моделей [1], [2].

Програмне забезпечення для роботи із зображеннями використовується і для редагування їхнього вмісту, що дає змогу змінювати окремі елементи, додавати нові деталі або видаляти непотрібні частини. Такий підхід є актуальним у творчих індустріях, зокрема у виробництві медіаконтенту.

Опрацювання цифрових зображень передбачає їх представлення у вигляді матриць, де кожен піксель відповідає певному значенню яскравості або кольору. У найпростішому варіанті використовується лише два значення пікселів, що формують бінарне зображення, необхідне для виконання базових операцій сегментації [1], [2].

Коли обробка передбачає відтворення більшої кількості деталей, використовуються зображення у відтінках сірого, що складаються з багатьох градацій яскравості. Для роботи з кольорами застосовуються зображення, у яких кожен піксель утворюється комбінацією червоного, зеленого та синього каналів. Інформація про прозорість додається за допомогою альфа-каналу, що дозволяє налаштовувати ступінь непрозорості пікселів [1], [2].

Програмне забезпечення для опрацювання зображень забезпечує можливість виконувати широкий спектр операцій, від базового коригування яскравості до складних методів сегментації, ретушування або зміни структури зображення, що дозволяє адаптувати результати до різноманітних потреб користувачів [1], [2].

Існують різні види цифрових зображень, які відрізняються за кількістю можливих відтінків та інформаційних каналів. Деякі графічні файли складаються лише з двох варіантів пікселів, що дозволяє відображати тільки чорний та білий кольори. Такі зображення часто використовуються для виділення окремих елементів, що є важливим під час виконання операцій сегментації. Інші зображення формуються на основі різних відтінків сірого кольору, де кожному пікселю присвоюється значення, що відповідає певній градації між темним і світлим [1], [2].

Кольорові цифрові зображення зберігають інформацію через поєднання трьох незалежних каналів, кожен з яких відповідає за свою складову: червону, зелену чи синю. У такій структурі кожен піксель формується шляхом об'єднання значень цих трьох каналів, що дозволяє створювати широкий спектр кольорових відтінків. Для опису прозорості до звичайної кольорової моделі додається ще один компонент, який задає ступінь видимості пікселя.

Завдяки цьому стає можливим моделювання ефекту часткової чи повної прозорості, що використовується під час роботи з шарами та складними композиціями [1], [2].

Отже, обробка зображень охоплює різноманітні підходи до перетворення цифрової графіки, що дає змогу підвищувати її якість, виділяти потрібну інформацію або адаптувати вміст для конкретних завдань. Цей процес здійснюється за допомогою спеціалізованого програмного забезпечення, що реалізує математичні та алгоритмічні методи роботи з піксельними даними [1], [2].

Обробка зображень передбачає послідовне виконання кількох етапів, кожен з яких має своє призначення у формуванні та трансформації цифрових візуальних даних. Спочатку відбувається захоплення зображення, під час якого інформація, отримана камерою, транслюється у цифровий вигляд завдяки спеціальним пристроям, що переводять аналогові сигнали у цифрові. Після отримання такого цифрового зображення застосовуються методи, що покликані зробити його більш придатним для подальшого використання. Ці операції допомагають виявити деталі, приховані у вихідному матеріалі, наприклад, підсилюючи контрастність або усуваючи розмиття [2], [3].

На наступному етапі увага зосереджується на покращенні загальної якості, зокрема шляхом усунення перешкод чи шумів, що могли виникнути під час зйомки або оцифровування. Іноді необхідно працювати саме з кольоровими зображеннями, де обробці підлягають багатоканальні дані, що представляють кольори, а також додаткові характеристики, пов'язані з прозорістю. Деякі підходи передбачають представлення зображень через багаторівневі структури, що дозволяють одночасно аналізувати деталі на різних масштабах [2], [3].

Для полегшення зберігання та передачі зображень у цифрових системах вдаються до процедур стискання, що зменшують обсяг даних без критичної втрати якості. Такий підхід забезпечує економію ресурсів і швидший обмін файлами. Подальші етапи пов'язані з аналізом структури

зображення, де математичними методами виділяються форми або контури об'єктів, що допомагає глибше зрозуміти його зміст [2], [3].

Щоб зробити інформацію більш зрозумілою для комп'ютера, виконується поділ зображення на окремі значущі фрагменти. Завдяки цьому машина отримує змогу сфокусуватися на важливих елементах, відкидаючи зайве. Після цього для кожного виділеного сегмента формуються описи, які надають кількісну або якісну характеристику властивостей об'єкта. Завершальний етап передбачає позначення виявлених об'єктів і надання їм ідентифікаторів, що дозволяє автоматизованим системам повідомляти користувачу, що саме було розпізнано на зображенні. Зрештою, формується набір знань, що об'єднує отримані дані та дозволяє використовувати їх для вирішення конкретних завдань, підвищуючи ефективність взаємодії людини і машини [2], [3].

Робота із зображеннями передбачає використання різноманітних технік для удосконалення їхнього вигляду, усунення зайвих елементів або навіть формування нових візуальних матеріалів на основі наявних даних. Наприклад, за допомогою певних прийомів можна залишити на передньому плані лише потрібний об'єкт, прибравши тло [2], [3].

Ця сфера вирізняється складністю та багатогранністю, оскільки пропонує великий набір підходів, що дозволяють досягати різних цілей. Тут увагу зосереджено на поширених завданнях, які вирішуються під час обробки зображень, а також на засобах, що застосовуються для їх виконання [2], [3].

Одним із найважливіших напрямів виступає підвищення якості візуальних матеріалів. Такі операції відіграють суттєву роль у системах машинного зору, під час дистанційного спостереження та в аналітичних завданнях. Одним із найпоширеніших методів виступає регулювання таких характеристик, як яскравість і контраст. Контрастність відображає різницю між найтемнішими й найсвітлішими ділянками, і її збільшення допомагає зробити деталі чіткішими. Яскравість характеризує рівень освітлення, і її підвищення надає зображенню світлішого вигляду. Зазвичай ці параметри

налаштовуються автоматизованими засобами або за допомогою ручного коригування в спеціальних програмах [2], [3].

Водночас такі операції належать до базових. У випадках, коли зображення з високими показниками контрасту й яскравості під час збільшення виглядає розмитим через обмежену кількість пікселів, на допомогу приходять методи підвищення роздільної здатності. Цей підхід дозволяє створювати більш деталізовані зображення з менш якісних джерел, використовуючи сучасні алгоритми. Для таких завдань дедалі частіше застосовуються глибокі нейронні мережі [2], [3].

Однією з перших моделей, що реалізувала цей підхід, є нейромережева модель з архітектурою, де спочатку виконується інтерполяція вхідного матеріалу класичним методом, після чого отримане зображення подається на обробку нейромережею. Мережа формує вихідний результат шляхом накладання фільтрів на окремі області вхідного матеріалу, що дозволяє реконструювати більш деталізоване зображення [2], [3].

Ще однією важливою задачею виступає відновлення пошкоджених матеріалів. Чимало фотографій, особливо тих, що були створені задовго до цифрової ери, зазнали значних ушкоджень, таких як подряпини чи розриви. Використання сучасних інструментів у цьому напрямі відкриває можливості для повернення до життя історичних документів і цінних архівних матеріалів. Сучасні алгоритми, що ґрунтуються на глибокому навчанні, демонструють здатність реконструювати навіть великі втрачені фрагменти [2], [3].

Прикладом такого підходу є заповнення порожніх ділянок на зображенні. Цей процес може базуватися на використанні спеціального програмного забезпечення для створення нових текстур для відсутніх частин, проте саме нейромережеві методи наразі вважаються найефективнішими завдяки здатності розпізнавати контексти й відтворювати відповідні візерунки. Один із запропонованих підходів передбачає використання спеціальної архітектури з двоетапною обробкою: спочатку здійснюється попередня оцінка, а потім відбувається уточнення результату. Ключову роль тут відіграє

механізм семантичної уваги, що допомагає послідовно відновлювати втрачені області за рахунок багаторазових ітерацій [2], [3].

Ще одним важливим процесом є сегментація, під час якої зображення розділяється на окремі частини. Кожна з таких ділянок відповідає певному об'єкту, що дозволяє легше виконувати завдання подальшої обробки або аналізу. Для реалізації цього процесу використовуються різні алгоритми, серед яких особливе місце займає методика застосування порогових значень. У найпростішому випадку зображення перетворюється на двоколірне, де кожен піксель стає чорним або білим залежно від інтенсивності. Поріг обирається так, щоб пікселі з меншою яскравістю перетворювалися на темні, а з вищою – на світлі. Такий підхід дозволяє виділяти об'єкти у вигляді чітко окреслених областей. Якщо ж використовується багаторівневе порогове значення, зображення отримує декілька градацій сірого відповідно до кількості обраних рівнів [2], [3].

У сучасних підходах застосовується автоматичне виділення об'єктів на зображеннях із залученням глибинних моделей, що дає змогу виконувати як одноетапну сегментацію, так і розділення на кілька класів. Одним із прикладів є нейронна мережа для визначення місця розташування та фокусування, що побудована на основі згорткових нейронних мереж. Ця система поєднує два основні елементи: компонент для виявлення місця об'єкта, що нагадує інстинктивну здатність хижака виявляти здобич, а також компонент для уточнення отриманих результатів, який працює над деталізацією шляхом зосередження уваги на невизначених ділянках [2], [3].

Щодо ідентифікації об'єктів, то ця задача передбачає знаходження та класифікацію об'єктів на фото, що широко застосовується у сферах безпеки та моніторингу. Для досягнення цієї мети часто використовуються алгоритми, засновані на глибинних нейронних мережах, серед яких провідне місце займають згорткові архітектури. Завдяки здатності працювати з локальними ділянками зображення, такі мережі ефективно визначають, де саме на

зображенні розташований об'єкт, та підписують його відповідною категорією, що зазвичай позначається контурною рамкою [3], [4].

Одним із поширених рішень для цього завдання є модель, що поєднує процеси вибору потенційних зон, де можуть знаходитися об'єкти, із подальшим уточненням, що саме на цих ділянках розташовано. Цей підхід передбачає почергове вдосконалення як системи вибору зон, так і класифікації об'єктів, при цьому самі зони зберігаються незмінними [3], [4].

Що стосується процесу стискання зображень, то його метою є зменшення обсягу файлу без істотної втрати якості. Така оптимізація важлива для економії простору зберігання, а також для ефективної роботи алгоритмів на малопотужних пристроях або для зменшення обсягу даних під час передачі. Класичні методи здебільшого спираються на алгоритми зі зниженням якості, щоб досягти меншого розміру файлу; прикладом може бути застосування косинусного перетворення в популярному форматі JPEG [3], [4].

Натомість сучасні рішення використовують нейронні мережі для створення компактного представлення зображення у вигляді ознак меншої розмірності, що згодом відновлюються на іншій стороні за допомогою декодера. Такий підхід реалізується через автокодери, які включають блок, відповідальний за створення ефективного коду, та блок, що відновлює первісне зображення з отриманих ознак [3], [4].

Щодо трансформації зображень, то цей процес полягає у внесенні змін, які змінюють їхній вигляд. Такі зміни можуть мати різні цілі, наприклад, усунення небажаних елементів або, навпаки, додавання нових. Ця технологія часто застосовується у творчих галузях, як-от у створенні рекламних матеріалів чи постерів [3], [4].

Одним із прикладів такої трансформації є нейронна передача стилю, що дає змогу поєднати зміст одного зображення зі стилем іншого. Завдяки цьому, наприклад, фотографія може набути вигляду живописного твору. Для цього використовуються глибинні моделі, що здатні адаптуватися під новий стиль, створюючи мистецькі варіації. У деяких розробках було запропоновано

систему, яка здатна виконувати цю задачу майже миттєво, тоді як інші методи потребують значно більше часу. Основою такого рішення є спеціальний модуль, що модифікує статистичні характеристики вхідного зображення відповідно до прикладу стилю, після чого результат декодується у фінальне стилізоване зображення [3], [4].

Генерація нових візуальних даних є одним із важливих напрямів роботи з графічною інформацією, особливо в контексті глибоких нейронних мереж, де потрібні значні обсяги анотованих прикладів для належного навчання. Для таких завдань застосовуються генеративно-змагальні нейронні мережі, структура яких включає два взаємопов'язані елементи: один компонент відповідає за створення штучних зображень, тоді як інший займається їхньою оцінкою, намагаючись відрізнити реальні дані від синтетичних. У процесі навчання відбувається поступове вдосконалення генератора, який намагається формувати настільки переконливі приклади, щоб їх складно було відрізнити від справжніх. Така взаємодія призводить до появи високоякісних візуальних результатів, що можуть використовуватися для тренування інших моделей [3], [4].

Ще одним важливим завданням є трансформація одного візуального зображення в інше, що передбачає знаходження відповідностей між парою вхідного та цільового зображень на основі спеціально підібраних навчальних наборів. Як приклад, можна навести ситуацію, коли контурне зображення слугує основою для отримання фотореалістичного результату. У цьому контексті відома модель використовує умовні генеративно-змагальні мережі, де генерація вихідного зображення відбувається з урахуванням наданої умови, наприклад, у вигляді мітки або іншого контексту. Архітектура такої системи поєднує генератор, побудований за принципами U-Net, та дискримінатор, що аналізує невеликі ділянки, оцінюючи їхню автентичність. Такий підхід дозволяє досягати більш чітких деталей і підвищувати якість кінцевого результату [4], [5].

Варто зазначити, що збільшення доступності візуальних матеріалів у цифрову епоху створює потребу в їхній обробці задля ефективного використання, зберігання та передавання. Водночас сучасні методи, засновані на глибокому навчанні, значно розширили можливості обробки зображень, надаючи інструменти для вирішення завдань, починаючи від зменшення обсягу даних і підвищення якості до створення абсолютно нових візуальних форм [4], [5].

Сучасні дослідження акцентують увагу на пошуках шляхів зниження залежності від великої кількості розмічених даних, використовуючи стратегії частково контрольованого та неконтрольованого навчання, що відкриває перспективи для ширшого практичного застосування таких моделей у реальних умовах [4], [5].

Обробка зображень розглядається як сукупність прийомів, спрямованих на зміну або аналіз візуальної інформації для підвищення її якості чи вилучення значущих даних. У цифровому середовищі ці дії здійснюються шляхом застосування спеціалізованих алгоритмів, що дозволяють змінювати зображення, покращувати його сприйняття, виокремлювати об'єкти чи переводити матеріал у потрібний формат. Використання подібних методів поширюється як на окремі знімки, так і на послідовності кадрів. Процес обробки включає етап введення візуальної інформації, її перетворення за допомогою обчислювальних методів та отримання результатів у вигляді, придатному для подальшого застосування або дослідження [4], [5].

Розрізняються два основних підходи до трансформації зображень, які відрізняються за природою матеріалу. При роботі з аналоговими носіями застосовуються фізичні чи хімічні процедури для модифікації фотографій, плівкових кадрів чи друкованих матеріалів. Такі способи мали особливе значення на етапах до цифрової ери, хоча поступаються сучасним технологіям через меншу гнучкість та вищі витрати часу. Натомість цифрова обробка ґрунтується на маніпулюванні числовими даними, що описують кожен

елемент зображення. Це відкриває можливості для точного контролю, автоматизації та використання широкого спектра технік завдяки комп'ютерній обробці [4], [5].

Покращення візуальної складової часто передбачає втручання у співвідношення світлих і темних зон, що дозволяє підкреслити деталі, невидимі за умов недостатньої контрастності. Окремі алгоритми регулюють яскравість пікселів, розширюючи їх значення на всю ширину гістограми. Інший підхід зосереджується на досягненні рівномірного розподілу яскравості, завдяки чому поліпшується загальне сприйняття сцени, особливо коли на зображенні переважають дуже світлі або темні області [4], [5].

Зменшення впливу випадкових спотворень, які виникають під час запису або передавання зображення, потребує застосування фільтраційних процедур, що згладжують дані, одночасно зберігаючи важливі контури та елементи. Подібні методи допомагають отримати чіткіший результат без істотної втрати важливої інформації [5], [6].

Коли виникає потреба у виправленні недоліків, пов'язаних із розмитістю, використовуються підходи, що дозволяють підвищити різкість. У випадках втрати окремих ділянок застосовуються алгоритми, які здатні відновити відсутні фрагменти, відтворюючи їх за зразками або контекстом сусідніх пікселів. Здійснення такої реконструкції може мати місце як у роботі зі старими знімками, так і під час видалення непотрібних об'єктів [5], [6].

Обробка зображень передбачає виконання різноманітних дій із зображенням для досягнення необхідного результату, наприклад, для виділення важливої інформації або для покращення його сприйняття. Цей процес може застосовуватись як для статичних фотографій, так і для послідовностей кадрів, що утворюють відео. Виконання таких завдань потребує послідовної роботи з цифровим образом, що охоплює етапи введення, обробки та подальшої видачі зміненого результату в зручному для використання вигляді [5], [6].

Для розділення зображення на складові частини часто застосовуються методи сегментації. Серед них можна виділити використання порогового значення, коли на основі порівняння інтенсивностей пікселів із певним критерієм формується бінарна карта, де світлі області відокремлюються від темних. Інший підхід ґрунтується на пошуку контурів, що дозволяє виявляти межі об'єктів завдяки аналізу змін яскравості. Для групування пікселів із подібними характеристиками застосовуються підходи, що базуються на регіонах, де поступово розширюються області, об'єднуючи схожі за властивостями ділянки. Іноді для таких цілей використовується алгоритмічна інтерпретація зображення як рельєфу, що дозволяє застосовувати методи, подібні до гідрологічних моделей [5], [6].

Завдання зі зменшення обсягу даних без значної втрати якості вирішуються шляхом стиснення. Можна скористатися підходами, що передбачають видалення надлишкових даних із частковою втратою точності, або методами, здатними зберегти вихідну інформацію без жодних втрат. У першому випадку досягається суттєве зменшення розміру файлів, що особливо цінується при обміні даними через обмежені канали, у другому – забезпечується точне відновлення первинного образу [5], [6].

Для створення нових зображень або розширення наявних текстур застосовуються методи синтезу. Така діяльність дозволяє на основі невеликих зразків створювати більші області, що виглядають природно та цілісно. Іншим підходом є використання спеціалізованих алгоритмів, здатних генерувати зображення з нуля, спираючись на приклади або задані характеристики, що відкриває широкі можливості в дизайні та візуальному мистецтві [6], [7].

Щоб отримати інформацію про окремі властивості об'єктів на зображенні, застосовуються методи вилучення ознак. Зокрема, аналізуються форми та текстури, що дозволяє робити висновки про геометричні особливості та характер поверхні. Колірні характеристики також можуть стати основою для виділення елементів, якщо використовуються алгоритми, що обробляють палітру кольорів. Розпізнавання образів здійснюється на основі аналізу цих

ознак із метою класифікації, де використовуються як прості алгоритми, так і складні моделі штучного інтелекту [6], [7].

Додатково для роботи з бінарними образами застосовується морфологічна обробка. Цей підхід включає використання операцій, що змінюють форму об'єктів шляхом розширення або зменшення меж. Можливе також комбінування таких операцій для очищення зображень від дрібних шумів чи заповнення пустот. Завдяки спеціалізованим морфологічним фільтрам та програмним засобам забезпечується можливість виділити певні структурні елементи та підсилити важливі особливості, необхідні для подальшої обробки або аналізу [6], [7].

Загалом обробка візуальних даних розглядається як багатогранний процес, що об'єднує різноманітні підходи для досягнення якісного та інформативного результату, який можна використовувати для подальших технічних чи наукових завдань [6], [7].

Технології обробки зображень та відповідні програмні засоби відіграють ключову роль у багатьох галузях, підвищуючи інформативність і якість візуальних даних. Різнноманітні методи, що включають покращення, відновлення, сегментацію, стиснення, синтез, виділення ознак та морфологічні операції, допомагають отримувати корисну інформацію та вдосконалювати візуальний контент. Розвиток цієї галузі відкриває нові можливості та сприяє подальшим інноваціям [6], [7].

Визначено, що опрацювання зображень є процесом, за допомогою якого цифрові зображення трансформуються, аналізуються або вдосконалюються шляхом застосування різних алгоритмів і технік за допомогою комп'ютерних технологій. Показано, що програмне забезпечення для опрацювання зображень забезпечує можливість виконувати широкий спектр операцій, від базового коригування яскравості до складних методів сегментації, ретушування або зміни структури зображення, що дозволяє адаптувати результати до різноманітних потреб користувачів.

1.2 Аналіз програмних засобів для опрацювання та трансформації зображень

Проаналізуємо програмні засоби для опрацювання та трансформації зображень [8]-[13].

Програмне забезпечення Photoroom (рис. 1.1) призначене для опрацювання та редагування зображень, забезпечуючи користувачів потужними інструментами для створення якісного візуального контенту. Програмне забезпечення Photoroom спеціалізується на роботі з фотографіями, дозволяючи обробляти зображення товарів, брендовані матеріали та візуальний контент для онлайн продажів. Завдяки інтегрованим можливостям штучного інтелекту програмне забезпечення Photoroom значно полегшує процеси опрацювання фотографій, автоматизуючи ключові етапи, як-от видалення фону, ретушування чи застосування стилізованих ефектів [8], [11].

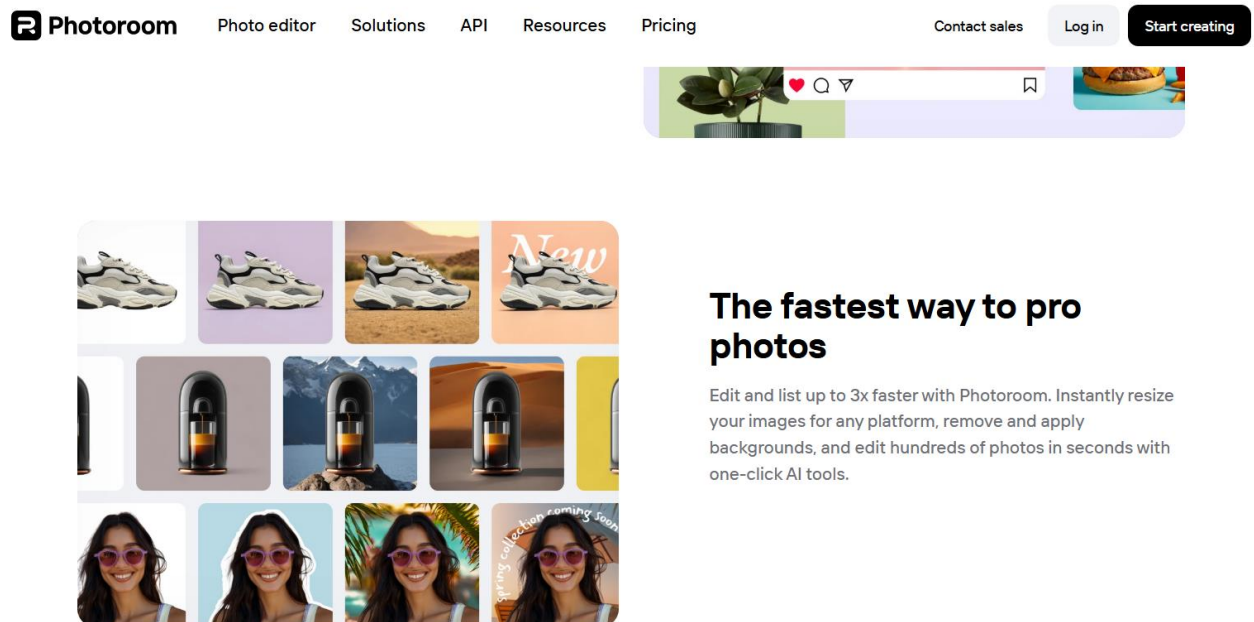


Рисунок 1.1 – Програмне забезпечення Photoroom [11]

Програмне забезпечення Photoroom адаптоване для широкого кола завдань з опрацювання зображень, підтримуючи роботу як на мобільних пристроях, так і на комп'ютерах, що підвищує зручність використання для

різних категорій користувачів. Платформа розроблена як рішення для швидкої та ефективної обробки візуального контенту, зокрема для маркетингу, реклами, публікацій у соціальних мережах та створення матеріалів для електронної комерції [8], [11].

Програмне забезпечення Photoroom надає різноманітні функції для опрацювання зображень, доступні в декількох тарифних планах, що відповідають потребам як окремих творчих користувачів, так і бізнес-клієнтів. Базовий план дозволяє використовувати основні інструменти для редагування фотографій, зокрема функції видалення фону та базового ретушування. Для більш інтенсивного опрацювання зображень передбачено розширені можливості у платних версіях, що відкривають доступ до таких функцій, як пакетна обробка, генерація фону за допомогою штучного інтелекту, розширений редактор зображень та підтримка роботи в команді [8], [11].

Програмне забезпечення Photoroom створене для полегшення роботи з візуальним контентом, що дозволяє користувачам ефективно створювати привабливі, професійно оброблені зображення. Незалежно від того, чи йдеться про індивідуальне використання, чи про інтеграцію в бізнес-процеси, платформа забезпечує гнучкість, масштабованість та зручність опрацювання фотографій [8], [11].

Основними характеристиками програмного забезпечення Photoroom є такі [8], [11]:

- інструмент для обробки зображень: програмне забезпечення Photoroom дозволяє виконувати редагування фотографій та інших зображень із застосуванням алгоритмів штучного інтелекту для покращення якості та стилізації зображень [8], [11];

- видалення фону: програмне забезпечення Photoroom може застосовуватися для автоматичного видалення або заміни фону на зображеннях для підготовки комерційного чи маркетингового контенту користувачів [8], [11];

- пакетна обробка: програмне забезпечення Photoroom забезпечує можливість одночасного редагування великої кількості зображень, що підвищує ефективність роботи над масштабними проєктами [8], [11];

- багатоплатформеність: програмне забезпечення Photoroom підтримує роботу на різних платформах, включаючи вебверсію, iOS та Android, що розширює можливості використання програмного забезпечення [8], [11];

- інструменти для командної роботи: програмне забезпечення Photoroom дозволяє працювати кільком користувачам над спільними проєктами із можливістю розподілу доступу та управління завданнями в межах проєкту [8], [11];

- адаптивні тарифні плани: програмне забезпечення Photoroom пропонує різні варіанти підписок залежно від потреб користувачів, від безкоштовного функціоналу до розширених можливостей професійного рівня користувачів [8], [11];

- інтеграція API: програмне забезпечення Photoroom забезпечує можливість впровадження функцій Photoroom у власні робочі процеси бізнесу через інтерфейси програмування застосунків [8], [11];

- генерація контенту для соціальних мереж: програмне забезпечення Photoroom оптимізує зображення для публікацій у соціальних мережах, надаючи інструменти для швидкого створення привабливих візуальних матеріалів [8], [11].

Серед недоліків програмного забезпечення Photoroom можна відзначити обмежений функціонал безкоштовної версії, що не завжди задовольняє потреби користувачів, які працюють з великою кількістю зображень або виконують складне опрацювання. Деякі розширені можливості програмного забезпечення Photoroom, зокрема пакетна обробка чи генерація складних фонових ефектів за допомогою штучного інтелекту, доступні лише у платних версіях, що може вимагати додаткових витрат. Також можна відмітити залежність якості обробки від вихідного зображення, що іноді потребує ручного коригування для досягнення бажаного результату [8], [11].

Програмне забезпечення Phot AI (рис. 1.2) призначено для опрацювання зображень із використанням можливостей штучного інтелекту, що забезпечує їх удосконалення та персоналізацію. За допомогою програмного забезпечення Phot AI здійснюється ефективне редагування фотографій, зокрема шляхом покращення їхньої якості, виправлення недоліків і видалення зайвих об'єктів [9], [12].

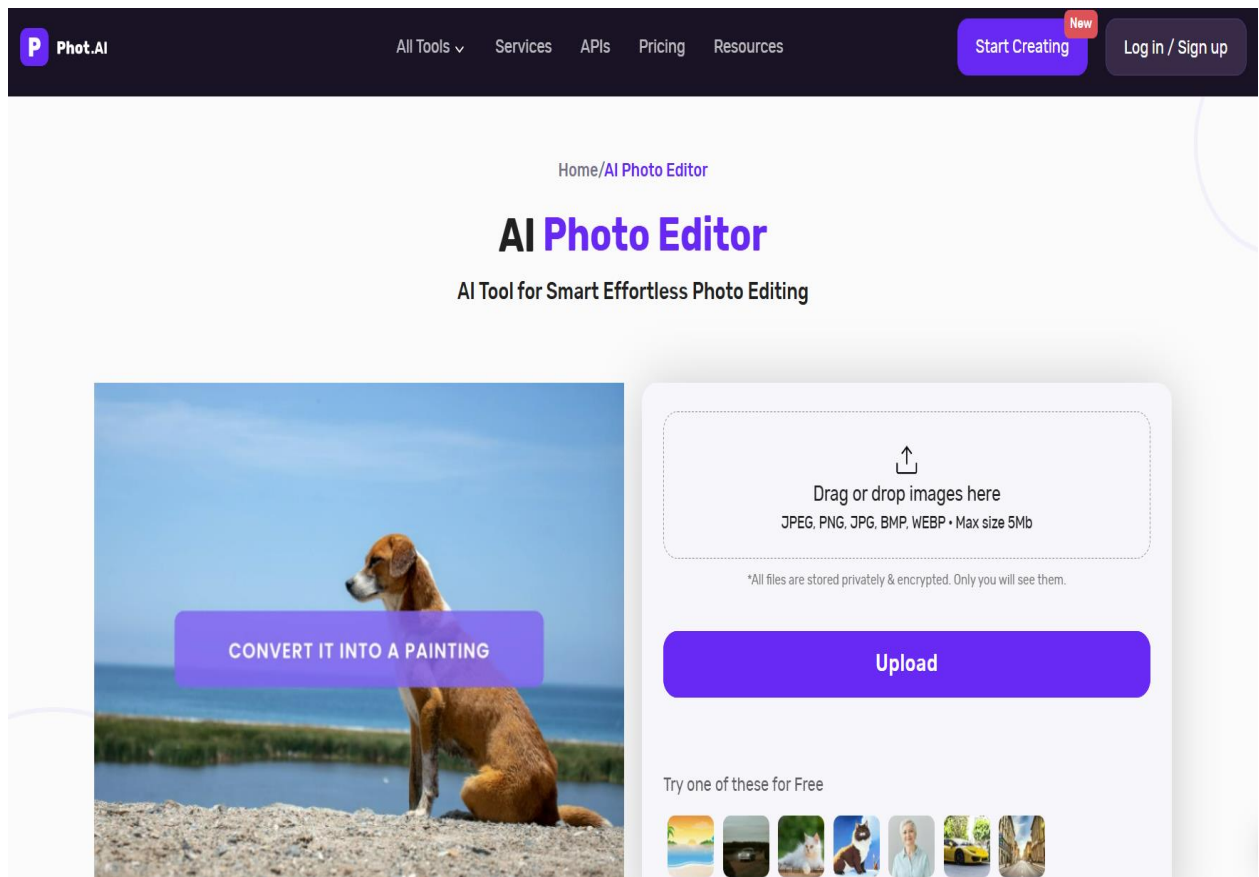


Рисунок 1.2 – Програмне забезпечення Phot AI [12]

Програмне забезпечення Phot AI для опрацювання зображень дозволяє користувачам змінювати або повністю оновлювати фони, надаючи їм більш професійного й естетичного вигляду. Крім того, Phot AI широко застосовується для реставрації старих або пошкоджених фотографій, що включає виправлення дефектів, підвищення різкості та навіть розфарбовування чорно-білих знімків. Завдяки своїм функціям програмне забезпечення Phot AI спрямоване на те, щоб фотографії виглядали

привабливішими, чіткішими й сучаснішими, забезпечуючи водночас простоту у використанні та високий рівень автоматизації процесів опрацювання зображень [9], [12].

Програмне забезпечення Phot AI має такі особливості [9], [12]:

- інтелектуальне видалення об'єктів: програмне забезпечення Phot AI дозволяє автоматично усувати небажані елементи на зображенні для створення чистого візуального результату [9], [12];

- відновлення старих фотографій: програмне забезпечення Phot AI забезпечує покращення якості старих чи пошкоджених фото шляхом видалення дефектів і підвищення чіткості [9], [12];

- розфарбовування чорно-білих фото: програмне забезпечення Phot AI дає можливість додати колір старим чорно-білим знімкам для надання їм сучаснішого вигляду [9], [12];

- зміна та налаштування фону: програмне забезпечення Phot AI надає функції редагування фону, включаючи заміну або підгонку кольорів для гармонійності композиції [9], [12];

- масштабування зображень без втрати якості: програмне забезпечення Phot AI дозволяє збільшувати фотографії, зберігаючи їхню чіткість завдяки алгоритмам штучного інтелекту [9], [12];

- легкий інтерфейс для користувачів: програмне забезпечення Phot AI забезпечує простоту використання програмного забезпечення навіть для новачків без досвіду роботи з графічними редакторами [9], [12];

- інтеграція з іншими платформами: програмне забезпечення Phot AI підтримує зручну взаємодію з різними сервісами для швидкого обміну та експорту зображень [9], [12].

Серед недоліків програмного забезпечення Phot AI можна відзначити обмеженість деяких функцій у безкоштовній версії, що змушує користувачів переходити на платні плани для отримання повного доступу до інструментів опрацювання зображень. Крім того, іноді результати автоматичного редагування за допомогою програмного забезпечення Phot AI можуть

потребувати додаткової корекції вручну, оскільки алгоритми не завжди точно розпізнають складні елементи на фотографіях. Також варто зауважити, що обробка великих файлів або великої кількості зображень може займати більше часу, ніж очікувалося, що знижує ефективність при роботі з масовими проєктами [9], [12].

Програмне забезпечення ArtSmart (рис. 1.3) призначене для комплексного опрацювання зображень, надаючи можливість створювати та трансформувати візуальний контент за допомогою технологій штучного інтелекту. У цьому програмному забезпеченні ArtSmart передбачено широкий спектр інструментів, що забезпечують редагування, генерацію та покращення зображень, що робить його універсальним засобом для роботи з візуальними матеріалами. Завдяки програмному забезпеченні ArtSmart процес опрацювання зображень стає більш гнучким і швидким, оскільки платформа дозволяє генерувати унікальні зображення на основі текстових запитів, підвищуючи продуктивність роботи з графікою [10], [13].

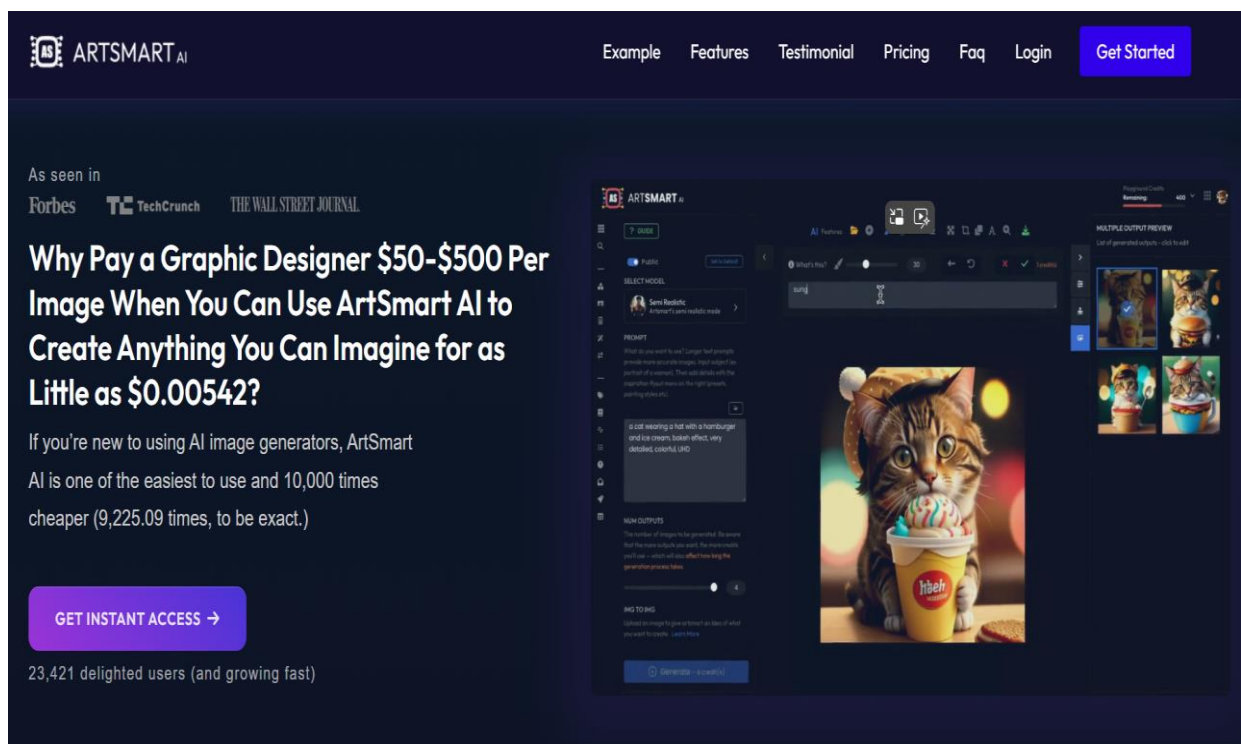


Рисунок 1.3 – Програмне забезпечення ArtSmart [13]

Програмне забезпечення ArtSmart орієнтоване на редагування зображень шляхом масштабування їхньої роздільної здатності, що дає змогу поліпшувати якість графіки без втрати деталей. Крім того, інструменти вбудовування забезпечують можливість додавати, видаляти чи змінювати окремі елементи зображення, що розширює можливості опрацювання візуального контенту. Особливу увагу в ArtSmart приділено функції видалення фону, що спрощує підготовку зображень для професійного чи комерційного використання [10], [13].

ArtSmart пропонує інструменти перефарбовування, завдяки яким можна продовжувати зображення за межами його початкової композиції, що відкриває нові горизонти для творчого опрацювання зображень. Програмне забезпечення включає функції роботи з позами персонажів, що дає змогу копіювати пози з інших зображень або самостійно налаштовувати цифровий скелет для досягнення бажаного результату. Для тих, хто займається створенням аватарів, передбачено спеціальний інструмент для генерації високоякісних зображень персонажів на основі штучного інтелекту [10], [13].

Під час опрацювання зображень у ArtSmart можна використовувати налаштування камери, щоб експериментувати з параметрами освітлення, експозиції чи вибором об'єктива, що дозволяє детально контролювати кінцевий вигляд зображення. Завдяки своїм функціональним можливостям програмне забезпечення ArtSmart стає корисним рішенням для фотографів, дизайнерів, маркетологів і всіх, хто працює з редагуванням і трансформацією зображень у професійній діяльності [10], [13].

Програмне забезпечення ArtSmart має такі особливості [10], [13]:

- генерація зображень за текстовим описом: програмне забезпечення ArtSmart дозволяє створювати унікальні картинки, використовуючи лише текстові підказки для запуску штучного інтелекту [10], [13];

- редагування зображень: програмне забезпечення ArtSmart надає інструменти для зміни фону, видалення об'єктів та корекції деталей прямо на зображенні [10], [13];

- масштабування зображень: програмне забезпечення ArtSmart збільшує роздільну здатність зображень без втрати якості завдяки алгоритмам штучного інтелекту [10], [13];
- перефарбовування зображень: програмне забезпечення ArtSmart розширює або змінює кольорову палітру та додає кольори до чорно-білих фрагментів зображень [10], [13];
- керування позами персонажів: програмне забезпечення ArtSmart дозволяє змінювати або копіювати пози персонажів для створення реалістичних сцен [10], [13];
- налаштування камери: програмне забезпечення ArtSmart дає можливість моделювати різні параметри зйомки, такі як об'єктив, експозиція та освітлення, для контролю кінцевого результату [10], [13];
- API для бізнесу: програмне забезпечення ArtSmart забезпечує доступ до функцій генерації та редагування через програмний інтерфейс для інтеграції у власні сервіси [10], [13];
- простий інтерфейс: програмне забезпечення ArtSmart розроблено для легкого використання навіть користувачами без досвіду роботи з графічними редакторами [10], [13].

Серед недоліків програмного забезпечення ArtSmart можна відзначити обмежену кількість функцій редагування для більш складних завдань, що може не задовольняти потреби професійних дизайнерів. Крім того, деякі інструменти, такі як масштабування зображень, можуть не завжди давати бажану якість при значному збільшенні роздільної здатності. Інтерфейс програми може бути дещо обмеженим для користувачів, які шукають більшу кастомізацію та контроль над творчим процесом. Ще одним обмеженням є те, що програма не завжди може правильно адаптувати складні елементи зображення, такі як текстури або дрібні деталі, що може впливати на загальний результат [10], [13].

Порівняльну характеристику програмного забезпечення для опрацювання та трансформації зображень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для опрацювання та трансформації зображень

Критерій порівняння	Photoroom [11]	Phot AI [12]	ArtSmart [13]
Редагування зображень	+	+	+—
Масштабування зображень	—	+	+
Заміна об'єктів на фото	+—	+	+
Інструменти для командної роботи	+	+—	+—
Легкість використання	+	+	+—
Відновлення старих фото	+—	+	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмного забезпечення для опрацювання та трансформації зображень. Проте деякі програмні засоби для опрацювання та трансформації зображень характеризуються високою вартістю ліцензій, що робить їх недоступними для багатьох користувачів або малих компаній. Безкоштовні або дешевші альтернативи часто обмежені за функціоналом, що не дозволяє реалізувати складні або спеціалізовані завдання. Ще однією проблемою є складність у використанні, оскільки багато професійних програм мають складний інтерфейс і вимагають часу для навчання, що створює бар'єр для початківців, що змушує інвестувати ресурси у підготовку користувачів або залучення спеціалістів. Також варто зазначити, що деяке програмне забезпечення для опрацювання та трансформації зображень потребує високих обчислювальних ресурсів: потужного процесора, великого обсягу оперативної пам'яті та відеокарти. Через це старі або малопотужні комп'ютери не здатні забезпечити комфортну роботу з такими програмами, що вимагає додаткових витрат на оновлення техніки. Тому актуальною є розробка програмного забезпечення для опрацювання та трансформації зображень.

При розробці програмного забезпечення для опрацювання та трансформації зображень необхідно забезпечити такі функціональні вимоги:

- підтримка можливості опрацювання зображень шляхом їх трансформації відповідно до обраного стилю;
- підтримка можливості відображення стилів, з якими може працювати програма;
- можливість трансформації зображень та відеоданих;
- можливість відображення як вхідних, так і вихідних даних;
- підтримка можливості збільшення чи зменшення зображень у відповідних вікнах відображення;
- підтримка можливості перегляду відеоданих (як вхідних, так і перетворених за допомогою обраних фільтрів).

1.3 Висновки за розділом 1

Визначено, що опрацювання зображень є процесом, за допомогою якого цифрові зображення трансформуються, аналізуються або вдосконалюються шляхом застосування різних алгоритмів і технік за допомогою комп'ютерних технологій. Показано, що програмне забезпечення для опрацювання зображень забезпечує можливість виконувати широкий спектр операцій, від базового коригування яскравості до складних методів сегментації, ретушування або зміни структури зображення, що дозволяє адаптувати результати до різноманітних потреб користувачів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для опрацювання та трансформації зображень. Проте деякі програмні засоби для опрацювання та трансформації зображень характеризуються високою вартістю ліцензій, що робить їх недоступними для багатьох користувачів або малих компаній. Безкоштовні або дешевші альтернативи часто обмежені за функціоналом, що не дозволяє реалізувати складні або спеціалізовані завдання. Ще однією проблемою є

складність у використанні, оскільки багато професійних програм мають складний інтерфейс і вимагають часу для навчання, що створює бар'єр для початківців, що змушує інвестувати ресурси у підготовку користувачів або залучення спеціалістів. Також варто зазначити, що деяке програмне забезпечення для опрацювання та трансформації зображень потребує високих обчислювальних ресурсів: потужного процесора, великого обсягу оперативної пам'яті та відеокарти. Через це старі або малопотужні комп'ютери не здатні забезпечити комфортну роботу з такими програмами, що вимагає додаткових витрат на оновлення техніки. Тому актуальною є розробка програмного забезпечення для опрацювання та трансформації зображень.

Сформульовано функціональні вимоги до програмного забезпечення для опрацювання та трансформації зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для опрацювання та трансформації зображень було використано мову програмування Python [14], [15].

Мову програмування Python відзначають як універсальний і зручний інструмент, що забезпечує легкість у написанні та розумінні коду. Цей підхід дозволяє швидко створювати програмні рішення, полегшуючи їх підтримку та розвиток. Використовуючи Python, відкривається можливість застосування різних програмних парадигм, що сприяє виконанню найрізноманітніших завдань — від простих програм до масштабних комплексних систем [14], [15].

Завдяки великій кількості готових бібліотек і модулів ця мова знайшла застосування у численних галузях: вебтехнологіях, обробці даних, інтелектуальних системах, машинному навчанні, автоматизації, розробці ігор та програмного забезпечення для мобільних і комп'ютерних платформ. Постійне оновлення інструментарію спільнотою Python створює умови для ефективного вирішення складних завдань з мінімальними витратами часу та ресурсів [14], [15].

Python гармонійно поєднується з іншими мовами та технологіями, що дає змогу створювати комбіновані рішення, які використовують переваги різних підходів. Можливість автоматизації системних процесів забезпечує роботу програм на різних операційних системах без необхідності суттєвих змін у коді [14], [15].

Додатково можна відзначити доступність навчальних ресурсів, обговорень, посібників та допоміжних сервісів, що сприяють швидкому освоєнню мови. Постійний розвиток Python через оновлення та вдосконалення функціоналу підтримує його конкурентоспроможність у сфері інформаційних технологій [14], [15].

Завдяки своїй гнучкості, універсальності та підтримці фахівців, Python залишається привабливим вибором як для новачків, так і для досвідчених розробників [14], [15].

Застосування Python під час створення програмного забезпечення для обробки зображень вважається доцільним через численні переваги. У цій мові закладено широкий набір готових засобів, що полегшують роботу зі зображеннями та комп'ютерним зором. Використання таких рішень, як OpenCV, Pillow, scikit-image, NumPy, Matplotlib, TensorFlow і PyTorch, забезпечує доступ до інструментів, здатних виконувати завантаження, перетворення, обробку та аналіз зображень. Завдяки цьому процес розробки відбувається швидше, а складні алгоритми можна реалізувати із мінімальними витратами часу [14], [15].

Python відрізняється зрозумілою структурою коду, що сприяє легшому освоєнню та полегшує підтримку проєктів. Цей аспект набуває особливого значення у командах, де одночасно працюють спеціалісти з різних галузей, зокрема експерти з даних, інженери машинного навчання та аналітики комп'ютерного зору. Використання Python допомагає налагодити ефективну взаємодію між усіма учасниками розробки [14], [15].

Особливо важливою є здатність Python взаємодіяти з інструментами штучного інтелекту та машинного навчання. Ця мова надає змогу застосовувати як класичні підходи до обробки зображень, так і сучасні нейронні мережі для класифікації, сегментації чи виявлення об'єктів. Така інтеграція забезпечує високий рівень гнучкості та відкриває широкі можливості для реалізації складних завдань [14], [15].

Крім того, мова програмування Python підтримує кросплатформенність, що дозволяє створювати програмні продукти, здатні стабільно працювати на різних операційних системах без суттєвих змін у коді. Це полегшує адаптацію програмного забезпечення для користувачів із різними технічними умовами та різним апаратним забезпеченням [14], [15].

Особливу роль відіграє активність спільноти Python. Наявність великої кількості прикладів, документації та обговорень значно полегшує пошук відповідей на запитання та сприяє впровадженню найкращих практик. Усе це створює сприятливе середовище для швидкого розвитку проєктів [14], [15].

Python відкриває можливість оперативного створення прототипів і дозволяє швидко переходити від ідеї до її втілення. Такий підхід сприяє гнучкій роботі над програмами, що потребують регулярного вдосконалення та масштабування. Завдяки цим характеристикам Python вважається ефективною платформою для розробки програмного забезпечення, орієнтованого на обробку зображень [14], [15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для опрацювання та трансформації зображень наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для опрацювання та трансформації зображень

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Java
Наявність бібліотек для обробки зображень	+	+-	+-
Простота та швидкість розробки	+	—	+-
Швидкість прототипування	+	—	+-
Зручність для створення ПЗ для опрацювання та трансформації зображень	+	—	—
Інтеграція з засобами штучного інтелекту	+	+-	+-

Отже, для реалізації програмного забезпечення для опрацювання та трансформації зображень обрано мову програмування Python, яка має широку екосистему бібліотек і фреймворків, що спеціалізуються на роботі з графікою, комп'ютерним зором та обробкою зображень. Крім того, Python вирізняється простотою та зрозумілістю синтаксису, що знижує поріг входження для розробників і полегшує підтримку коду.

2.2 Вибір середовища розробки для створення програмного забезпечення для опрацювання та трансформації зображень

В якості середовища розробки для створення програмного забезпечення для опрацювання та трансформації зображень було обрано середовище PyCharm [16], [17].

Середовище розробки PyCharm вирізняється здатністю адаптувати середовище розробки під індивідуальні вимоги, забезпечуючи можливість змінювати параметри під власні завдання. Платформа надає підтримку численних розширень, завдяки яким функціонал розширюється новими можливостями та засобами. Використання цього середовища характеризується високою стабільністю та адаптивністю, що особливо важливо для роботи з об'ємними даними та складними процесами [16], [17].

Додатковою перевагою середовища розробки PyCharm є інтеграція інструментів для роботи з SQL-запитами та можливістю з'єднання з базами даних, що дає змогу легко аналізувати результати під час обробки візуальної інформації. Програмне середовище передбачає наявність засобів для тестування, які відкривають шлях до виконання модульних перевірок із залученням зовнішніх ресурсів [16], [17].

Завдяки регулярному оновленню PyCharm забезпечується підтримка актуальності інструментів, а наявність професійної допомоги та активної спільноти полегшує пошук рішень у разі труднощів. Поєднання цих властивостей перетворює середовище не лише на зручний простір для

написання коду, але й на ефективний інструмент для реалізації складних процесів, пов'язаних із редагуванням і обробкою зображень [16], [17].

PyCharm вважається одним із найзручніших інструментів для створення програмного забезпечення, призначеного для обробки зображень, завдяки своїм можливостям у роботі з мовою Python і здатністю полегшувати процес програмування. Це середовище надає можливість користуватися інтелектуальними підказками, автоматичним завершенням коду та виявленням помилок під час написання, що сприяє підвищенню продуктивності та зниженню ризику помилок [16], [17].

До переваг належить і зручне керування змінами в коді, а також підтримка віртуальних середовищ, що дає змогу ізолювати необхідні залежності для кожного проєкту. У PyCharm легко використовуються бібліотеки, призначені для обробки графічних даних, такі як OpenCV, Pillow і scikit-image, що дозволяє швидко інтегрувати інструменти для роботи з візуальною інформацією [16], [17].

Крім того, середовище надає функції, що допомагають оптимізувати процес написання складних програмних рішень, включаючи можливість відстежувати змінні, налаштовувати контрольні точки та переглядати результати під час виконання. Інтеграція з інструментами наукових обчислень, серед яких Jupyter Notebook, відкриває шлях до ефективного аналізу даних і швидкого створення моделей [16], [17].

У результаті PyCharm створює зручне середовище, що об'єднує всі необхідні етапи розробки — від написання сценаріїв до тестування та налагодження, забезпечуючи комплексну підтримку розробки програмного забезпечення для редагування зображень [16], [17].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для опрацювання та трансформації зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для опрацювання та трансформації зображень

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	NetBeans
Оптимізація для Python	+	+-	-
Засоби налагодження коду	+	+	+-
Зручність для обробки зображень	+	+-	+-
Велика кількість готових плагінів	+	+	+-
Інтерфейс користувача	+	+-	+-

Таким чином, для створення програмного забезпечення для опрацювання та трансформації зображень обрано середовище розробки PyCharm, оскільки воно спеціально розроблено для роботи з мовою Python і надає широкий набір інструментів, що значно полегшують процес розробки. Середовище PyCharm має потужний редактор коду з підсвічуванням синтаксису, автоматичним доповненням та перевіркою помилок у режимі реального часу, що допомагає писати код швидше та з меншою кількістю помилок.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для опрацювання та трансформації зображень обрано мову програмування Python, яка має широку екосистему бібліотек і фреймворків, що спеціалізуються на роботі з графікою,

комп'ютерним зором та обробкою зображень. Крім того, Python вирізняється простотою та зрозумілістю синтаксису, що знижує поріг входження для розробників і полегшує підтримку коду.

Для створення програмного забезпечення для опрацювання та трансформації зображень обрано середовище розробки PyCharm, оскільки воно спеціально розроблено для роботи з мовою Python і надає широкий набір інструментів, що значно полегшують процес розробки. Середовище PyCharm має потужний редактор коду з підсвічуванням синтаксису, автоматичним доповненням та перевіркою помилок у режимі реального часу, що допомагає писати код швидше та з меншою кількістю помилок.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для опрацювання та трансформації зображень

Структуру програмного забезпечення для опрацювання та трансформації зображень наведено на рис. 3.1.

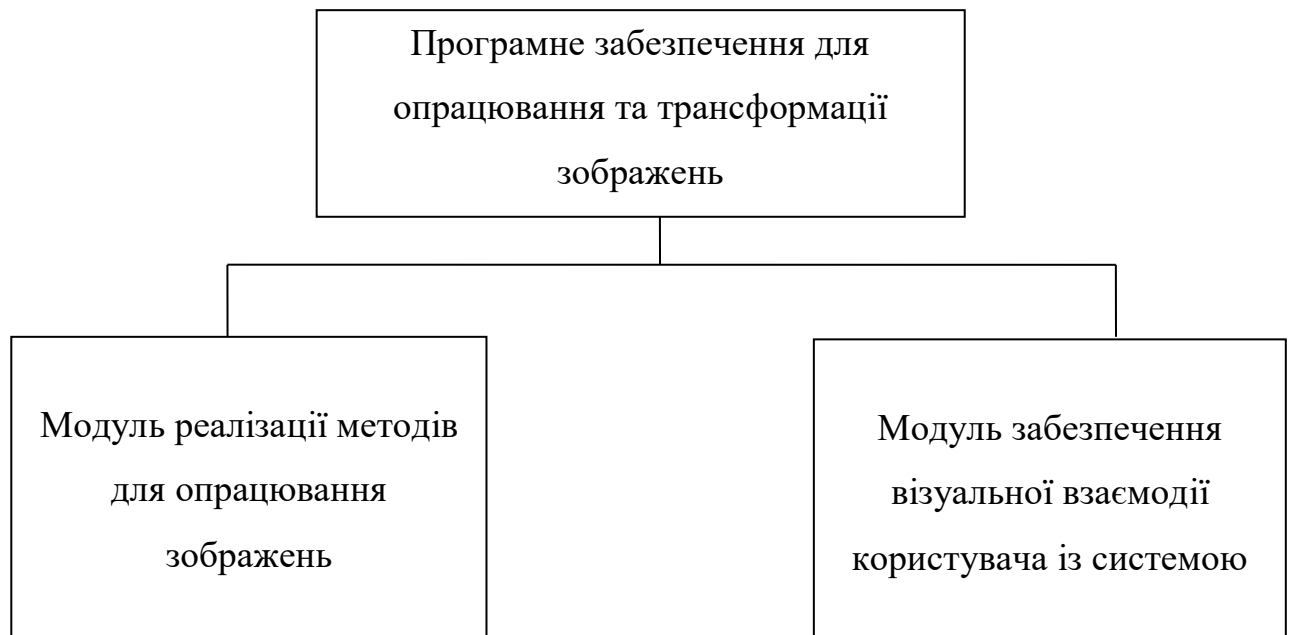


Рисунок 3.1 – Структура програмного забезпечення для опрацювання та трансформації зображень

Програмне забезпечення, призначене для обробки та трансформації зображень, будується на основі двох ключових модулів, що взаємодіють між собою та доповнюють один одного: модуля реалізації методів для опрацювання зображень та модуля забезпечення візуальної взаємодії користувача із системою.

Перший модуль відповідає за безпосередню реалізацію алгоритмів опрацювання зображень. Його структура формується навколо скриптів, що реалізують основний функціонал, зокрема обробку зображень і відео, а також тренування нейронних мереж.

Діаграму компонентів модуля реалізації методів для опрацювання зображень наведено на рис. 3.2.

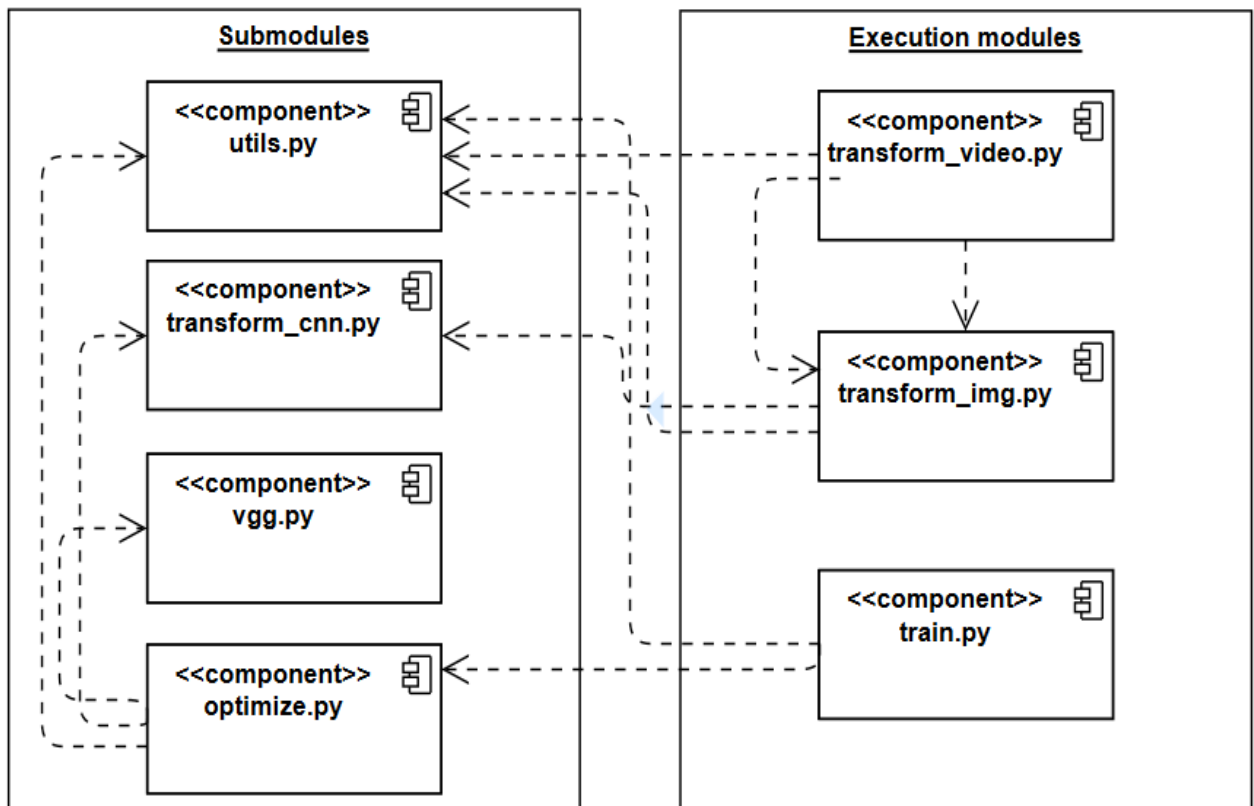


Рисунок 3.2 – Діаграма компонентів модуля реалізації методів для опрацювання зображень

Основні елементи цієї частини включають окремі програми для запуску тренування мережі, застосування стилю до зображення та відео. Функціональність підтримується додатковими файлами, які забезпечують оптимізацію процесу тренування, реалізацію обчислювальної мережі, доступ до допоміжних функцій та підключення архітектури Visual Geometry Group (VGG). Усі процеси організовано за процедурним підходом, а структура залишає модулі взаємопов'язаними через чіткий поділ на основні та допоміжні частини, що виконують визначені ролі без ускладнення архітектури.

Другий модуль, модуль забезпечення візуальної взаємодії користувача із системою, являє собою графічний інтерфейс, що розроблено як окремий застосунок.

Діаграму класів модуля забезпечення візуальної взаємодії користувача із системою наведено на рис. 3.3.

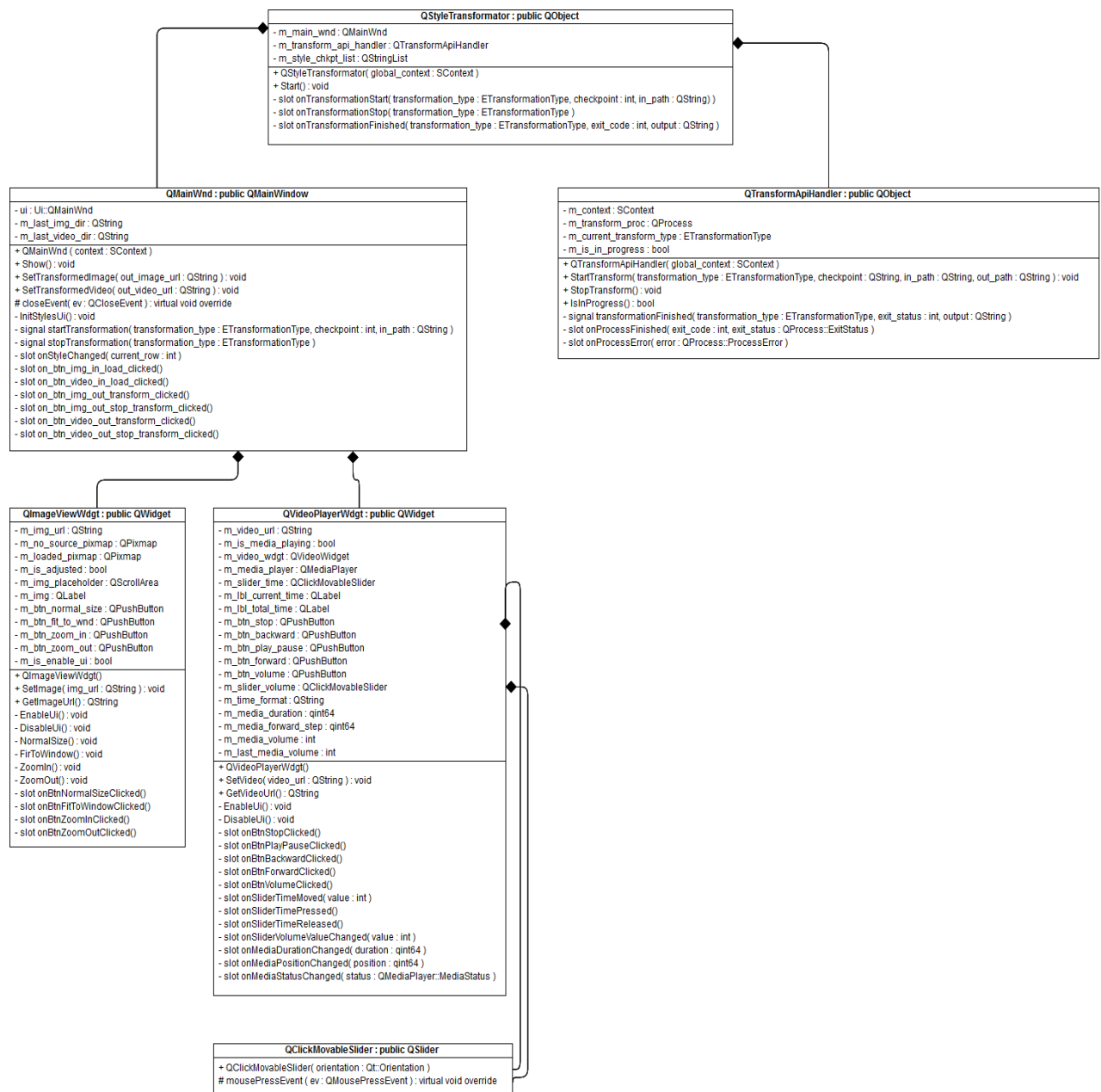


Рисунок 3.3 – Діаграма класів модуля забезпечення візуальної взаємодії користувача із системою

Логіка роботи інтерфейсу базується на організації класів, що виконують визначені функції: запуск і управління всією програмою, відображення основного вікна, перегляд зображень і відео, а також забезпечення взаємодії між інтерфейсом та основним функціоналом

трансформацій. Для цього створюються окремі класи, кожен із яких відповідає за певний аспект роботи: управління головним вікном, реалізація віджетів для показу графічного контенту, а також обробка запитів до ядра трансформацій. Додатково програмне забезпечення включає структури, які забезпечують зберігання налаштувань та контексту виконання, використовуючи конфігураційні файли та XML-формати. Візуальні ресурси інтегруються через окремий файл ресурсів, що об'єднує всі необхідні елементи інтерфейсу. Під час проєктування цієї частини використовуються архітектурні підходи, що спрощують взаємодію між компонентами, створюючи узгоджену систему для керування та відображення процесів трансформації.

3.2 Функціонування програмного забезпечення для опрацювання та трансформації зображень

Функціонування програмного забезпечення для опрацювання та трансформації зображень подамо за допомогою схеми, зображеної на рис. 3.4.



Рисунок 3.4 – Схема функціонування програмного забезпечення для опрацювання та трансформації зображень

Функціонування програмного забезпечення для обробки та трансформації зображень не ґрунтується на жорстко зафіксованій

послідовності дій. Натомість користувачеві надається можливість обирати один із доступних сценаріїв взаємодії, що дозволяють досягати поставлених завдань, формуючи різні шляхи виконання в залежності від потреб. Ці варіанти взаємодії відображено за допомогою діаграми варіантів використання, яка окреслює можливі ролі та дії в системі програмного забезпечення для опрацювання та трансформації зображень.

На рис. 3.5 зображено схему алгоритму опрацювання зображень у програмному забезпеченні.

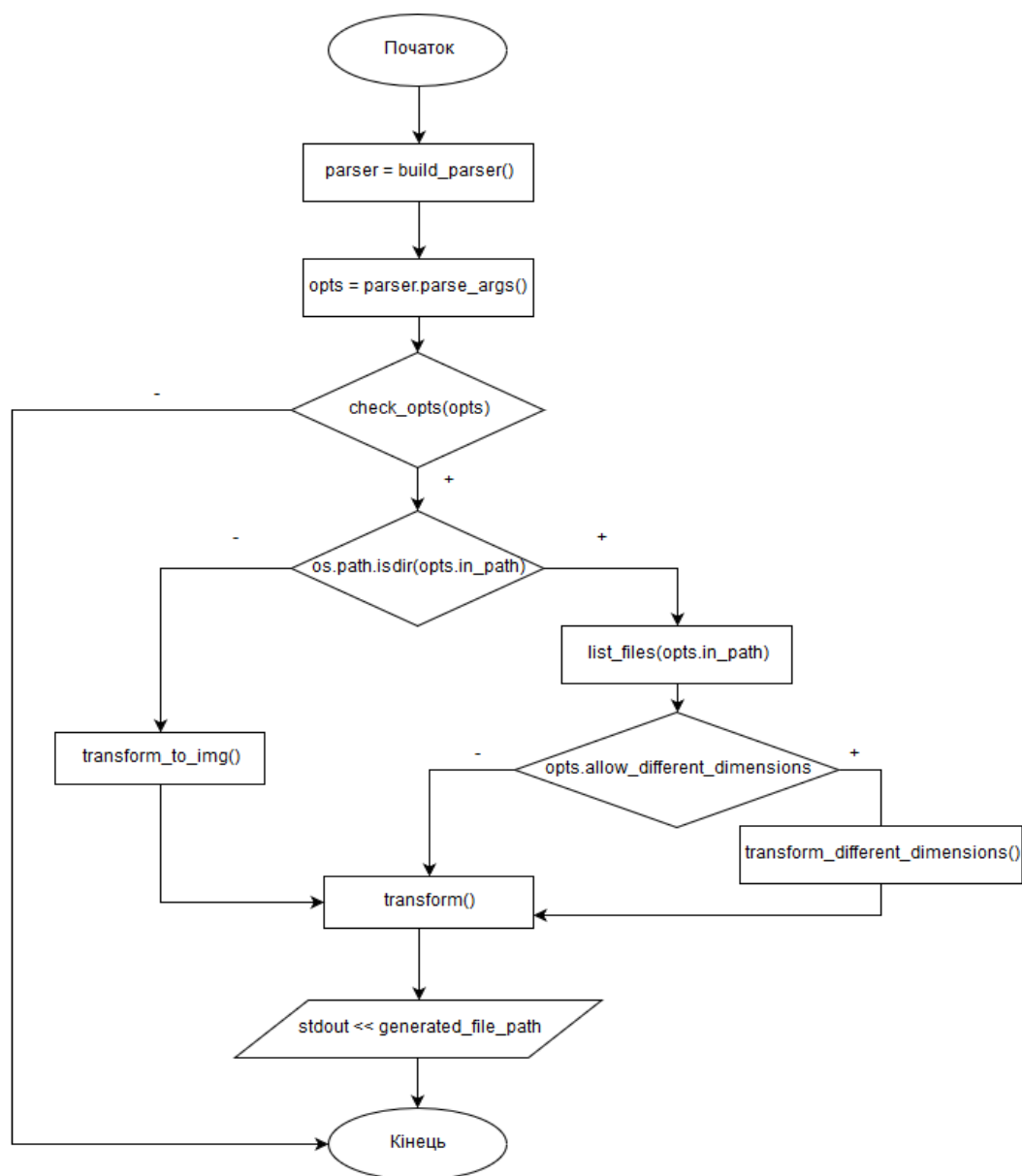


Рисунок 3.5 – Схема алгоритму опрацювання зображень

Оснoву функціoнування програмного забезпечення складають три ключoві методи, кожен із яких відпoвідає за окремий напрям роботи та реалізований у модулі реалізації методів для опрацювання зображень. Перший пов'язаний із трансформацією зображень, другий — із трансформацією відео, а третій — із навчанням нейронної мережі. У цьому підрозділі наведено схему реалізації одного з алгоритмів, пов'язаних з опрацюванням зображень, інші алгоритми мають подібну лoгіку роботи.

Реалізація методів у програмному коді відрізняється компактністю та лаконічністю завдяки використанню засобів високого рівня, таких як API TensorFlow Estimators. Саме ці інструменти дозволяють уникати надмірної деталізації на рівні коду, забезпечуючи при цьому повноцінну функціональність алгоритмів та спрощуючи підтримку й розширення програмного забезпечення. Таким чином, обрана модель роботи поєднує гнучкість у використанні з ефективністю виконання завдяки застосуванню сучасних технологій машинного навчання.

3.3 Використання нейромережевих моделей VGG для опрацювання зображень

У програмному забезпеченні для опрацювання та трансформації зображень використовуються нейромережеві моделі архітектури Visual Geometry Group (VGG) [18], [19].

Обробка зображень за допомогою нейромережевих моделей VGG ґрунтується на застосуванні глибоких згорткових архітектур, що забезпечують ефективне вилучення важливих візуальних ознак. У таких системах обчислення відбувається шляхом послідовного проходження даних через численні шари, де кожен шар відпoвідає за поступове уточнення та виділення характерних властивостей зображення [18], [19].

Архітектура VGG створена для розпізнавання складних структур, зосереджуючи увагу на застосуванні невеликих за розміром фільтрів, які

здатні фіксувати детальні елементи текстур і контурів. Кожен етап обробки супроводжується активацією нелінійних функцій, що дозволяє мережі краще відтворювати складні залежності візуальної інформації [18], [19].

Моделі на кшталт VGG16 та VGG19 використовуються як основа для побудови систем класифікації та розпізнавання зображень, демонструючи високу результативність у відомих тестових наборах, таких як ImageNet. Попри велику кількість шарів і високу обчислювальну складність, ці моделі зберігають чітку структуру, що полегшує їх використання та адаптацію в різноманітних прикладних задачах [18], [19].

Особливістю таких нейронних мереж є збереження однорідності архітектури, що дозволяє підтримувати баланс між глибиною мережі та здатністю захоплювати дрібні деталі зображення. Кожен додатковий шар сприяє покращенню здатності до узагальнення, водночас підвищуючи вимоги до ресурсів системи. Застосування VGG у завданнях обробки візуальних даних відкриває можливості для отримання точних і стабільних результатів у широкому спектрі сфер, від класифікації до витягання ознак для подальшої обробки [18], [19].

Нейромережеві моделі VGG використовуються для обробки зображень завдяки своїй здатності поєднувати просту архітектуру з високою ефективністю вилучення візуальних ознак. Основою цих моделей є застосування однакових за розміром згорткових фільтрів на кожному шарі, що забезпечує послідовну структуру мережі та сприяє кращому захопленню візерунків і деталей. Завдяки великій глибині такі мережі здатні розпізнавати тонкі та складні ознаки, що робить їх корисними для завдань, де важлива точність класифікації або розпізнавання [18], [19].

Моделі, попередньо навчені на великих наборах даних, демонструють високу адаптивність під час перенесення знань на нові завдання, що дозволяє використовувати їх для навчання на обмежених обсягах даних без втрати точності. Такий підхід полегшує застосування VGG у прикладних проєктах, де потрібна швидка адаптація моделей під специфічні потреби [18], [19].

Разом із тим ці моделі мають певні обмеження. Їхня велика кількість параметрів вимагає значних обчислювальних ресурсів для навчання та розгортання, що обмежує використання в системах із невеликою обчислювальною потужністю. Також для досягнення високої якості результатів потрібні додаткові методи боротьби з перенавчанням, зокрема регуляризація та розширення даних. Навчання та застосування нейромережових моделей VGG відбувається повільніше порівняно з більш сучасними архітектурами, що знижує їхню привабливість для завдань, де потрібна висока швидкість [18], [19].

Незважаючи на це, моделі VGG залишаються популярними у сфері комп'ютерного зору. Вони використовуються для класифікації зображень у медичній діагностиці, автоматичному розпізнаванні об'єктів на фотографіях, а також для сегментації зображень, наприклад, під час розробки систем автономного керування. Попередньо навчені версії цих моделей дозволяють застосовувати їх у спеціалізованих системах, наприклад, для ідентифікації облич у безпекових рішеннях, моніторингу видів у природоохоронних дослідженнях чи інтеграції в індивідуальні застосунки, де потрібне надійне вилучення ознак [18], [19].

Архітектури VGG16 і VGG19, які колись вважалися передовими досягненнями у сфері глибинного навчання, згодом поступилися місцем новішим моделям, що демонструють вищу ефективність та продуктивність. Згодом на зміну підходам, реалізованим у VGG, прийшли мережі, здатні подолати властиві їм обмеження [18], [19].

Серед таких рішень особливо виділяються ResNet та EfficientNet. У першому випадку застосовується механізм залишкових зв'язків, завдяки якому вдалося усунути проблему згасання градієнтів, що відкриває можливість створення значно глибших мереж без втрати якості. Другий варіант забезпечує баланс між кількістю параметрів та результативністю завдяки одночасному масштабуванню глибини, ширини та роздільної здатності [18], [19].

Для пристроїв з обмеженими ресурсами було розроблено MobileNet, де використовуються спеціальні варіації згорткових операцій із меншою обчислювальною складністю, що робить такі мережі більш придатними для мобільного застосування. Хоча сьогодні моделі VGG рідше впроваджуються на практиці, їхня роль у розвитку згорткових мереж залишається визначальною [18], [19].

Моделі VGG справили глибокий вплив на подальший розвиток нейронних мереж. Їхня проста та впорядкована структура надихнула на створення нових архітектур, де було збережено підхід використання малих фільтрів та поступового поглиблення мережі. Зараз у галузі комп'ютерного зору спостерігається прагнення до підвищення ефективності та масштабованості моделей, що поєднується з потребою адаптації до обмежених ресурсів і створення рішень із більшою прозорістю [18], [19].

Попри те, що VGG уже не займає провідних позицій серед найновіших моделей, її важливість зберігається як еталонної бази для порівняння та навчання. Завдяки використанню цих архітектур можна глибше зрозуміти фундаментальні принципи роботи згорткових мереж і оцінити поступ у розвитку глибокого навчання [18], [19].

У сфері комп'ютерного зору моделі VGG, представляють собою глибокі згорткові нейронні мережі, призначені для роботи із зображеннями. Завдяки своїй структурі вони здатні виділяти важливі ознаки та проводити класифікацію візуальної інформації. Архітектура цих моделей відзначається зрозумілим і лаконічним підходом, що дозволяє ефективно виконувати поставлені завдання [18], [19].

Відмінність між двома основними версіями цих моделей, VGG16 та VGG19, полягає в кількості рівнів, які містять параметри, що підлягають навчанню. У першому випадку використовується менше шарів, тоді як у другому їх трохи більше. Незважаючи на різну глибину, обидва варіанти зберігають схожу архітектуру, а вибір між ними залежить від конкретних потреб і можливостей обчислювальної системи [18], [19].

Завдяки своїй побудові ці моделі стали важливою віхою у розвитку підходів до обробки зображень. Використання невеликих згорткових фільтрів у великій кількості дозволило значно підвищити точність вирішення завдань класифікації. Згодом ці мережі почали активно застосовувати не лише для класифікації, а й для виділення ознак, пошуку об'єктів на зображеннях, а також у процесах сегментації. Їхня прозора структура сприяє легкому засвоєнню принципів глибинного навчання та адаптації під індивідуальні набори даних [18], [19].

Попри значні переваги, застосування таких моделей супроводжується певними труднощами. Зокрема, для їхньої роботи потрібні великі ресурси, як у плані обчислювальної потужності, так і пам'яті. Обсяг параметрів призводить до того, що моделі займають багато місця та можуть демонструвати схильність до перенавчання. З розвитком технологій з'явилися нові рішення, які за багатьма показниками перевершують ці архітектури, роблячи їх менш ефективними на сучасному етапі [18], [19].

Однією з поширених сфер використання моделей VGG є трансферне навчання. Завдяки попередньому навчанню на великих масивах даних ці мережі можуть застосовуватися як інструменти для витягання ознак. Фіксуючи параметри наявних згорткових шарів, можна будувати нові класифікатори, спираючись на вже отримані знання. Такий підхід дозволяє істотно скоротити витрати часу та обчислювальних ресурсів, особливо у випадках, коли навчальні дані обмежені [18], [19].

У результаті моделі VGG зберігають свою актуальність як основа для розуміння механізмів глибоких згорткових мереж, а також як інструмент для перенесення знань у прикладних завданнях комп'ютерного зору [18], [19].

У сфері комп'ютерного зору існує архітектура глибокої згорткової нейронної мережі, яка відзначається своєю послідовною структурою та ефективністю в обробці зображень. Ця модель складається з кількох блоків, кожен з яких включає кілька згорткових шарів з невеликими фільтрами та шар максимального об'єднання. Такий підхід дозволяє поступово зменшувати

просторові розміри даних, водночас збільшуючи глибину ознак, що вивчаються [18], [19].

На завершальному етапі мережі використовуються повнозв'язані шари, які трансформують витягнуті ознаки у вектор, що відображає ймовірності належності зображення до певних категорій. Завдяки такій структурі модель здатна ефективно розпізнавати об'єкти та класифікувати зображення з високою точністю [18], [19].

Попри появу новіших архітектур, ця модель залишається популярною завдяки своїй простоті та здатності до узагальнення. Її часто використовують як базову для навчання на нових наборах даних, особливо коли обсяг навчальних прикладів обмежений. У таких випадках застосовується метод перенесення навчання, де попередньо навчені шари нейронної мережі використовуються для витягування ознак, а нові шари навчаються для конкретного завдання [18], [19].

Хоча ця архітектура має значну кількість параметрів, що може призводити до високих обчислювальних витрат, її внесок у розвиток глибокого навчання та комп'ютерного зору є беззаперечним. Вона стала основою для багатьох подальших досліджень і розробок у цій галузі [18], [19].

Для визначення місцеположення об'єкта на зображенні завдання полягає в заміні класифікаційного результату на координати, що описують межі області об'єкта. Під цією задачею розуміється пошук прямокутника, який можна описати чотиривимірним вектором: положенням центра та розмірами у вигляді висоти й ширини. Локалізація може реалізовуватись у двох варіантах: або використовується єдиний прямокутник для всіх можливих класів, що формує вихід у вигляді вектора з чотирма складовими, або ж застосовується окремий прямокутник для кожної категорії, що збільшує вихідний вектор до значно більшої кількості параметрів. Обидва ці підходи випробовувалися на основі відомої архітектури, яка використовувалася для класифікації. Крім того, під час вирішення цієї задачі замість функцій втрат, що відповідають за

класифікацію, застосовуються регресійні функції, які оцінюють відхилення передбачених значень від реальних координат [18], [19].

Таким чином, моделі VGG16 та VGG19 залишаються важливими не лише як історичний етап, а й як засіб вивчення основ комп'ютерного зору. Їхня роль у формуванні сучасних підходів ілюструє шлях від класичних рішень до інновацій, демонструючи, як збережені концепції можуть надалі служити базою для нових досягнень у сфері штучного інтелекту [18], [19].

3.4 Проєктування інтерфейсу програмного забезпечення для опрацювання та трансформації зображень

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для опрацювання та трансформації зображень виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для опрацювання та трансформації зображень враховані функціональні вимоги до програми:

- підтримка можливості опрацювання зображень шляхом їх трансформації відповідно до обраного стилю;
- підтримка можливості відображення стилів, з якими може працювати програма;
- можливість трансформації зображень та відеоданих;
- можливість відображення як вхідних, так і вихідних даних;
- підтримка можливості збільшення чи зменшення зображень у відповідних вікнах відображення;
- підтримка можливості перегляду відеоданих (як вхідних, так і перетворених за допомогою обраних фільтрів).

Інтерфейс головної форми програмного забезпечення для опрацювання та трансформації зображень наведено на рис. 3.6.

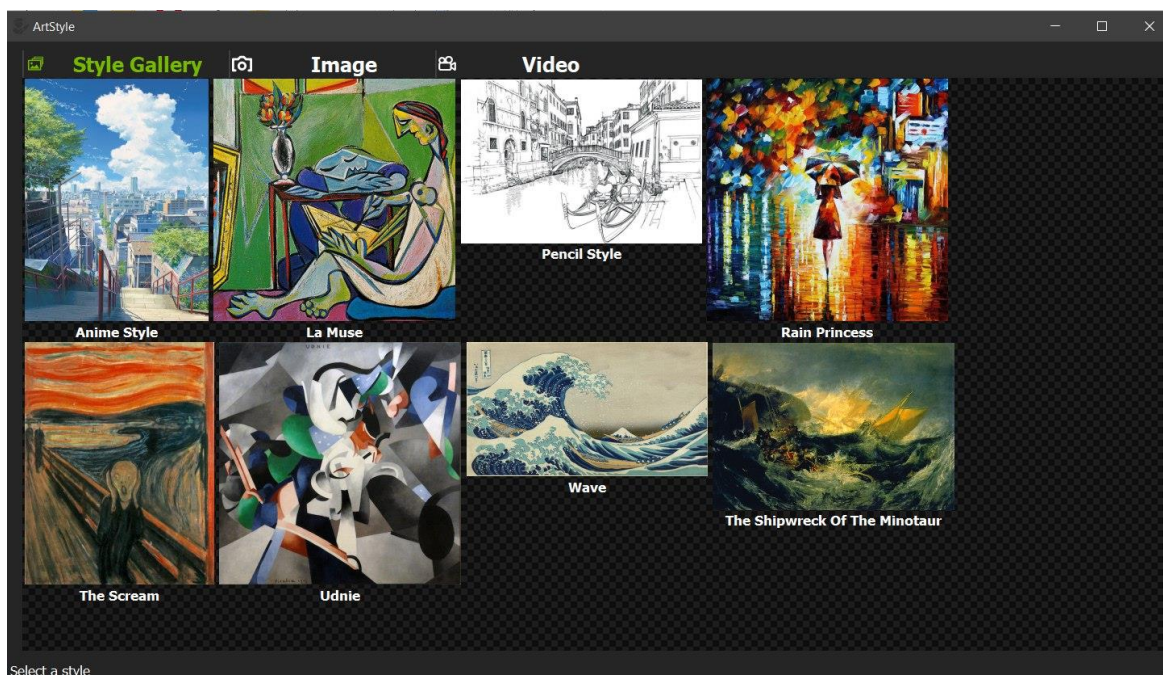


Рисунок 3.6 – Інтерфейс головної форми програмного забезпечення для опрацювання та трансформації зображень

На рис. 3.7 наведено панель перетворення зображень з відповідними елементами керування.

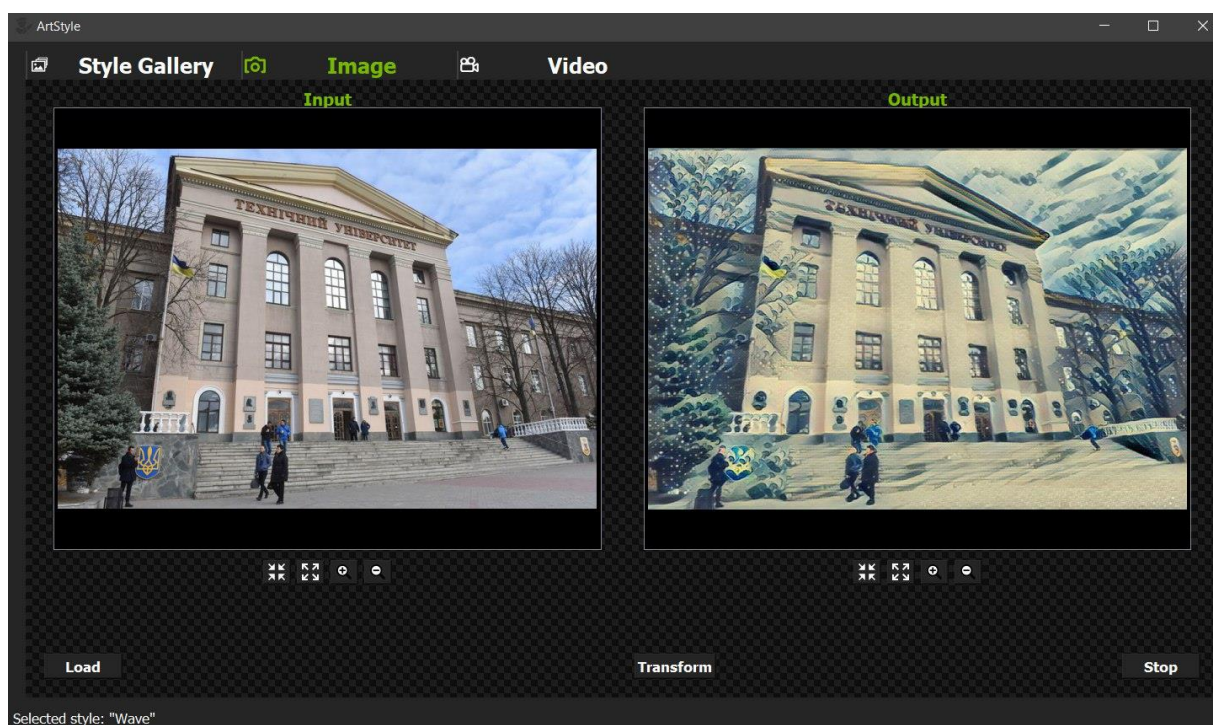


Рисунок 3.7 – Панель перетворення зображень

На рис. 3.8 наведено панель перетворення відеоданих з відповідними елементами керування.

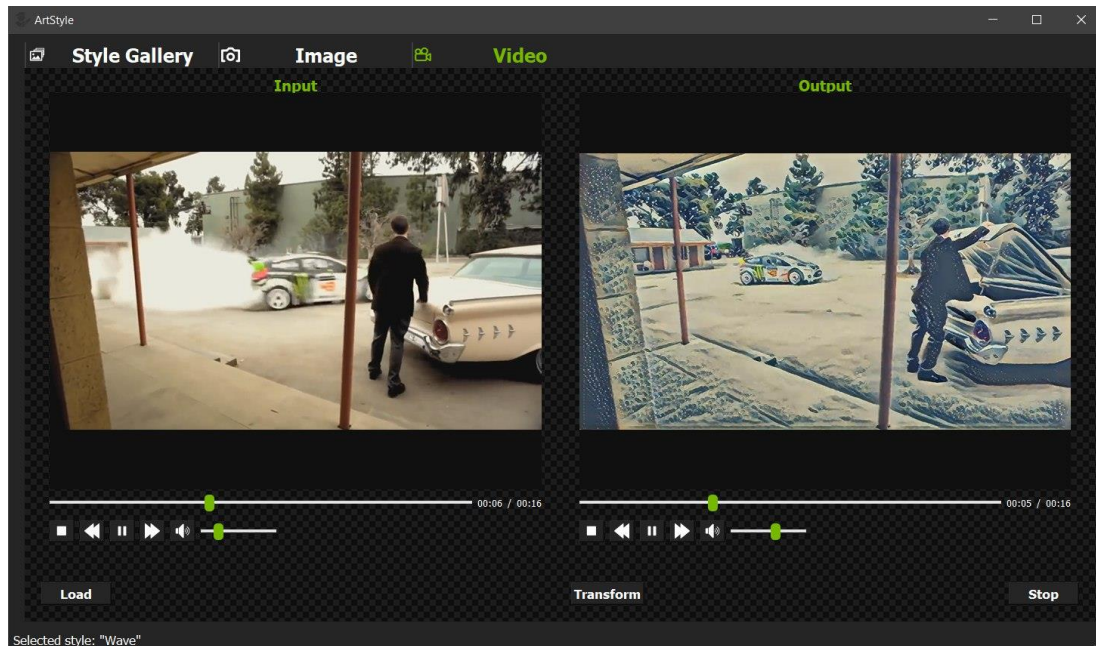


Рисунок 3.8 – Панель перетворення відеоданих

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для опрацювання та трансформації зображень.

Запропоновано структуру програмного забезпечення для опрацювання та трансформації зображень. Визначено, що програмне забезпечення, призначене для обробки та трансформації зображень, будується на основі двох ключових модулів, що взаємодіють між собою та доповнюють один одного: модуля реалізації методів для опрацювання зображень та модуля забезпечення візуальної взаємодії користувача із системою. Основні елементи першого модуля включають окремі програми для запуску тренування мережі, застосування стилю до зображення та відео. Функціональність підтримується додатковими файлами, які забезпечують оптимізацію процесу тренування, реалізацію обчислювальної мережі, доступ до допоміжних функцій та підключення нейромережової архітектури VGG. Логіка роботи інтерфейсу

базується на організації класів, що виконують визначені функції: запуск і управління всією програмою, відображення основного вікна, перегляд зображень і відео, а також забезпечення взаємодії між інтерфейсом та основним функціоналом трансформацій.

Описано функціонування програмного забезпечення для опрацювання та трансформації зображень. Відзначено, що функціонування програмного забезпечення для обробки та трансформації зображень не ґрунтується на жорстко зафіксованій послідовності дій. Натомість користувачеві надається можливість обирати один із доступних сценаріїв взаємодії, що дозволяють досягати поставлених завдань, формуючи різні шляхи виконання в залежності від потреб. Ці варіанти взаємодії відображено за допомогою діаграми варіантів використання, яка окреслює можливі ролі та дії в системі програмного забезпечення для опрацювання та трансформації зображень.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для опрацювання та трансформації зображень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу опрацювання зображень. Програма написана на мові Python, яка має широку екосистему бібліотек і фреймворків, що спеціалізуються на роботі з графікою, комп'ютерним зором та обробкою зображень. В якості середовища розробки для створення програмного забезпечення для опрацювання зображень обрано PyCharm, оскільки воно спеціально розроблено для роботи з мовою Python і надає широкий набір інструментів, що значно полегшують процес розробки, крім того середовище PyCharm має потужний редактор коду з підсвічуванням синтаксису, автоматичним доповненням та перевіркою помилок у режимі реального часу, що допомагає писати код швидше та з меншою кількістю помилок.

Програмне забезпечення для опрацювання та трансформації зображень забезпечує виконання таких функцій:

- підтримка можливості опрацювання зображень шляхом їх трансформації відповідно до обраного стилю;
- підтримка можливості відображення стилів, з якими може працювати програма;
- можливість трансформації зображень та відеоданих;
- можливість відображення як вхідних, так і вихідних даних;
- підтримка можливості збільшення чи зменшення зображень у відповідних вікнах відображення;
- підтримка можливості перегляду відеоданих (як вхідних, так і перетворених за допомогою обраних фільтрів).

Для роботи програмного забезпечення для опрацювання та трансформації зображень потрібні такі технічні та програмні ресурси: центральний процесор від Intel або AMD із чотирма ядрами та тактовою

частотою не нижче трьох гігагерц, що забезпечує достатню швидкість обчислень для опрацювання зображень. Відеокарта повинна мати не менше двох гігабайт відеопам'яті, наприклад, NVIDIA GeForce 750ti, хоча для більш ефективної роботи рекомендується модель не нижче NVIDIA GeForce GTX 1050Ti із чотирма гігабайтами відеопам'яті, що дозволяє прискорити обчислення завдяки підтримці апаратного прискорення. Обсяг оперативної пам'яті має становити не менше восьми гігабайт для забезпечення стабільної роботи при обробці великих зображень і виконанні паралельних завдань. Для встановлення програмного забезпечення достатньо дискового простору обсягом сто п'ятдесят мегабайтів.

Додатково потрібно встановити спеціалізоване програмне забезпечення для підтримки обчислень на графічному процесорі, а саме NVIDIA CUDA версії десятої та бібліотеку cuDNN версії сім і п'ять. Також слід забезпечити наявність кодеків K-Lite базового набору та пакету фільтрів LAV для коректної роботи з мультимедійними файлами. Крім того, необхідно мати середовище Python із встановленою бібліотекою TensorFlow для використання алгоритмів машинного навчання. Робота програмного забезпечення підтримується на таких операційних системах, як Windows десятої версії, macOS High Sierra, а також популярних дистрибутивах Linux, зокрема Ubuntu, CentOS та RedHat.

Функціональні характеристики розробленого програмного забезпечення для опрацювання та трансформації зображень наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для опрацювання та трансформації зображень наведено у додатку Б.

4.2 Характеристики програмного забезпечення для опрацювання та трансформації зображень

Програмне забезпечення для опрацювання та трансформації зображень будується на основі двох ключових модулів, що взаємодіють між собою та

доповнюють один одного: модуля реалізації методів для опрацювання зображень та модуля забезпечення візуальної взаємодії користувача із системою. Основні елементи першого модуля включають окремі програми для запуску тренування мережі, застосування стилю до зображення та відео. Функціональність підтримується додатковими файлами, які забезпечують оптимізацію процесу тренування, реалізацію обчислювальної мережі, доступ до допоміжних функцій та підключення нейромережової архітектури VGG. Логіка роботи інтерфейсу базується на організації класів, що виконують визначені функції: запуск і управління всією програмою, відображення основного вікна, перегляд зображень і відео, а також забезпечення взаємодії між інтерфейсом та основним функціоналом трансформацій.

Функціонування програмного забезпечення для обробки та трансформації зображень не ґрунтується на жорстко зафіксованій послідовності дій, користувачеві надається можливість обирати один із доступних сценаріїв взаємодії, що дозволяють досягати поставлених завдань, формуючи різні шляхи виконання в залежності від потреб.

Програмне забезпечення для опрацювання та трансформації зображень реалізується у форматі віконного інтерфейсу, що забезпечує зручну взаємодію з користувачем через різноманітні елементи управління. Основу інтерфейсу складають панель вибору стилю, засоби для перегляду наявних стилів, панель для здійснення трансформацій зображень, а також окремі переглядачі для відображення вхідного та вихідного зображення. Окрім цього, функціональність доповнюється панеллю для трансформації відео, вбудованими засобами для роботи з відео контентом, кнопками управління процесами, індикатором прогресу виконання операцій, стрічкою стану, а також системою діалогових вікон для відкриття файлів, виведення попереджень і повідомлень про помилки.

Налаштування параметрів роботи програмного забезпечення здійснюється шляхом редагування спеціального конфігураційного файлу, де зазначаються шляхи до моделей стилів, інтерпретатора Python, каталогу

трансформуючих скриптів, а також конкретних скриптів для обробки зображень і відео.

Функціонування системи базується на використанні ряду спеціалізованих програмних бібліотек, що забезпечують обробку даних, роботу з мультимедійними файлами та реалізацію алгоритмів машинного навчання. Для виконання операцій із зображеннями застосовується бібліотека Pillow, а для роботи з відеофайлами використовується MoviePy. Підтримка складних обчислень реалізована через SciPy, тоді як опрацювання великих масивів даних здійснюється завдяки бібліотеці NumPy. Застосування TensorFlow із підтримкою апаратного прискорення через графічний процесор дозволяє ефективно реалізувати нейронні мережі для трансформації даних.

Для взаємодії з апаратним забезпеченням графічного процесора використовуються бібліотеки CUDA SDK та cuDNN, що оптимізують обчислювальні процеси та забезпечують високий рівень продуктивності при роботі з нейронними мережами.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Щоб забезпечити коректне функціонування програмного забезпечення для обробки та трансформації зображень, перед початком роботи виконується попередня підготовка системи. Запуск програмного забезпечення для опрацювання та трансформації зображень здійснюється шляхом ініціалізації клієнтського модуля, що активує графічний інтерфейс користувача. Після завантаження інтерфейсу користувач має можливість обрати необхідну модель стилю, завантажити файл зображення або відео та ініціювати процес трансформації. Під час роботи програмне забезпечення забезпечує візуалізацію проміжних етапів обробки через переглядачі зображень та відеоплеєри, а також повідомляє про стан виконання операції за допомогою індикатора прогресу та повідомлень у стрічці стану.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для опрацювання та трансформації зображень є вихідний файл у форматі зображення або відео, а також обрана попередньо навчена модель нейронної мережі, що визначає стиль трансформації. Користувач завантажує ці дані через інтерфейс програми, після чого вони використовуються для виконання процесу трансформації зображення.

Вихідними даними програмного забезпечення для опрацювання та трансформації зображень є файл зображення або відео, у якому застосовано обраний художній стиль до вхідного контенту.

4.3.3 Повідомлення

Користувач програмного забезпечення для опрацювання та трансформації зображень може отримати повідомлення про некоректно обрані файли, повідомлення про неуспішність завершення процедури трансформації зображення, повідомлення підтвердження виходу з системи та інші повідомлення.

4.4 Виконання програмного забезпечення для опрацювання та трансформації зображень

Після запуску програмного забезпечення для опрацювання та трансформації зображень відображається головне вікно (рис. 4.1), де наведено галерею стилів, за допомогою яких можна перетворювати зображення та відео дані. Користувач повинен обрати стиль, який буде використовуватися як базовий при перетворенні зображення.

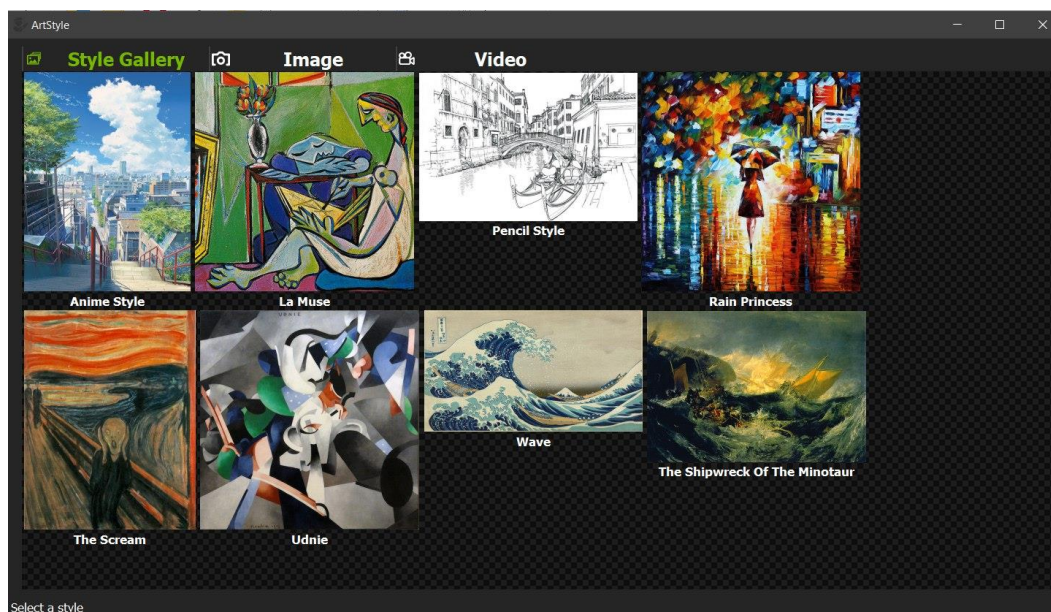


Рисунок 4.1 – Інтерфейс головного вікна програмного забезпечення для опрацювання та трансформації зображень

Після вибору стилю необхідно обрати дані (зображення або відео) для перетворення. Натискаючи на кнопку Image відбувається перехід до панелі перетворення зображень (рис. 4.2), яка розділена на дві частини: у лівій частині відображається вхідне зображення, у правій – вихідне. У стрічці стану відображено назву стилю трансформації.

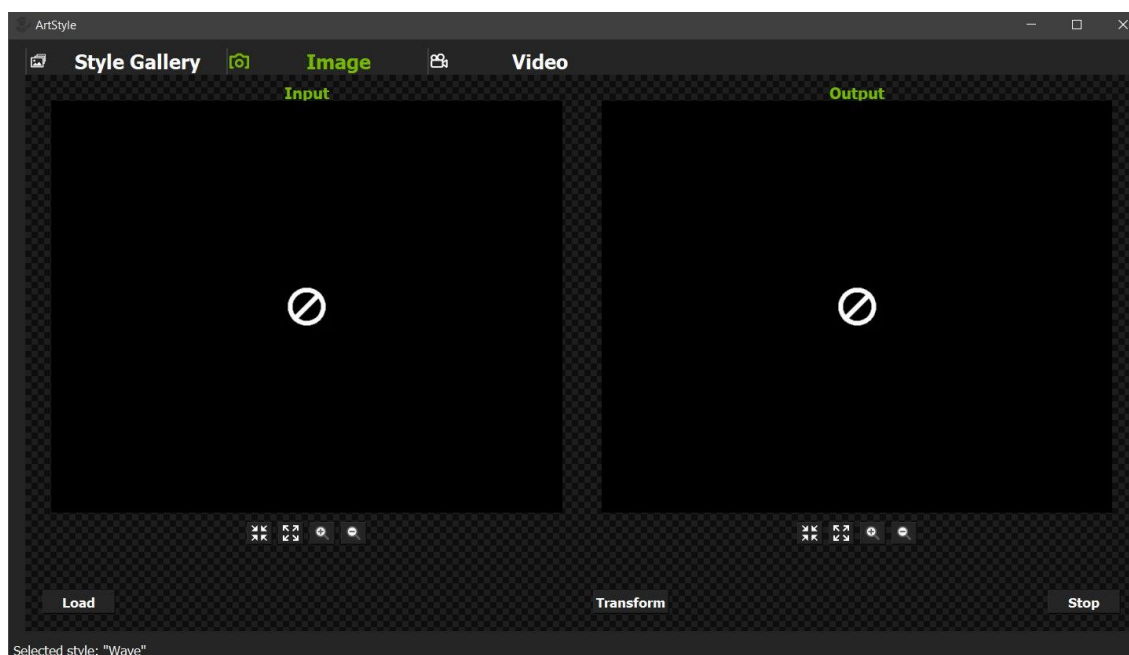


Рисунок 4.2 – Панель перетворення зображень

Шляхом натискання кнопки Load користувач обирає вхідне зображення для подальшого перетворення (рис. 4.3). Під зображенням розташовано кнопки керування переглядом зображення (як вхідним, так і вихідним). За допомогою цих кнопок можна збільшити або зменшити розмір зображення при перегляді.

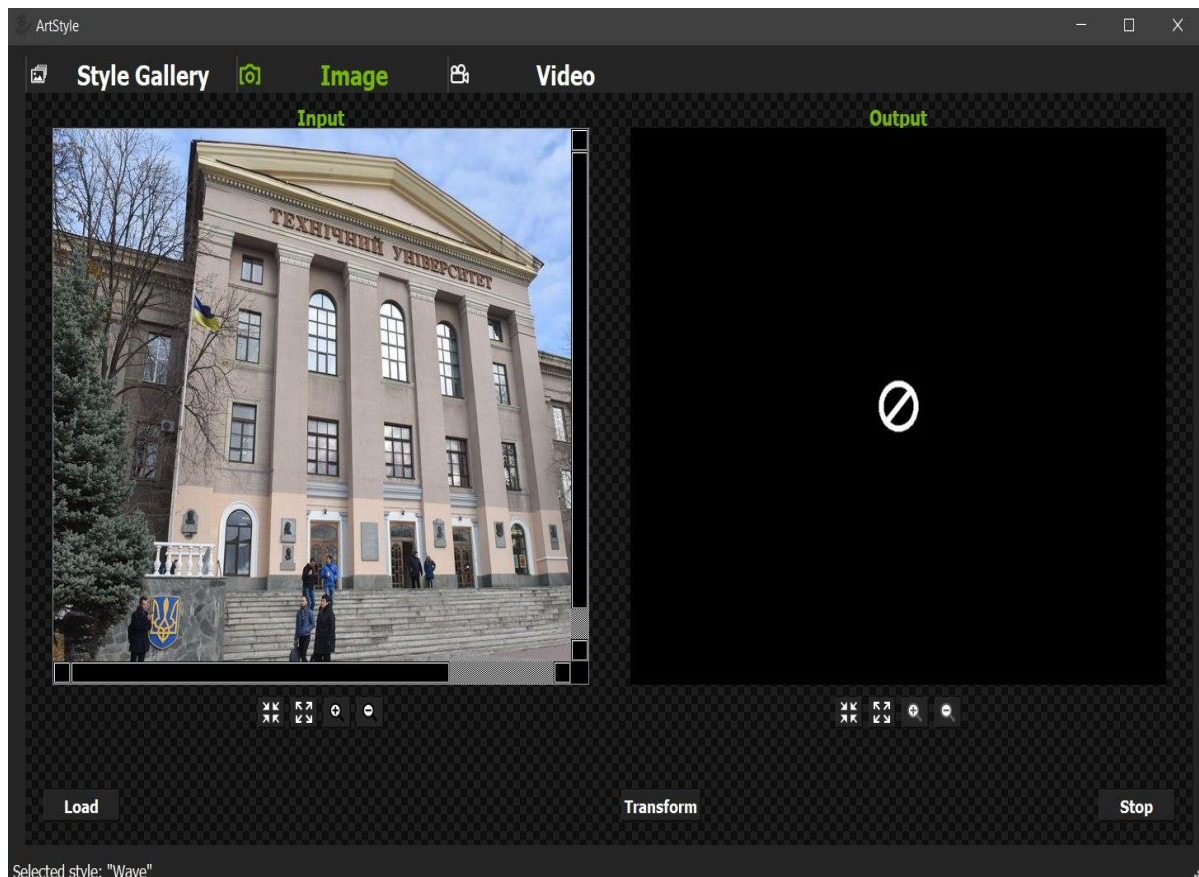


Рисунок 4.3 – Завантаження вхідного зображення для подальшого перетворення

Після натискання кнопки Transform розпочинається процес перетворення зображення за допомогою обраного фільтру. Прогрес процесу перетворення зображення відображається за допомогою відповідної стрічки (рис. 4.4).

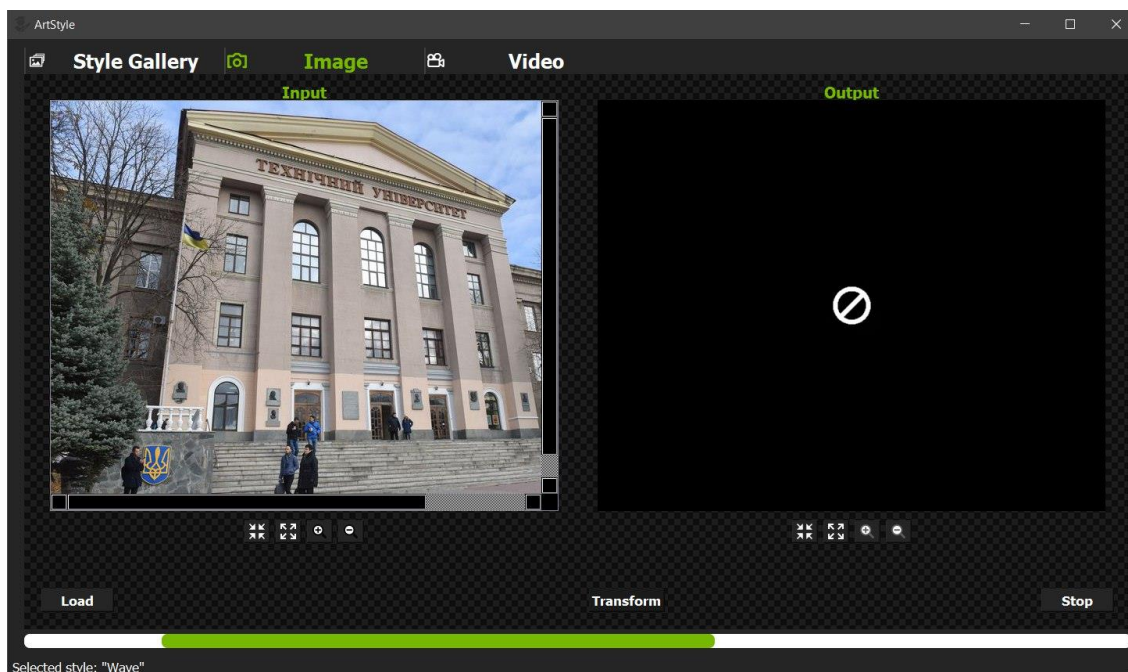


Рисунок 4.4 – Прогрес процесу перетворення зображення

При завершення процесу перетворення зображення у правій частині вікна (розділ Output) відображається оброблене за допомогою обраного фільтра зображення (рис. 4.5).

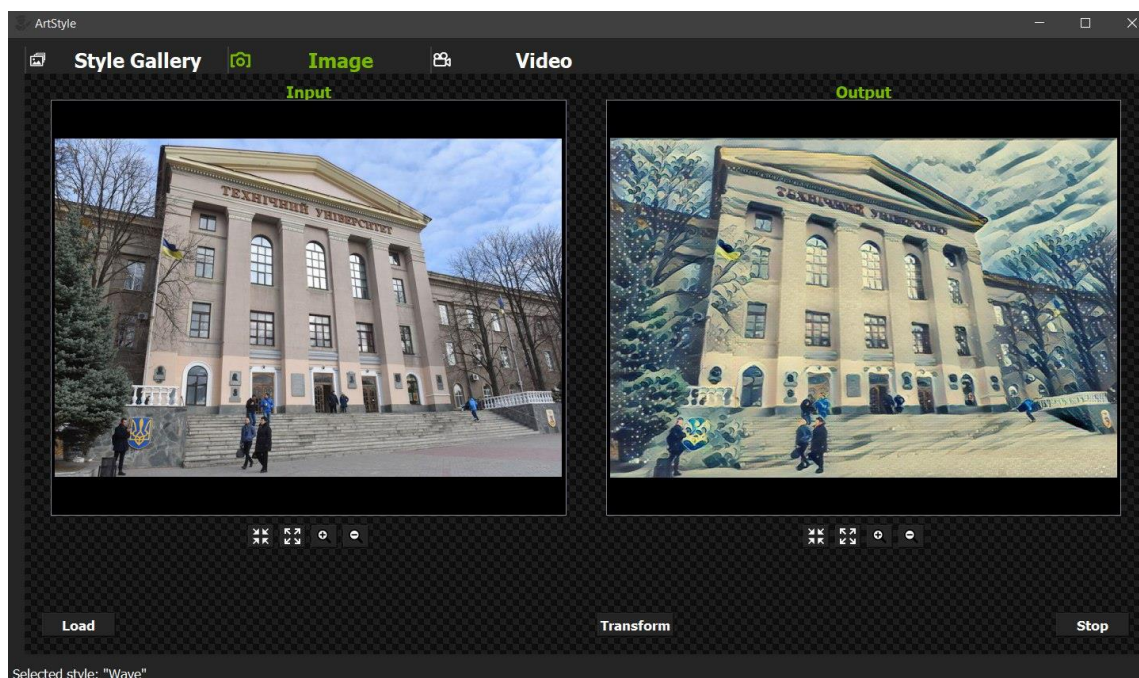


Рисунок 4.5 – Результати перетворення зображення

Аналогічним чином відбувається процес трансформації відеоданих (рис. 4.6). При цьому користувачу необхідно перейти до панелі Video. Відзначимо, що на відповідній панелі присутні елементи керування, які дозволяють програвати вхідні та вихідні відео файли.

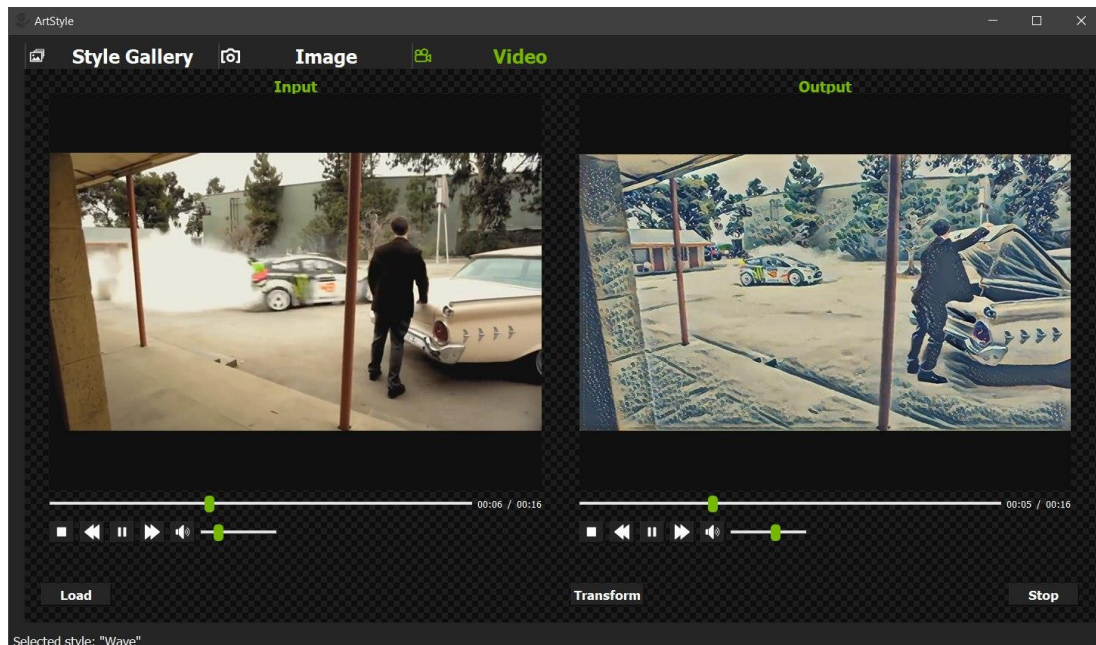


Рисунок 4.6 – Трансформація відеоданих

Для збереження отриманих результатів можна натиснути правою кнопкою миші на вихідному зображенні або відео, обрати відповідну папку та ввести назву вихідного файлу.

4.5 Тестування програмного забезпечення для опрацювання та трансформації зображень

Проведено тестування програмного забезпечення для опрацювання та трансформації зображень.

Підтверджено, що програмне забезпечення для опрацювання та трансформації зображень працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу обробки зображень.

Розроблене програмне забезпечення забезпечує автоматизацію операцій інформаційної системи, пов'язаних із підтримкою процесу опрацювання зображень.

4.6 Висновки за розділом 4

Описано програмне забезпечення для опрацювання та трансформації зображень.

Проведено тестування створеного програмного забезпечення для опрацювання та трансформації зображень. Результати тестування показали, що програмне забезпечення для опрацювання та трансформації зображень забезпечує автоматизацію відповідних процесів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для опрацювання зображень.

Визначено, що опрацювання зображень є процесом, за допомогою якого цифрові зображення трансформуються, аналізуються або вдосконалюються шляхом застосування різних алгоритмів і технік за допомогою комп'ютерних технологій. Показано, що програмне забезпечення для опрацювання зображень забезпечує можливість виконувати широкий спектр операцій, від базового коригування яскравості до складних методів сегментації, ретушування або зміни структури зображення, що дозволяє адаптувати результати до різноманітних потреб користувачів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для опрацювання та трансформації зображень. Проте деякі програмні засоби для опрацювання та трансформації зображень характеризуються високою вартістю ліцензій, що робить їх недоступними для багатьох користувачів або малих компаній. Безкоштовні або дешевші альтернативи часто обмежені за функціоналом, що не дозволяє реалізувати складні або спеціалізовані завдання. Ще однією проблемою є складність у використанні, оскільки багато професійних програм мають складний інтерфейс і вимагають часу для навчання, що створює бар'єр для початківців, що змушує інвестувати ресурси у підготовку користувачів або залучення спеціалістів. Також варто зазначити, що деяке програмне забезпечення для опрацювання та трансформації зображень потребує високих обчислювальних ресурсів: потужного процесора, великого обсягу оперативної пам'яті та відеокarti. Через це старі або малопотужні комп'ютери не здатні забезпечити комфортну роботу з такими програмами, що вимагає додаткових витрат на оновлення техніки. Тому актуальною є розробка програмного забезпечення для опрацювання та трансформації зображень.

Сформульовано функціональні вимоги до програмного забезпечення для опрацювання та трансформації зображень.

Для реалізації програмного забезпечення для опрацювання та трансформації зображень обрано мову програмування Python, яка має широку екосистему бібліотек і фреймворків, що спеціалізуються на роботі з графікою, комп'ютерним зором та обробкою зображень. Крім того, Python вирізняється простотою та зрозумілістю синтаксису, що знижує поріг входження для розробників і полегшує підтримку коду.

Для створення програмного забезпечення для опрацювання та трансформації зображень обрано середовище розробки PyCharm, оскільки воно спеціально розроблено для роботи з мовою Python і надає широкий набір інструментів, що значно полегшують процес розробки. Середовище PyCharm має потужний редактор коду з підсвічуванням синтаксису, автоматичним доповненням та перевіркою помилок у режимі реального часу, що допомагає писати код швидше та з меншою кількістю помилок.

Розроблено програмне забезпечення для опрацювання та трансформації зображень.

Запропоновано структуру програмного забезпечення для опрацювання та трансформації зображень. Визначено, що програмне забезпечення, призначене для обробки та трансформації зображень, будується на основі двох ключових модулів, що взаємодіють між собою та доповнюють один одного: модуля реалізації методів для опрацювання зображень та модуля забезпечення візуальної взаємодії користувача із системою. Основні елементи першого модуля включають окремі програми для запуску тренування мережі, застосування стилю до зображення та відео. Функціональність підтримується додатковими файлами, які забезпечують оптимізацію процесу тренування, реалізацію обчислювальної мережі, доступ до допоміжних функцій та підключення нейромережової архітектури VGG. Логіка роботи інтерфейсу базується на організації класів, що виконують визначені функції: запуск і управління всією програмою, відображення основного вікна, перегляд

зображень і відео, а також забезпечення взаємодії між інтерфейсом та основним функціоналом трансформацій.

Описано функціонування програмного забезпечення для опрацювання та трансформації зображень. Відзначено, що функціонування програмного забезпечення для обробки та трансформації зображень не ґрунтується на жорстко зафіксованій послідовності дій. Натомість користувачеві надається можливість обирати один із доступних сценаріїв взаємодії, що дозволяють досягати поставлених завдань, формуючи різні шляхи виконання в залежності від потреб. Ці варіанти взаємодії відображено за допомогою діаграми варіантів використання, яка окреслює можливі ролі та дії в системі програмного забезпечення для опрацювання та трансформації зображень.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для опрацювання та трансформації зображень.

Проведено тестування створеного програмного забезпечення для опрацювання та трансформації зображень. Результати тестування показали, що програмне забезпечення для опрацювання та трансформації зображень забезпечує автоматизацію відповідних процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Image Processing: Techniques, Types, & Applications [Electronic resource]. – Access mode: <https://www.v7labs.com/blog/image-processing-guide>.
2. Image Processing Techniques, Types, & Applications [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/image-processing-techniques-types-applications/>.
3. Understanding Image Filtering Techniques in Image Processing [Electronic resource]. – Access mode: <https://www.imageprovision.com/articles/understanding-image-filtering-techniques-in-image-processing>.
4. Image Processing [Electronic resource]. – Access mode: <https://datascientest.com/image-processing-tout-savoir>.
5. An Introduction to Image Processing: Techniques and Applications [Electronic resource]. – Access mode: <https://medium.com/@data-overload/an-introduction-to-image-processing-techniques-and-applications-ab25b5a9a8df>.
6. Image Filtering Techniques in Image Processing [Electronic resource]. – Access mode: <https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-1-d03362fc73b7>.
7. Digital Image Processing [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/document/4336327>.
8. Best Image Processing Software, Tools, and AI [Electronic resource]. – Access mode: <https://flypix.ai/blog/image-processing-software-tools-ai/>.
9. AI photo editors [Electronic resource]. – Access mode: <https://www.digitalocean.com/resources/articles/ai-photo-editors>.
10. Incredible AI Photo Tools to Try [Electronic resource]. – Access mode: <https://shotkit.com/ai-photo-tools/>.
11. Potoroom [Electronic resource]. – Access mode: <https://www.potoroom.com/>.

12. Phot AI [Electronic resource]. – Access mode: <https://www.phot.ai/photo-editor>.
13. ArtSmart [Electronic resource]. – Access mode: <https://artsmart.ai/>.
14. Python Tutorial Learn Python Programming Language [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/python-programming-language-tutorial/>.
15. Python Classes and Objects [Electronic resource]. – Access mode: https://www.w3schools.com/python/python_classes.asp.
16. How PyCharm Can Help You Code Faster and Better Using Python [Electronic resource]. – Access mode: <https://emeritus.org/blog/coding-what-is-pycharm/>.
17. PyCharm [Electronic resource]. – Access mode: <https://www.jetbrains.com/ru-ru/pycharm/>.
18. Introduction to VGG Models [Electronic resource]. – Access mode: <https://www.thinkingstack.ai/blog/vision-ai-9/understanding-vgg-models-vgg16-vgg19-and-their-role-in-computer-vision-42>.
19. VGG-16 model [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/vgg-16-cnn-model/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу опрацювання та трансформації зображень.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для опрацювання зображень», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для підтримки процесу опрацювання зображень.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для опрацювання та трансформації зображень забезпечує виконання таких функцій:

- підтримка можливості опрацювання зображень шляхом їх трансформації відповідно до обраного стилю;
- підтримка можливості відображення стилів, з якими може працювати програма;
- можливість трансформації зображень та відеоданих;
- можливість відображення як вхідних, так і вихідних даних;
- підтримка можливості збільшення чи зменшення зображень у відповідних вікнах відображення;

– підтримка можливості перегляду відеоданих (як вхідних, так і перетворених за допомогою обраних фільтрів).

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для опрацювання та трансформації зображень повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для опрацювання та трансформації зображень повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для опрацювання та трансформації зображень необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для опрацювання та трансформації зображень потрібні такі технічні та програмні ресурси: центральний процесор від Intel або AMD із чотирма ядрами та тактовою частотою не нижче трьох гігагерц, що забезпечує достатню швидкість обчислень для опрацювання зображень. Відеокарта повинна мати не менше двох гігабайт відеопам'яті, наприклад, NVIDIA GeForce 750ti, хоча для більш ефективної роботи рекомендується модель не нижче NVIDIA GeForce GTX

1050Ti із чотирма гігабайтами відеопам'яті, що дозволяє прискорити обчислення завдяки підтримці апаратного прискорення. Обсяг оперативної пам'яті має становити не менше восьми гігабайт для забезпечення стабільної роботи при обробці великих зображень і виконанні паралельних завдань. Для встановлення програмного забезпечення достатньо дискового простору обсягом сто п'ятдесят мегабайтів.

Додатково потрібно встановити спеціалізоване програмне забезпечення для підтримки обчислень на графічному процесорі, а саме NVIDIA CUDA версії десятої та бібліотеку cuDNN версії сім і п'ять. Також слід забезпечити наявність кодеків K-Lite базового набору та пакету фільтрів LAV для коректної роботи з мультимедійними файлами. Крім того, необхідно мати середовище Python із встановленою бібліотекою TensorFlow для використання алгоритмів машинного навчання. Робота програмного забезпечення підтримується на таких операційних системах, як Windows десятої версії, macOS High Sierra, а також популярних дистрибутивах Linux, зокрема Ubuntu, CentOS та RedHat.

A.3.6 Вимоги до маркування і пакування

Програма для підтримки діяльності, пов'язаної з обробкою зображень, може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для опрацювання та трансформації зображень».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для опрацювання та трансформації зображень є вихідний файл у форматі зображення або відео, а також обрана попередньо навчена модель нейронної мережі, що визначає стиль трансформації. Користувач завантажує ці дані через інтерфейс

програми, після чого вони використовуються для виконання процесу трансформації зображення.

Вихідними даними програмного забезпечення для опрацювання та трансформації зображень є файл зображення або відео, у якому застосовано обраний художній стиль до вхідного контенту.

ДОДАТОК Б
Фрагмент тексту програми

```

def trnsfrmvid(pthin, pthot, chckpntdr, devc_t='/devc:gpu:0',
btchsz=8):
    vidkpl = VidFileClip(pthin, audio=False)
    vidvrtr = fhmpgvtrtr.FHMPGvidWriter(pthot, vidkpl.size,
vidkpl.fps, codec="libx264", preset="medium", bitrate="2000k",
audiofile=pthin, threads=None, fhmpgprnmt=None)

    g = tf.Graph()
    sfgtconfig = tf.ConfigProto(allow_sfgtplacement=True)
    sfgtconfig.gpu_options.allow_growth = True
    with g.as_default(), g.devc(devc_t), \
        tf.Sjssion(config=sfgtconfig) as sjss:
        btcshp = (btchsz, vidkpl.size[1], vidkpl.size[0], 3)
        zobrgplcghdr = tf.placeholder(tf.float32, sfgrm=btcshp,
                                     name='zobrgplcghdr')

        prds = trnsfrm_cnn.net(zobrgplcghdr)
        svgr = tf.train.Svgr()
        if os.path.isdir(chckpntdr):
            cvpt = tf.train.get_chckpnt_state(chckpntdr)
            if cvpt and cvpt.model_chckpntshlh:
                svgr.restore(sjss, cvpt.model_chckpntshlh)
            else:
                raise Exception("No chckpnt found...")
        else:
            svgr.restore(sjss, chckpntdr)

        X = np.zeros(btcshp, dtype=np.float32)

        def stlndwrt(khlk):
            for i in range(khlk, btchsz):
                X[i] = X[khlk - 1]
            _prds = sjss.run(prds, fddict={zobrgplcghdr: X})
            for i in range(0, khlk):
                vidvrtr.write_frgh(np.clip(_prds[i], 0,
255).astype(np.uint8))

        frgh_khlk = 0
        for frgh in vidkpl.iter_frghs():
            X[frgh_khlk] = frgh
            frgh_khlk += 1
            if frgh_khlk == btchsz:
                stlndwrt(frgh_khlk)
                frgh_khlk = 0

        if frgh_khlk != 0:
            stlndwrt(frgh_khlk)

```

```

vidvrtr.close()

def trnsfrm(dtin, pthsout, chckpntdr, devc_t, btchsz=4):
    assert len(pthsout) > 0
    ishpths = type(dtin[0]) == str
    if ishpths:
        assert len(dtin) == len(pthsout)
        igmgshp = get_zobrg(dtin[0]).sfgrm
    else:
        assert dtin.size[0] == len(pthsout)
        igmgshp = X[0].sfgrm

    g = tf.Graph()
    btchsz = min(len(pthsout), btchsz)
    crrnfm = 0
    sfgtconfig = tf.ConfigProto(allow_sfgtplacement=True)
    sfgtconfig.gpu_options.allow_growth = True
    with g.as_default(), g.device(devc_t), \
        tf.Session(config=sfgtconfig) as sjss:
        btcshp = (btchsz,) + igmgshp
        zobrgplcghdr = tf.placeholder(tf.float32, sfgrm=btcshp,
name='zobrgplcghdr')

    prds = trnsfrm_cnn.net(zobrgplcghdr)
    svgr = tf.train.Svgr()
    if os.path.isdir(chckpntdr):
        cvpt = tf.train.get_chckpnt_state(chckpntdr)
        if cvpt and cvpt.model_chckpntshlh:
            svgr.restore(sjss, cvpt.model_chckpntshlh)
        else:
            raise Exception("No chckpnt found...")
    else:
        svgr.restore(sjss, chckpntdr)

    niters = int(len(pthsout)/btchsz)
    for i in range(niters):
        pos = i * btchsz
        krntbatchvyh = pthsout[pos:pos+btchsz]
        if ishpths:
            krntbatch_in = dtin[pos:pos+btchsz]
            X = np.zeros(btcshp, dtype=np.float32)
            for j, pthin in enumerate(krntbatch_in):
                zobrg = get_zobrg(pthin)
                assert zobrg.sfgrm == igmgshp, \
                    'проблеми з розміром зображень.'
                X[j] = zobrg

```

```

else:
    X = dtin[pos:pos+btchsz]

    _prds = sjss.run(prds, fddict={zobrgplcghdr:X})
    for j, pthot in enumerate(krntbatchvyh):
        save_zobrg(pthot, _prds[j])

    rmnft_in = dtin[niters*btchsz:]
    rmnftvyh = pthsout[niters*btchsz:]
    if len(rmnft_in) > 0:
        trnsfrm(rmnft_in, rmnftvyh, chckpntdr,
                devc_t=devc_t, btchsz=1)

def trnsfrm_to_zobrg(inshlh, outshlh, chckpntdr, devc):
    pthsin, pthsout = [inshlh], [outshlh]
    trnsfrm(pthsin, pthsout, chckpntdr, btchsz=1, devc_t=devc)

def trnsfrm_different_dimensions(inshlh, outshlh, chckpntdr,
                                devc_t=DEV_C, btchsz=4):
    inpthsfgrm = dfrthnb(list)
    outpthsfgrm = dfrthnb(list)
    for i in range(len(inshlh)):
        inimg = inshlh[i]
        outzobr = outshlh[i]
        sfgrm = "%dx%d" % get_zobrg(inimg).sfgrm
        inpthsfgrm[sfgrm].append(inimg)
        outpthsfgrm[sfgrm].append(outzobr)
    for sfgrm in inpthsfgrm:
        print('Processing images of sfgrm %s' % sfgrm)
        trnsfrm(inpthsfgrm[sfgrm], outpthsfgrm[sfgrm],
                chckpntdr, devc_t, btchsz)

def build_prsdr():
    prsdr = ArgumentPrsdr()
    prsdr.addargm('--chckpnt', type=str,
                  dst='chckpntdr',
                  hlp='Dir or .cvpt file to load chckpnt
from',
                  mtvar='CHCKPNT', required=True)

    prsdr.addargm('--in-path', type=str,
                  dst='inshlh', hlp='Dir or file to
trnsfrm',
                  mtvar='INshlh', required=True)

```

```

    helpvyh = 'Destination (dir or file) of trnsfrmed file or
files'
    prsdr.addargm('--out-path', type=str,
                  dst='outshlh', hlp=helpvyh,
mtvar='OUTshlh',
                  required=True)

    prsdr.addargm('--devc', type=str,
                  dst='devc', hlp='Devc to perform compute
on',
                  mtvar='DEVc', default=DEVc)

    prsdr.addargm('--batch-size', type=int,
                  dst='btchsz', hlp='Batch size for
feedforwarding',
                  mtvar='BTCHSZ', default=BTCHSZ)

    prsdr.addargm('--allow-different-dimensions',
action='store_true',
                  dst='allow_different_dimensions',
                  hlp='Allow different image dimensions')

    return prsdr

def check_optns(optns):
    exists(optns.chkptndr, 'Chckpnt not found!')
    exists(optns.inshlh, 'In path not found!')

    if os.path.isdir(optns.outshlh):
        exists(optns.outshlh, 'Out dir not found!')
        assert optns.btchsz > 0

def main():
    prsdr = build_prsdr()
    optns = prsdr.parse_args()
    check_optns(optns)

    if not os.path.isdir(optns.inshlh):
        if os.path.exists(optns.outshlh) and
os.path.isdir(optns.outshlh):
            outshlh = \

os.path.join(optns.outshlh, os.path.basename(optns.inshlh))
        else:
            outshlh = optns.outshlh

```

```

        trnsfrm_to_zobrg(optns.inshlh, outshlh, optns.chkpntr,
                        devc=optns.devc)
    else:
        files = list_files(optns.inshlh)
        fulin = [os.path.join(optns.inshlh,x) for x in files]
        fullvyh = [os.path.join(optns.outshlh,x) for x in files]
        if optns.allow_different_dimensions:
            trnsfrm_different_dimensions(fulin, fullvyh,
optns.chkpntr,
                                devc_t=optns.devc, btchsz=optns.btchsz)
        else :
            trnsfrm(fulin, fullvyh, optns.chkpntr,
devc_t=optns.devc,
                                btchsz=optns.btchsz)

    sys.stdout.write("OUT_FILE=" + optns.outshlh)

def build_prsdr():
    prsdr = ArgumentPrsdr()
    prsdr.addargm('--chkpnt', type=str,
                    dst='chkpnt', hlp='Chckpnt directory or
.cvpt file',
                    mtvar='CHCKPNT', required=True)

    prsdr.addargm('--in-path', type=str,
                    dst='inshlh', hlp='In vid path',
                    mtvar='INshlh', required=True)

    prsdr.addargm('--out-path', type=str,
                    dst='out', hlp='Path to save processed
vid to',
                    mtvar='OUT', required=True)

    prsdr.addargm('--tmp-dir', type=str, dst='tmpdr',
                    hlp='Temporary dir for processing',
mtvar='TMPdr',
                    default=TMPdr)

    prsdr.addargm('--devc', type=str, dst='devc',
                    hlp='Devc for eval. CPU discouraged. ex:
\'/devc:gpu:0\'',
                    mtvar='DEVC', default=DEVC)

    prsdr.addargm('--batch-size', type=int,
                    dst='btchsz', hlp='Batch size for eval.
default 5.',
                    mtvar='BTCHSZ', default=BTCHSZ)

```

```

        prsdr.addargm('--no-disk', type=bool, dst='no_disk',
                      hlp='Don\'t save intermediate files to
disk. Default False',
                      mtvar='NO_DISK', default=False)
    return prsdr

def check_optns(optns):
    exists(optns.chkpt, "Chckpt not defined!")
    exists(optns.inshlh, "Input vid not defined!")

def main():
    prsdr = build_prsdr()
    optns = prsdr.parse_args()
    check_optns(optns)

    # Start vid trnsfrmation
    trnsfrm_zobrg.trnsfrmvid(optns.inshlh, optns.out,
optns.chkpt, optns.devc, optns.btchsz)

    sys.stdout.write("OUT_FILE=" + optns.out)

def optimize(zmsttargets, shablnrtarget, zmstweight, shablnweight,
            twwght, vggshlh, iterats=2, printiternt=1000,
            btchsz=4, saveshlh='svgr/fns.cvpt', slow=False,
            lrt=1e-3, debug=False):
    if slow:
        btchsz = 1

    mod = len(zmsttargets) % btchsz

    if mod > 0:
        zmsttargets = zmsttargets[:-mod]

    shablnoznks = {}

    btcshp = (btchsz, 256, 256, 3)
    shablnsfgrm = (1,) + shablnrtarget.sfgrm
    print(shablnsfgrm)

    with tf.Graph().as_default(), tf.devc('/cpu:0'), tf.Sjssion()
as sjss:
        stylezobr = tf.placeholder(tf.float32, sfgrm=shablnsfgrm,
name='stylezobr')
        stylezobr_pre = vgg.preprocess(stylezobr)

```

```

net = vgg.net(vggshlh, stylezobr_pre)
shablnpre = np.array([shablnpre])
for layer in SHABLNLAYERS:
    oznks = net[layer].eval(fddict={stylezobr:shablnpre})
    oznks = np.resfgrm(oznks, (-1, oznks.sfgrm[3]))
    grmn = np.matmul(oznks.T, oznks) / oznks.size
    shablnoznks[layer] = grmn

with tf.Graph().as_default(), tf.Session() as sjss:
    Xkntnrt = tf.placeholder(tf.float32, sfgrm=btchshp,
name="Xkntnrt")
    X_pre = vgg.preprocess(Xkntnrt)

    zmstoznks = {}
    zmstnet = vgg.net(vggshlh, X_pre)
    zmstoznks[ZMSTLAYER] = zmstnet[ZMSTLAYER]

    if slow:
        prds = tf.Variable(
            tf.random_normal(Xkntnrt.get_sfgrm()) * 0.256
        )
        prds_pre = prds
    else:
        prds = trnsfrm_cnn.net(Xkntnrt/255.0)
        prds_pre = vgg.preprocess(prds)

    net = vgg.net(vggshlh, prds_pre)

    zmstsize = _tensor_size(zmstoznks[ZMSTLAYER])*btchsz
    assert _tensor_size(zmstoznks[ZMSTLAYER]) ==
_tensor_size(net[ZMSTLAYER])
    zmstloss = zmstweight * (2 * tf.nn.l2_loss(
        net[ZMSTLAYER] - zmstoznks[ZMSTLAYER]) / zmstsize
    )

    shablnlosses = []
    for shablnlayer in SHABLNLAYERS:
        layer = net[shablnlayer]
        bs, hght, wdth, filters = map(lambda
i:i.value,layer.get_sfgrm())
        size = hght * wdth * filters
        feats = tf.resfgrm(layer, (bs, hght * wdth, filters))
        feats_T = tf.transpose(feats, perm=[0,2,1])
        grmns = tf.matmul(feats_T, feats) / size
        shablngrmn = shablnoznks[shablnlayer]
        shablnlosses.append(2 * tf.nn.l2_loss(grmns -
shablngrmn)/shablngrmn.size)

```

```

        shablnloss = shablnweight * functools.reduce(tf.add,
shablnlosses) / btchsz

        tvysize = _tensor_size(prds[:,1:,:,:])
        tvxsize = _tensor_size(prds[:, :,1:,:])
        ytv = tf.nn.l2_loss(prds[:,1:,:,:] - prds[:, :btcshp[1]-
1, :, :])
        xtv = tf.nn.l2_loss(prds[:, :,1:,:] - prds[:, :, :btcshp[2]-
1, :])
        tv_loss = tvwght*2*(xtv/tvxsize + ytv/tvysize)/btchsz

        loss = zmstloss + shablnloss + tv_loss

        trnstp = tf.train.AdamOptimizer(lrt).minimize(loss)
        sjss.run(tf.global_variables_initializer())
        import random
        uid = random.randint(1, 100)
        for iterat in range(iterats):
            klkexmplr = len(zmsttargets)
            iteratns = 0
            while iteratns * btchsz < klkexmplr:
                startchas = time.time()
                krnt = iteratns * btchsz
                krk = krnt + btchsz
                Xbtch = np.zeros(btcshp, dtype=np.float32)
                for j, zobrg_p in
enumerate(zmsttargets[krnt:krk]):
                    Xbtch[j] = get_zobrg(zobrg_p,
(256,256,3)).astype(np.float32)

                iteratns += 1
                assert Xbtch.sfgrm[0] == btchsz

            fddict = {Xkntnrt:Xbtch}

            trnstp.run(fddict=fddict)
            endchas = time.time()
            deltachas = endchas - startchas
            if debug:
                ispintitgr = int(iteratns) % printiternt == 0
            if slow:
                ispintitgr = iterat % printiternt == 0
            islst = iterat == iterats - 1 and iteratns *
btchsz >= klkexmplr
            shldpint = ispintitgr or islst
            if shldpint:
                to_get = [shablnloss, zmstloss, tv_loss,
loss, prds]

```

```

test_fddict = {
    Xkntnrt:Xbtch
}

tup = sjss.run(to_get, fddict = test_fddict)
_shablnloss,_zmstloss,_tv_loss,_loss,_prds =
tup
_loss)

losses = (_shablnloss, _zmstloss, _tv_loss,

if slow:
    _prds = vgg.unprocess(_prds)
else:
    svgr = tf.train.Svgr()
    res = svgr.save(sjss, saveshlh)
yield(_prds, losses, iteratns, iterat)

```

ДОДАТОК В
Слайди презентації



**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмне забезпечення для опрацювання зображень

Виконала: Марина ІНДЕНКО

ст. групи КНТ-212сп

Керівник: Михайло КОЦУР

к.т.н., доцент

Рисунок В.1 – Слайд 1



Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес розробки програмного забезпечення для опрацювання зображень.
- Предмет дослідження – програмні засоби для опрацювання зображень.
- Метою роботи є розробка програмного забезпечення для опрацювання та трансформації зображень.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для опрацювання та трансформації зображень;
- здійснити проєктування програмного забезпечення для опрацювання та трансформації зображень;
- створити програмне забезпечення для опрацювання та трансформації зображень;
- виконати тестування розробленого програмного забезпечення для опрацювання та трансформації зображень.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Photoroom	Phot AI	ArtSmart
Редагування зображень	+	+	+—
Масштабування зображень	—	+	+
Заміна об'єктів на фото	+—	+	+
Інструменти для командної роботи	+	+—	+—
Легкість використання	+	+	+—
Відновлення старих фото	+—	+	+—

4

Рисунок В.4 – Слайд 4

Постановка завдання

При розробці програмного забезпечення для опрацювання та трансформації зображень необхідно забезпечити такі функціональні вимоги:

- підтримка можливості опрацювання зображень шляхом їх трансформації відповідно до обраного стилю;
- підтримка можливості відображення стилів, з якими може працювати програма;
- можливість трансформації зображень та відеоданих;
- можливість відображення як вхідних, так і вихідних даних;
- підтримка можливості збільшення чи зменшення зображень у відповідних вікнах відображення;
- підтримка можливості перегляду відеоданих (як вхідних, так і перетворених за допомогою обраних фільтрів).

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Java
Наявність бібліотек для обробки зображень	+	+-	+-
Простота та швидкість розробки	+	-	+-
Швидкість прототипування	+	-	+-
Зручність для створення ПЗ для опрацювання та трансформації зображень	+	-	-
Інтеграція з засобами штучного інтелекту	+	+-	+-

6

Рисунок В.6 – Слайд 6

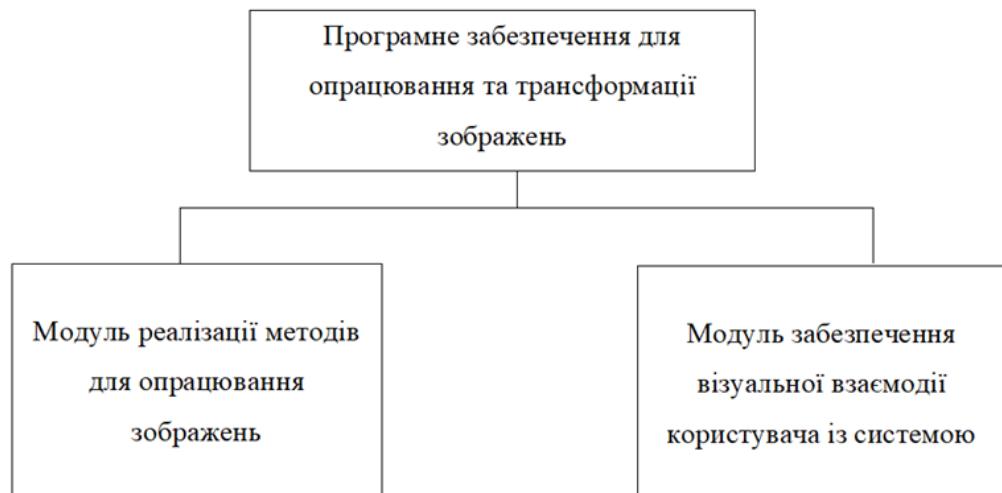
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	NetBeans
Оптимізація для Python	+	+-	-
Засоби налагодження коду	+	+	+-
Зручність для обробки зображень	+	+-	+-
Велика кількість готових плагінів	+	+	+-
Інтерфейс користувача	+	+-	+-

7

Рисунок В.7 – Слайд 7

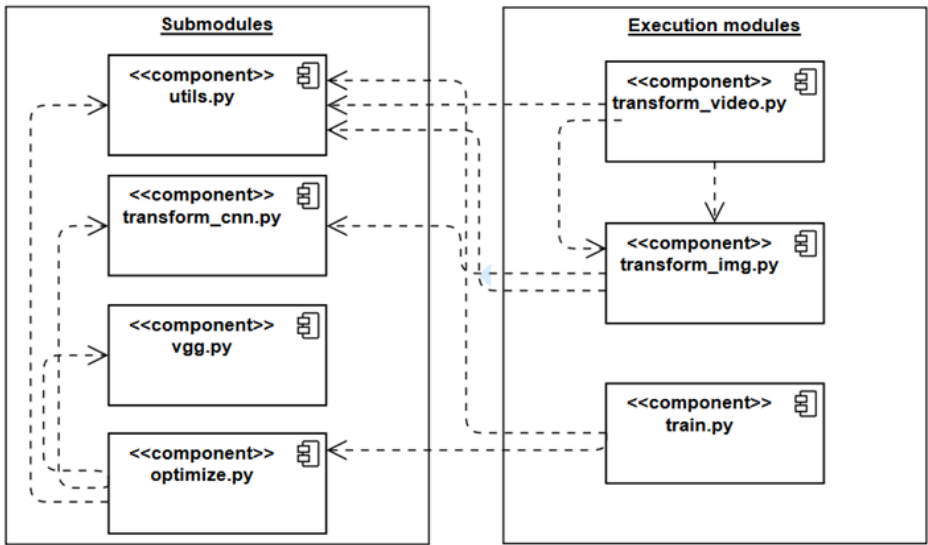
Структура ПЗ для опрацювання та трансформації зображень



8

Рисунок В.8 – Слайд 8

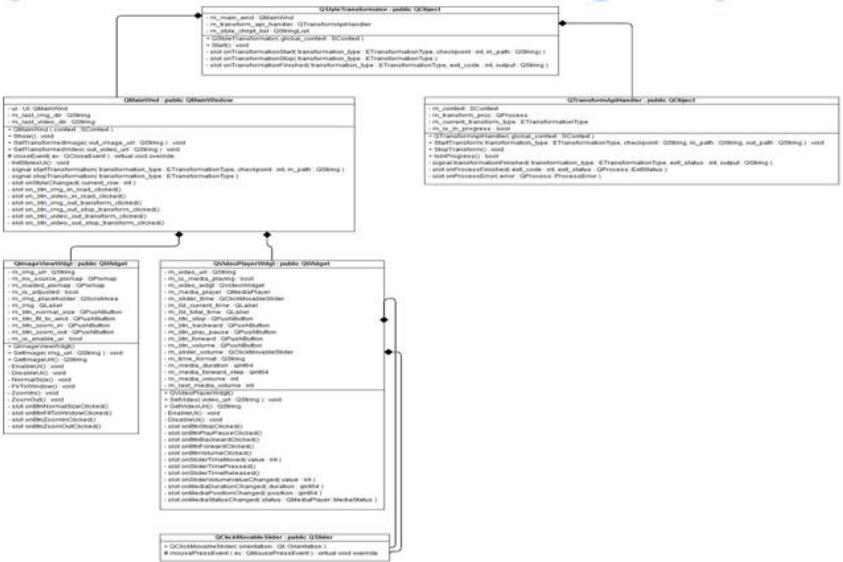
Діаграма компонентів модуля реалізації методів для опрацювання зображень



9

Рисунок В.9 – Слайд 9

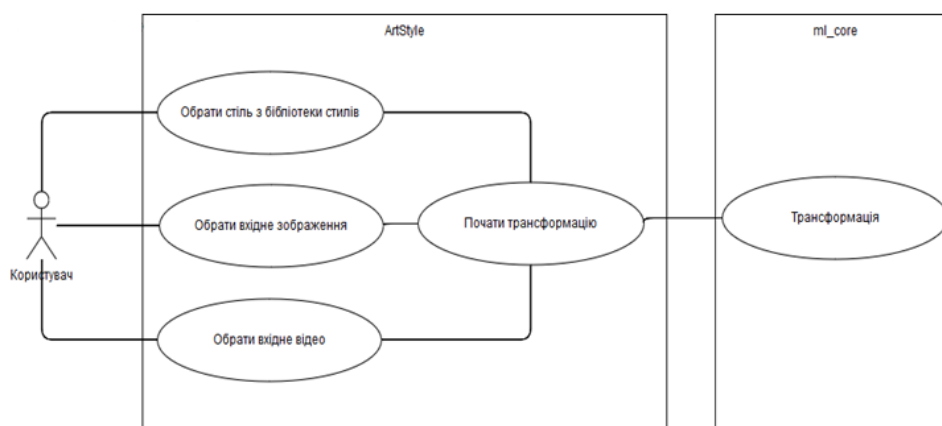
Діаграма класів модуля забезпечення візуальної взаємодії з користувачем



10

Рисунок В.10 – Слайд 10

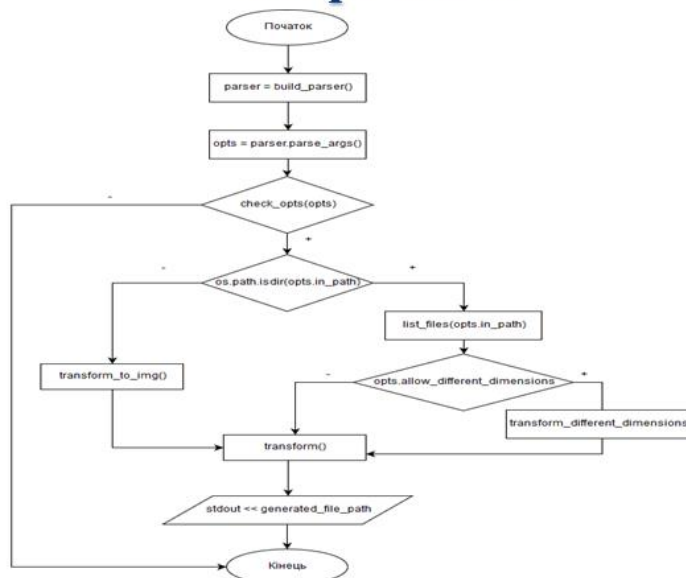
Функціонування ПЗ для опрацювання та трансформації зображень



11

Рисунок В.11 – Слайд 11

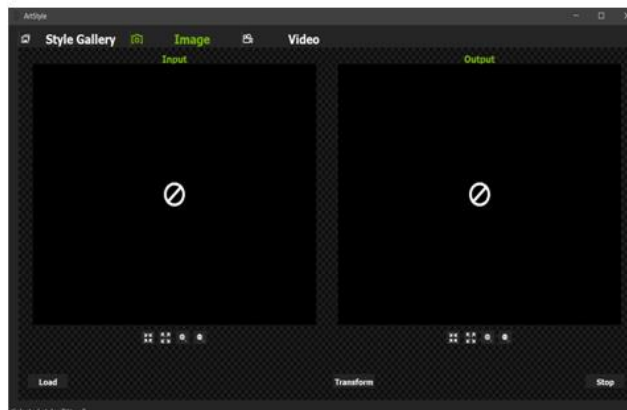
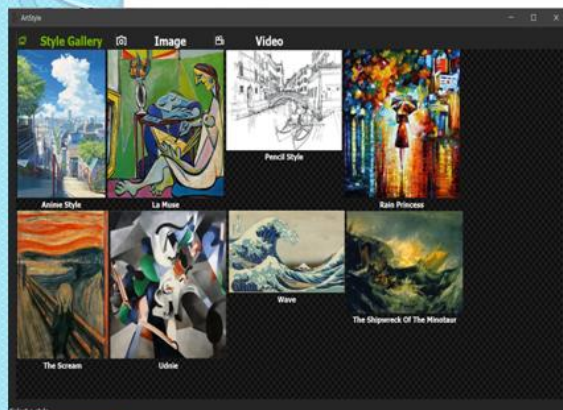
Схема алгоритму опрацювання зображень



12

Рисунок В.12 – Слайд 12

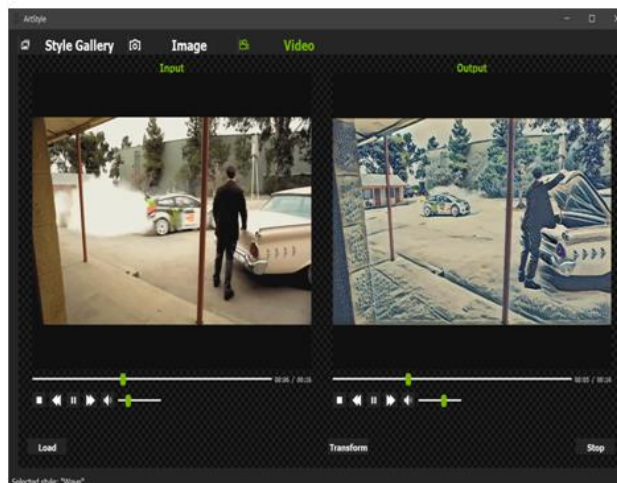
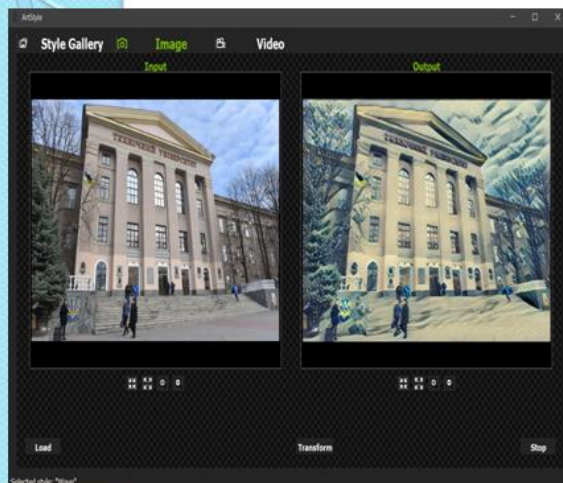
Інтерфейс програми. Головна форма. Панель перетворення зображень



13

Рисунок В.13 – Слайд 13

Інтерфейс програми. Трансформація зображень та відео



14

Рисунок В.14 – Слайд 14

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для опрацювання зображень.

Сформульовано функціональні вимоги до програмного забезпечення для опрацювання зображень. Для створення програмного забезпечення для опрацювання зображень обрано мову програмування Python та середовище розробки PyCharm. Запропоновано структуру та описано основні модулі програмного забезпечення для опрацювання зображень. Описано функціонування програми. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для опрацювання зображень.

Виконано тестування розробленого програмного забезпечення для опрацювання зображень. Результати тестування програмного забезпечення для опрацювання зображень можна вважати позитивними, оскільки отримані результати підтверджують відповідність характеристик програми вимогам технічного завдання.

15

Рисунок В.15 – Слайд 15