Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

за спеціальності 121 – Інженерія програмного забезпечення

На тему: Фоторедактор з функціями штучного інтелекту: розробка та реалізація

Засвідчую, що в цій кваліфікаційній роботі немає
запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-2
_____/А.С Таран/

Керівник
кваліфікаційної роботи _____ /А.В Козиков/

Завідувач кафедри _____ /А.М.Стрюк/

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М.Стрюк

«__» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-20-2 Тарану Антону Сергійовичу

1. Тема: Фоторедактор з функціями штучного інтелекту: розробка та реалізація. Затверджено наказом по КНУ №__ від «__» _____ 2024р.

2. Термін подання студентом закінченої роботи : «__» _____ 2024р.

3. Вихідні дані по роботі: розроблювана система повинна реалізовувати фоторедактор з функціями які працюють за допомогою штучного інтелекту.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): провести аналіз ринку та вже існуючих систем для обробки фото з елементами штучного інтелекту, визначитися з вимогами до фінальної версії розроблюваного програмного забезпечення, спроектувати систему, її функціонал, дизайн, розробити спроектовану систему та протестувати її, перевірити коректну роботу усього функціоналу.

5. Перелік ілюстративного матеріалу: скріншоти екранних форм, блок-схеми функціоналу системи.

Календарний план**:**

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Аналіз літератури та збір даних	23.01.2024 – 06.02.2024
2	Аналіз аналогів програмного забезпечення	06.02.2024 – 10.02.2024
3	Формулювання теми, вихідних даних та необхідних результатів	10.02.2024 – 13.02.2024
4	Формулювання та оформлення першого розділу роботи	14.02.2024 – 23.02.2024
5	Створення логіки програмної частини роботи	24.03.2024 – 05.03.2024
6	Розробка користувацького інтерфейсу	08.03.2024 – 22.03.2024
7	Формулювання та оформлення другого розділу роботи	23.03.2024 – 26.03.2024
8	Реалізація функціоналу програмного забезпечення	26.03.2024 – 04.04.2024
9	Формулювання та оформлення третього розділу роботи	09.04.2024 – 15.04.2024
10	Створення пояснювальної записки	16.04.2024 – 23.05.2024

Дата видачі завдання: «23» січня 2024р.

Студент _____/ А.С.Таран

Керівник роботи _____/А.В.Козиков

РЕФЕРАТ

ФОТОРЕДАКТОР З ФУНКЦІЯМИ ШТУЧНОГО ІНТЕЛЕКТУ: РОЗРОБКА ТА РЕАЛІЗАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОМЕРЕЖІ, ОБРОБКА ФОТО

Пояснювальна записка: 49 с., 1 табл., 13 рис., 1 дод., 10 джерел.

У сучасному світі фотографія має важливу роль повсякденному житті, ставши невід'ємною частиною спілкування, творчості, розваг. З розвитком технологій зйомки та обробки зображень зросла і потреба у вдосконаленні інструментів для редагування фотографій. Одним з напрямків цього вдосконалення є використання штучного інтелекту для створення потужних та ефективних фоторедакторів.

Ця кваліфікаційна робота присвячена розробці та реалізації фоторедактора з функціями штучного інтелекту. Головною метою є створення інструменту, який дозволить користувачам не лише ефективно обробляти фотографії, але й робити це більш швидко з використанням передових технологій.

У рамках даної роботи розглядається ряд функцій, які вбудовуються у фоторедактор за допомогою нейромереж. Серед них такі важливі можливості, як видалення об'єктів з фотографій, покращення якості зображень, видалення об'єкту та фону, а також застосування розумних фільтрів.

Результати роботи можуть мати практичне застосування у різних сферах, включаючи медіа, рекламу, та особисте використання. У роботі розглянуті основні етапи розробки та реалізації програмного забезпечення, а також надано огляд сучасних досягнень у цій області та перспективи подальшого розвитку.

ABSTRACT

PHOTO EDITOR WITH FUNCTIONS OF PIECE INTELLECT: DEVELOPMENT AND REALIZATION, ШТУЧНИЙ INTELEKT, NEUROMEREZHI, PHOTO PROCESSING

Thesis in 49 p., 1 table, 13 figure, 1 appendix, 10 source.

In the modern world, photography plays an important role in everyday life, becoming an integral part of communication, creativity, and entertainment. With the development of image capture and processing technologies, the need for improved photo editing tools has also grown. One of the directions of this improvement is the use of artificial intelligence to create powerful and efficient photo editors.

This qualification work is devoted to the development and implementation of a photo editor with artificial intelligence functions. The main goal is to create a tool that will allow users not only to process photos efficiently, but also to do so more quickly using advanced technologies.

This work discusses a number of functions that are built into a photo editor using neural networks. Among them are such important features as removing objects from photos, improving image quality, removing object and background, and applying smart filters.

The results of the work can have practical applications in various fields, including media, advertising, and personal use. The paper discusses the main stages of software development and implementation, as well as provides an overview of current achievements in this area and prospects for further developmen

ЗМІСТ

ВСТУП

У сучасному цифровому світі фотографія займає особливе місце, ставши не лише засобом фіксації моментів, але й потужним засобом для творчості та самореалізації. Завдяки швидкому розвитку технологій, фотографія пережила значні зміни, що дозволило розширити можливості обробки та покращення зображень.

У зв'язку зі зростанням інтересу до штучного інтелекту та його застосування в різних сферах життя, у тому числі й у фотографії, виникає

необхідність у створенні нових засобів обробки та редагування фотографій. Одним із таких інструментів є фоторедактор, що використовує функції штучного інтелекту для автоматизації та покращення процесу обробки фотографій.

Предметом даної кваліфікаційної роботи є розробка та реалізація фоторедактора з функціями штучного інтелекту. Основними функціями розробленого фоторедактора є видалення об'єктів з фотографій, покращення якості зображень, видалення об'єктів та фону, застосування розумних фільтрів, автоматична ретуш фотографій та збільшення їх роздільної здатності. Ці функції спрямовані на полегшення процесу обробки фотографій та підвищення якості вихідних зображень.

Метою даної роботи є дослідження можливостей використання штучного інтелекту в області фотографії. Результатом дослідження є створення фоторедактора з інтегрованими функціями штучного інтелекту, який може стати цінним інструментом для фотографів, дизайнерів та інших користувачів, що працюють з зображеннями.

У роботі буде проведений аналіз існуючих методів обробки зображень та впровадження їх у фоторедактор. Також буде розглянуто практичні аспекти реалізації такого редактора та його можливості у різних сценаріях використання.

1 СУЧАСНИЙ СТАН І АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Актуальність теми кваліфікаційної роботи

Враховуючи швидкий розвиток сучасних технологій і розширення сфери застосування цифрових зображень сьогодні, вони вимагають прогресу і розробки нових інструментів для використання.

Процес обробки стає більш автоматизованим і якіснішим завдяки застосуванню штучного інтелекту у фоторедакторах. Наприклад, функції ретушування та видалення об'єктів, що виконуються нейронними мережами, підвищення якості зображень, тощо.

Коли потрібно мати чудову якість обробки і мінімальний час для виконання завдань, а також зі збільшенням використання фотографій як в особистих, так і в комерційних або наукових цілях, фоторедактор з інтегрованим штучним інтелектом допомагає досягти мети і необхідності отримання результатів з мінімальними зусиллями.

Постійно зростаюча конкуренція на ринку програмного забезпечення для редагування фотографій зобов'язує його розробників постійно вдосконалювати свої продукти. Сьогодні, коли ШІ інтегрувався і в сферу обробки фотографій, це є однією з головних конкурентних переваг, яка привертає увагу користувачів і, безумовно, гарантує бездоганну обробку зображень.

Тому, 5 основних причин чому тема кваліфікаційної роботи є актуальною наступні:

1) швидкий розвиток штучного інтелекту. Прогрес у штучному інтелекті відкриває нові можливості для автоматизації редагування фотографій;

2) популярність соціальних медіа. Зростаюча кількість фотографій у соціальних мережах створює попит на ефективні інструменти редагування;

3) конкурентна боротьба на ринку фоторедакторів. Розробники шукають способи вдосконалити свої продукти, впроваджуючи нові технології;

4) потреба у високоякісних зображеннях. Бізнес і реклама потребують високоякісні зображення, що створює попит на редагування фотографій;

5) розвиток області комп'ютерного зору. Нові можливості у комп'ютерному зорі дозволяють впроваджувати нові методи та алгоритми у фоторедакторах.

1.2 Мета та завдання кваліфікаційної роботи

Метою даної роботи є дослідження можливості застосування штучного інтелекту в обробці фотографій та розробка програмного забезпечення на основі цих знань. Завданнями кваліфікаційної роботи є:

1) вивчити функціонал та ознайомитися з проблемами обробки зображень;

2) ознайомитися з набором методів та алгоритмів, що забезпечують процеси видалення об’єктів з фото, покращення якості, автоматичної ретуші на зображеннях і т.д;

3) розробити програмне рішення, яке повинно враховувати зручність використання та достатню кількість інструментів для редагування фотографій (в тому числі і базових функцій по типу редагування контрастності, яскравості, прозорості і т.д);

4) тестування та валідація програмного забезпечення. Після розробки необхідно провести серію тестів для перевірки функціональності та надійності програми. Виправлення помилок та вдосконалення інтерфейсу користувача також може бути частиною цього етапу. Оцінка продуктивності та порівняння з іншими програмами. Завдяки порівнянню створеного програмного забезпечення з іншими фоторедакторами на ринку, можна буде визначити сильні та слабкі сторони створеного фоторедактора;

5) завершальним етапом роботи є підготовка кваліфікаційного звіту, в якому будуть описані всі етапи дослідження, розробки та тестування програмного забезпечення, а також його результати та висновки;

1.3 Критичний аналіз літературних джерел за темою

Під час аналізу різноманітних літературних джерел з теми фоторедакторів та нейронних мереж мною виявлено деякі основні проблеми та виклики з якими можна стикнутися. Однією з ключових проблем є обмеження точності та надійності деяких функцій, які використовують нейромережі у фоторедакторах.

Наприклад, функція видалення об'єктів може призвести до непередбачуваних артефактів або спотворень на зображенні.

Іншою проблемою є обмежені можливості адаптації нейромереж до різних типів фотографій та сценаріїв. Модель може працювати добре з пейзажами, але показувати погані результати при обробці портретів, що ускладнює загальний процес редагування.

Додатковою проблемою є обмежені можливості з виправлення помилок, які виникають при використанні нейромереж. Оскільки ці алгоритми базуються на великій кількості даних для навчання, вони можуть показувати неправильні результати при редагуванні зображень, якщо вони виявляються поза областями, на яких були навчені.

Також важливо відзначити проблему з ефективністю та швидкістю обробки фотографій за допомогою нейромереж. Деякі обчислення можуть вимагати значних обчислювальних ресурсів та тривалого часу для обробки навіть одного зображення, що ускладнює їхнє практичне використання.

Отже, попри потенційні переваги використання нейромереж у фоторедакторах, важливо враховувати ці проблеми та вдосконалювати технології для забезпечення кращої якості та ефективності обробки фотографій.

Під час аналізу літературних джерел я також ознайомився з основними бібліотеками які в основному використовуються під час реалізації такого типу програмного забезпечення.

Основні бібліотеки, які можуть бути корисними:

1) OpenCV: Це одна з найпопулярніших бібліотек для обробки зображень та комп'ютерного зору. OpenCV надає широкий набір функцій для зчитування, обробки та збереження зображень, а також для взаємодії з ними, таких як зміна розміру, фільтрація, видалення об'єктів та багато іншого.

2) TensorFlow або PyTorch: Ці бібліотеки штучного інтелекту дозволяють реалізувати нейронні мережі для різних завдань, таких як видалення об'єктів, покращення роздільної здатності фото, розумні фільтри. Вони надають потужні інструменти для навчання та використання нейронних мереж.

3) Tkinter або PyQt: Ці бібліотеки використовуються для створення графічного інтерфейсу користувача (GUI) для десктопних програм на Python. Вони дозволяють створювати вікна, кнопки, меню та інші елементи інтерфейсу, що дозволяє зручно взаємодіяти з програмою.

2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1 Розробка та докладний опис функціональної схеми програми

Функціональна схема програми описує взаємодію та зв'язки між різними компонентами програми. Вона дає уявлення про те, як програма працює на загальному рівні.

1) користувацький інтерфейс (UI):
    1) вікно програми - відображення зображення для редагування та елементів інтерфейсу для вибору функцій;
    2) меню - навігаційні опції для користувача, які дозволяють виконувати різноманітні дії, такі як відкриття/збереження файлів, вибір функцій і т.д;
    3) панель інструментів - кнопки та інші елементи для вибору конкретних функцій редагування.
2) модуль завантаження та збереження зображень:
    1) завантаження зображень з файлів або камери;
    2) збереження зображень після редагування.
3) модуль обробки зображень:
    1) ретушування зображень - видалення дефектів, вирівнювання шкіри;
    2) видалення об'єктів - видалення об'єктів або фону зображення;
    3) адаптивне налаштування параметрів зображення (яркість, контраст, баланс кольорів);
    4) блюр - розмиття частин зображення для зниження шуму або створення ефекту м'якості;
    5) прозорість - зміна прозорості шарів або окремих елементів зображення;
    6) чіткість - підвищення різкості зображення.
4) модуль штучного інтелекту:

1) використання нейронних мереж для реалізації розумних функцій, таких як ретуш, видалення об'єктів;

2) навчання моделей на великих наборах зображень для поліпшення їхньої ефективності та точності;

3) розумні фільтри - застосування фільтрів на основі штучного інтелекту для поліпшення зображення;

4) апскейлінг зображення - збільшення розміру зображення з використанням алгоритмів усунення артефактів.

5) модуль управління функціями:

1) Відстеження вибору користувачем певних функцій;

2) Послідовна обробка зображень з врахуванням вибраних опцій.

6) модуль відображення результату:

1) Відображення обробленого зображення після кожного кроку редагування;

2) Можливість порівняти початкове та оброблене зображення.

7) модуль збереження результатів:

1) Збереження обробленого зображення з врахуванням всіх внесених змін.

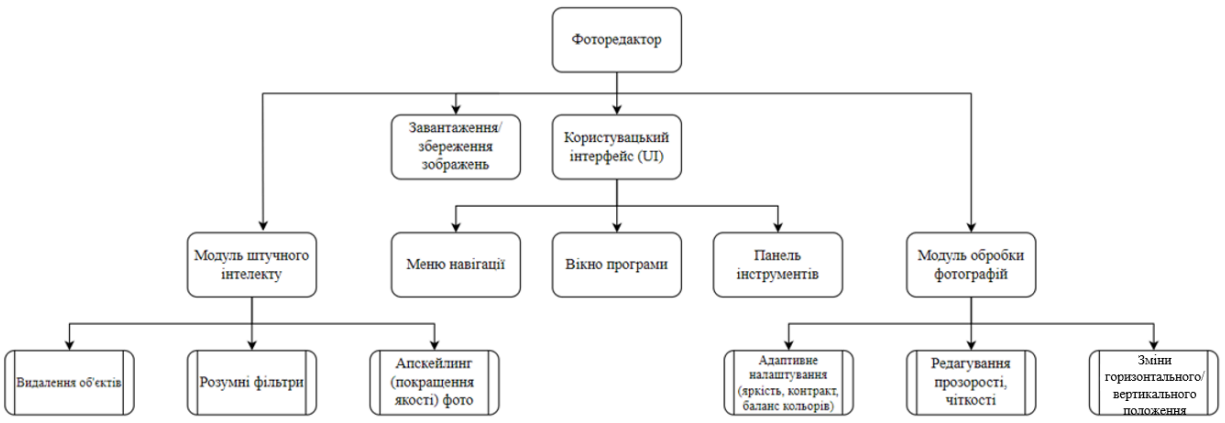

Рисунок 2.1 — Функціональна схема

Для більш детального розуміння роботи фінального програмного забезпечення можна також описати діаграму послідовностей.

Діаграма послідовності - це діаграма уніфікованої мови моделювання (UML), яка ілюструє послідовність повідомлень між об'єктами у взаємодії. Діаграма послідовності складається з групи об'єктів, які представлені лініями життя, і повідомлень, якими вони обмінюються в часі під час взаємодії.

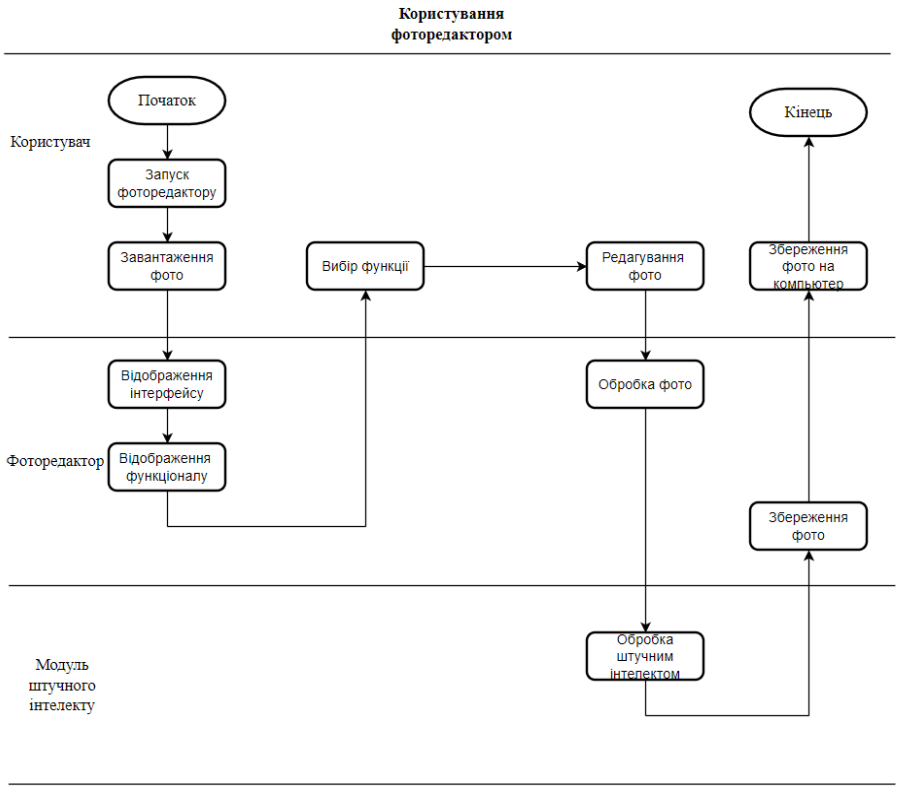

Рисунок 2.2 — Діаграма послідовності

Початок:

Користувач запускає фоторедактор.

Відкривається інтерфейс програми з доступними функціями.

Редагування:

Користувач завантажує фото, яке хоче відредагувати.

З'являється можливість вибрати функцію редагування.

Фоторедактор пропонує різні інструменти для обробки фото:

Обробка фото:

Зміна яскравості, контрастності, насиченості.

Корекція кольорів, балансу білого.

Різкість, розмиття, шуми.

Зміна розміру, кадрування, обертання.

Відображення функціоналу:

Додавання тексту.

Застосування фільтрів, ефектів.

Користувач може використовувати модуль штучного інтелекту для автоматичного покращення фото, видалення об’єктів.

Збереження:

Після редагування фото користувач може зберегти його на комп'ютер.

Фоторедактор може запропонувати додаткові опції:

Збереження в різних форматах.

Зміна розміру та якості зображення.

Додавання водяних знаків.

Кінець:

Після збереження фото процес редагування завершується.

2.2 Розробка та докладний опис алгоритму роботи програми

Для реалізації та проектування алгоритму роботи програми я вибрав мову програмування Python.

Python - це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою.

Обравши Python для реалізації фоторедактора з елементами штучного інтелекту, я врахував декілька факторів.

По-перше, Python має простий та зрозумілий синтаксис, що полегшує розробке. Крім того, велика кількість бібліотек, таких як OpenCV для обробки зображень та SciPy, дозволяє швидко та ефективно втілити різноманітні функції фоторедактора.

Python також має багато інструментів для створення графічного інтерфейсу, що дозволяє створювати зручний та привабливий UI (інтерфейс користувача). Крім того, Python є кросплатформеною мовою програмування, що дозволяє розробляти програми, які працюють на різних операційних системах без змін. Таким чином, обираючи Python, я отримав можливість швидко та ефективно реалізувати задуманий мною фунціонал.

Для реалізації фінального програмного забезпечення я використовую такі бібліотеки як:

Таблиця 2.1 – Використані бібліотеки під час розробки

Бібліотека	Версія	Опис
einops	0.4.1	Бібліотека для маніпуляції тензорами в стилі PyTorch з підтримкою зручного синтаксису.
ffmpeg	1.4	Колекція утиліт та бібліотек для роботи з відео та аудіо файлами у різних форматах.
ffmpeg_python	0.2.0	Python-обгортка для бібліотеки ffmpeg, що дозволяє виконувати різноманітні операції з відео та аудіо.
GPUtil	1.4.0	Утиліта для моніторингу та керування використанням GPU.

Продовження таблиці 2.1

Бібліотека	Версія	Опис
moviepy	1.0.3	Бібліотека для редагування медіа у Python, що надає простий інтерфейс для обробки відеофайлів.
numpy	1.22.4	Основна бібліотека для обчислень у Python, особливо в області наукових обчислень.

opencv_python	4.6.0.66	Бібліотека для комп'ютерного зору та обробки зображень.
panda3d	1.10.12	3D ігровий движок та фреймворк для розробки ігор, візуалізації та симуляцій.
pilgram	1.2.1	Бібліотека для обробки та фільтрації зображень в стилі Instagram.
Pillow	9.3.0	Fork бібліотеки PIL (Python Imaging Library) для обробки та маніпуляції зображеннями.
PyMatting	1.1.8	Бібліотека для математичного вирішення задач з обробки зображень, зокрема, матування.
PyQt6	6.4.0	Прив'язка Python до бібліотеки Qt, що дозволяє створювати графічні інтерфейси користувача.
pyqtgraph	0.13.1	Бібліотека для створення візуалізацій та графіків у Python.
qimage2ndarray	1.9.0	Утиліта для конвертації зображень з формату QImage у масиви NumPy.
rawpy	0.17.3	Бібліотека для читання та обробки RAW-зображень з камер.
requests	2.28.1	Бібліотека для взаємодії з HTTP-запитами.
scikit_image	0.19.3	Бібліотека для обробки та аналізу зображень на базі SciPy.
scipy	1.9.3	Бібліотека для наукових обчислень у Python.

Продовження таблиці 2.1

Бібліотека	Версія	Опис
timm	0.4.12	Бібліотека з моделями глибокого навчання для PyTorch.
torch	1.11.0	Бібліотека для машинного навчання, основана на тензорах, з широким спектром функцій та алгоритмів.

torchvision	0.12.0	Набір утиліт та архітектур для роботи з комп'ютерним зором, побудований на основі PyTorch.
tqdm	4.64.0	Бібліотека для створення прогрес-барів у консольних програмах Python.

Я обрав ці бібліотеки, тому що вони прості у використанні - мають зрозумілу документацію та приклади коду, що робить їх доступним. Також вони забезпечують високу продуктивність та швидкість роботи, дозволяють легко створювати складні та функціональні інтерфейси та алгоритми.

Для прикладу можна розібрати алгоритм роботи апскейлінгу (збільшення якості фото) з використанням цих бібліотек:

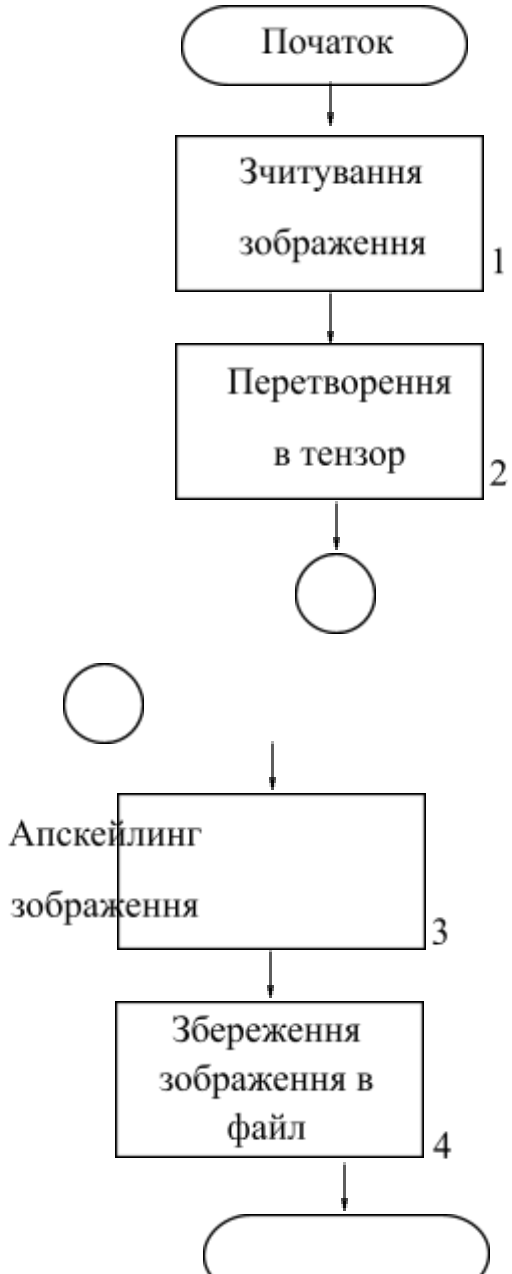

Рисунок 2.3 — Алгоритм роботи апскейлингу

```
import cv2
import numpy as np
import torch
import os
import quality_scaler_utilities as qsu

def convert_image_to_uint(img, n_channels=3):
    return img

def save_uint_image(img, img_path):
    img = np.squeeze(img)
    if img.ndim == 3:
        img = img[:, :, [2, 1, 0]]
    cv2.imwrite(img_path, img)

def convert_uint_to_tensor4(img):
    if img.ndim == 2:
        img = np.expand_dims(img, axis=2)
                                                          return
torch.from_numpy(np.ascontiguousarray(img)).permute(2,        0,
1).float().div(255.).unsqueeze(0)

def convert_tensor_to_uint(img):
    img = img.data.squeeze().float().clamp_(0, 1).cpu().numpy()
    if img.ndim == 3:
      img = np.transpose(img, (1, 2, 0))
    return np.uint8((img * 255.0).round())

def prepare_image_for_deeplearning(img, device):
    if 'cpu' in device:
        backend = torch.device('cpu')
    elif 'cuda' in device:
```

```
        backend = torch.device('cuda')
    elif 'dml' in device:
        backend = torch.device('dml')

    img = convert_image_to_uint(img, n_channels=3)
    img = convert_uint_to_tensor4(img)
    img = img.to(backend, non_blocking=True)
    return img

def upscale_and_save_image(img, model, result_path, device,
tiles_resolution):
    multiplier_num_tiles = 3

    img_tmp = cv2.imread(img)
    image_resolution = max(img_tmp.shape[1], img_tmp.shape[0])
    num_tiles = image_resolution / tiles_resolution

    if num_tiles <= 1:
            img_adapted = prepare_image_for_deeplearning(img,
device)
        with torch.no_grad():
            img_upscaled_tensor = model(img_adapted)
                                                img_upscaled =
convert_tensor_to_uint(img_upscaled_tensor)
        save_uint_image(img_upscaled, result_path)
        return

    num_tiles = round(num_tiles)
    if (num_tiles % 2) != 0:
        num_tiles += 1
    num_tiles = round(num_tiles * multiplier_num_tiles)

    num_tiles_applied = int(num_tiles / 2)
    how_many_tiles = int(pow(num_tiles / 2, 2))

    qsu.split_image(img, num_tiles_applied, num_tiles_applied)

    basename, ext = os.path.splitext(img)

    tiles = []
    for index in range(how_many_tiles):
        tiles.append(basename + "_" + str(index) + ext)

   with torch.no_grad():
        for tile in tiles:
             tile_adapted = prepare_image_for_deeplearning(tile,
device)
                                                tile_upscaled =
convert_tensor_to_uint(model(tile_adapted))
            save_uint_image(tile_upscaled, tile)
```

```
                qsu.reverse_split(tiles,     num_tiles_applied,
num_tiles_applied, result_path, True, False)
```

Рисунок 2.4 — Програмна реалізація апскейлеру зображень

Основні кроки включають роботи цього алгоритму включають:

1) конвертація зображень у формат, придатний для подальшої обробки нейронною мережею;

2) апскейлінг зображень за допомогою заданої моделі нейронної мережі;

3) збереження зображення.

Функція prepare_image_for_deeplearning підготовлює зображення до подальшого використання у нейронній мережі шляхом конвертації його в тензор та переміщення на відповідний пристрій (CPU або GPU).

Функція upscale_and_save_image виконує апскейлінг зображення та зберігає результат. Якщо зображення занадто маленьке для розділення на плитки, воно апскейлюється без розділення. В іншому випадку зображення розділяється на плитки, кожна з яких збільшується окремо, а потім плитки зливаються разом у зображення вищої роздільної здатності.

Основний файл main.py має наступну структуру:

```
class Gui(QtWidgets.QMainWindow):

    sliderChangeSignal = QtCore.pyqtSignal()

    def __init__(self, parent=None):
        super(Gui, self).__init__(parent)
        self.setWindowTitle('PhotoLab')
        self.setMinimumHeight(850)

        self.statusBar = QStatusBar()
        self.setStatusBar(self.statusBar)
```

```
                                                self.OpenShortcut     =
QtGui.QShortcut(QKeySequence("Ctrl+O"), self)
                self.OpenShortcut.activated.connect(self.OnOpen)


                                                 self.PasteShortcut   =
QtGui.QShortcut(QKeySequence("Ctrl+V"), self)

self.PasteShortcut.activated.connect(self.OnPaste)


                                                  self.SaveShortcut   =
QtGui.QShortcut(QKeySequence("Ctrl+S"), self)

self.SaveShortcut.activated.connect(self.OnSaveAs)
```

Рисунок 2.5 — Основна структура головного файлу main.py

Це класс GUI в якому реалізований весь інтерфейс фоторедактора.

2.3 Розробка інтерфейсу програми

Одним із ключових аспектів розробки фоторедактора з елементами штучного інтелекту є створення ефективного та зручного інтерфейсу програми. Інтерфейс визначає спосіб взаємодії користувача з програмою, його зрозумілість та ефективність напряму впливають на користувацький досвід та результативність роботи програми в цілому.

Ретельне проектування інтерфейсу дозволить забезпечити не лише зручність в користуванні програмою, а й максимально ефективне використання функціоналу фоторедактора та його інтелектуальних можливостей. Розгляд підходів до розробки інтерфейсу допоможе виявити оптимальні рішення для забезпечення високої продуктивності та задоволення потреб користувачів.

Базова схема головної сторінки фоторедактора наступна:

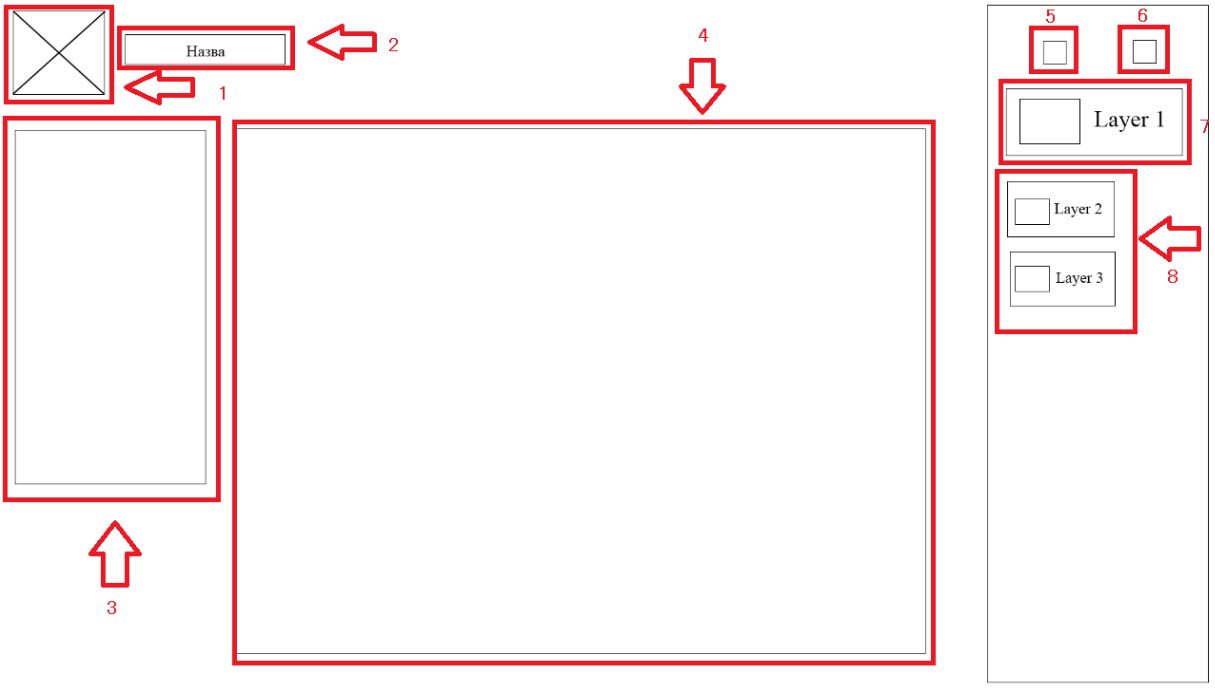

Рисунок 2.6 — Схема інтерфейсу головної сторінки фоторедактора

а) логотип:

Розташований у верхньому лівому кутку.

Відображає логотип програми.

б) назва:

Розташована під логотипом.

Відображає назву програми.

в) панель інструментів:

Розташована зліва від вікна перегляду зображення.

Містить кнопки та інструменти для редагування зображення.

г) відображення зображення:

Розташоване в центрі екрана.

Відображає зображення, яке редагується.

д) додати новий шар:

Кнопка, розташована в нижній частині панелі шарів.

Додає новий порожній шар до зображення.

е) видалити шар:

Кнопка, розташована в нижній частині панелі шарів.

Видаляє вибраний шар.

є) вибраний шар:

Шар, який активний для редагування.

Відображається синім кольором на панелі шарів.

ж) інші шари:

Шари, які не активні .

Відображаються сірим кольором на панелі шарів.

Вибір вибору розумних фільтрів буде виглядати наступним чином:

Рисунок 2.7 — Макет вікна з вибором фільтрів

Також для кожного зображення можна детально налаштовувати інші параметри.

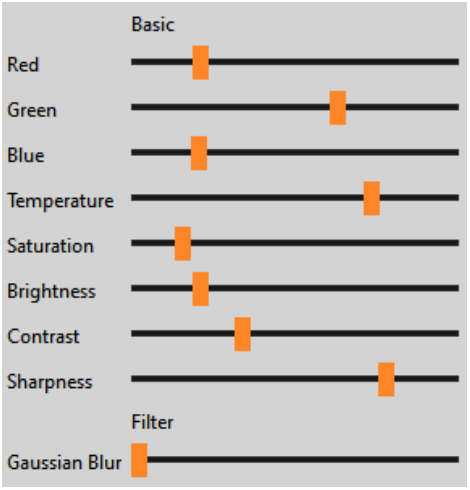

Рисунок 2.8 — Макет елементу налаштування параметрів зображення

Повзунок яскравості використовується для регулювання загальної яскравості зображення. Збільшення значення повзунка зробить зображення яскравішим, а зменшення значення – темнішим.

Повзунок контрастності використовується для регулювання контрастності зображення. Збільшення значення повзунка зробить зображення контрастнішим, а зменшення значення – менш контрастним.

Повзунок температури використовується для регулювання колірної температури зображення. Збільшення значення повзунка зробить зображення теплішим, а зменшення значення – холоднішим.

Повзунок насиченості використовується для регулювання насиченості кольорів зображення. Збільшення значення повзунка зробить кольори зображення більш насиченими, а зменшення значення – менш насиченими.

Повзунок різкості використовується для регулювання різкості зображення. Збільшення значення повзунка зробить зображення більш різким, а зменшення значення – менш різким.

3 РОЗРОБКА БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз обраної середовища програмування

Обравши Python та PyCharm для розробки редактора зображень з можливістю розширення, я спирався на кілька ключових факторів. Python відзначається простотою використання та читабельним синтаксисом, що полегшує швидку реалізацію ідей та спілкування з командою. Крім того, Python має широку екосистему бібліотек для обробки зображень, що дозволяє ефективно використовувати ресурси для створення редактора. Підтримка об'єктно-орієнтованого програмування робить Python ідеальним для створення модульних та розширюваних програм, що відповідають потребам проекту. Нарешті, PyCharm як потужне інтегроване середовище розробки надає всі необхідні інструменти для зручної та продуктивної роботи з кодом, забезпечуючи підтримку автодоповнення, рефакторингу та налагодження. Таким чином, обрана комбінація Python та PyCharm допомагає забезпечити зручність, ефективність та якість у процесі розробки редактора зображень.

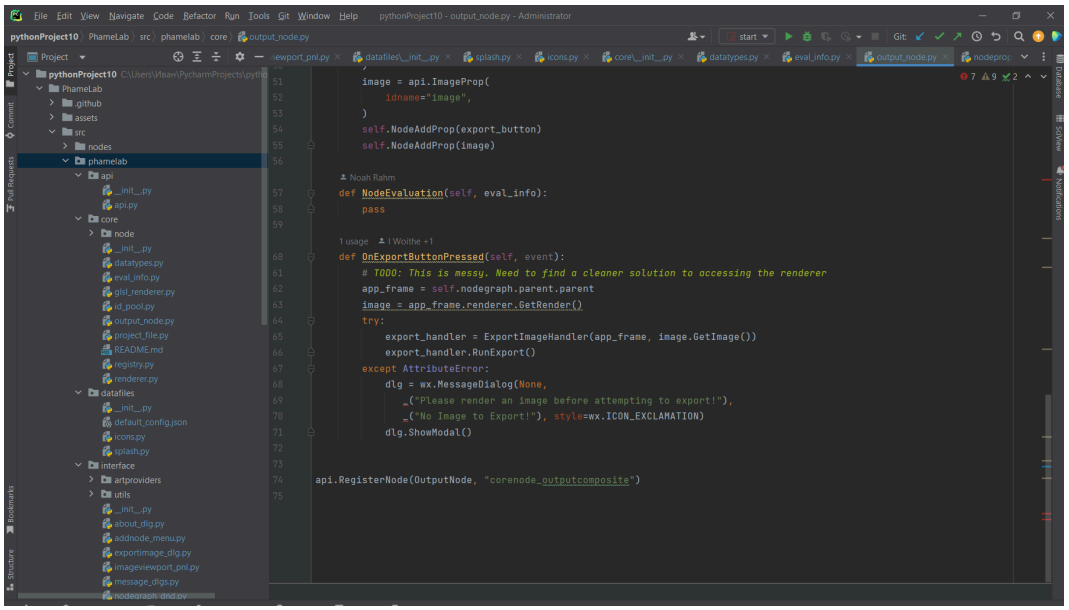

Рисунок 3.1 — Інтерфейс Pycharm

На рисунку 3.1 зверху ми можемо побачити інтегроване середовище розробки для мови програмування Python, яке надає розширені можливості для зручної та продуктивної роботи з кодом.

Призначення PyCharm включає в себе підтримку різноманітних інструментів для редагування, налагодження та аналізу коду, автодоповнення коду, інтеграцію з системами контролю версій, підтримку віртуальних середовищ та пакетних менеджерів, вбудовані інструменти для тестування та профілювання коду, а також інші корисні функції, що допомагають забезпечити ефективну та комфортну розробку програмного забезпечення на Python.

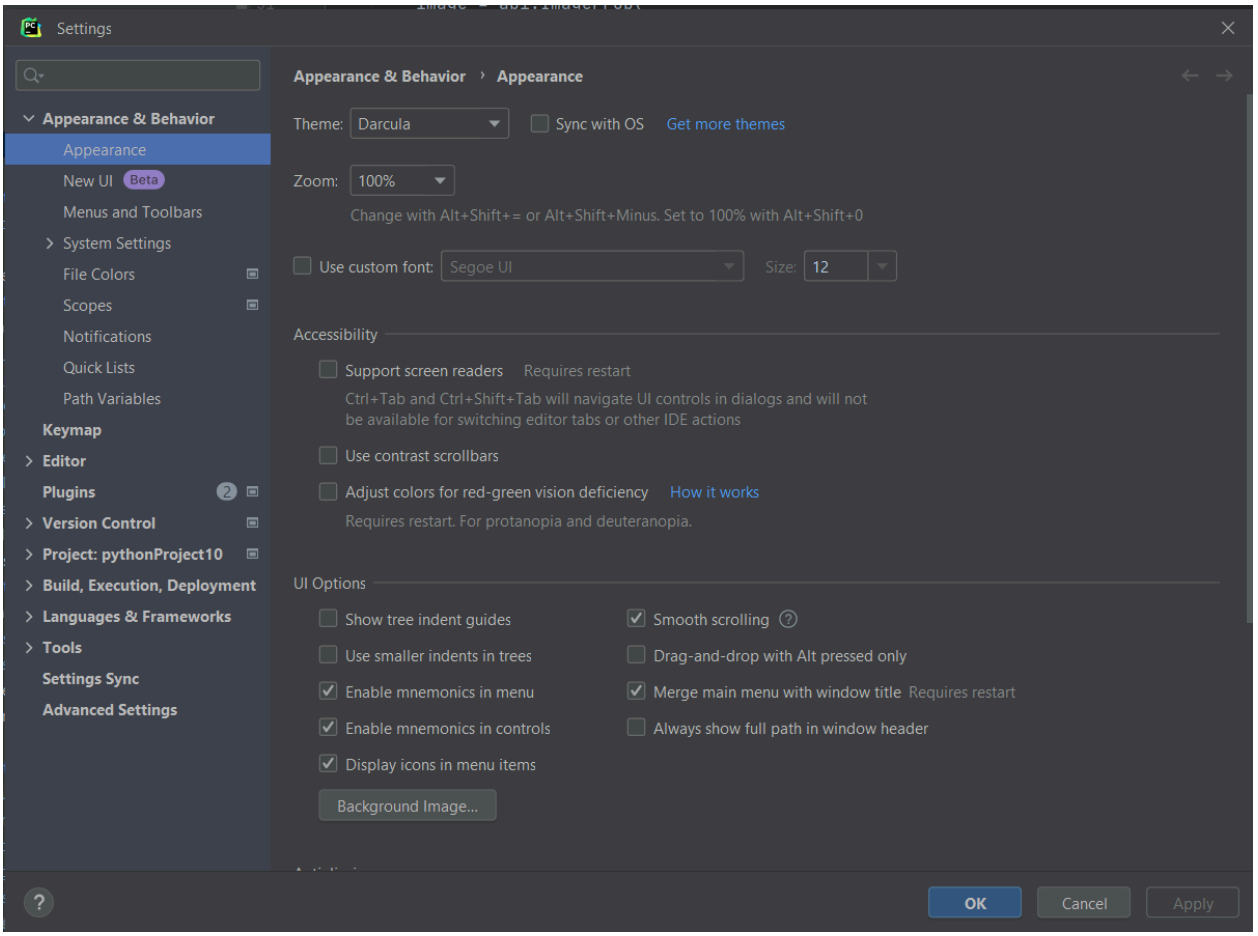

Рисунок 3.2 — Pycharm налаштування

Так-же ми маємо доступ до різних налаштувань, що дозволяє користувачам налаштовувати різні аспекти середовища розробки. Деякі з доступних налаштувань включають можливість змінювати теми та кольорову схему інтерфейсу, налаштовувати автодоповнення коду, налаштовувати інтеграцію з системами контролю версій, змінювати клавіатурні скорочення, налаштовувати зовнішні зовнішні інструменти, такі як лінтери та інші інструменти аналізу коду, налаштовувати відступи та інші правила форматування коду, а також вибирати та налаштовувати різні плагіни та додатки. Разом з тим, користувачі можуть налаштовувати робочі області та проектні налаштування для кожного конкретного проекту.

3.2 Програмна реалізація основних функцій

Костяком любої програмної реалізації є функція main для запуску основних функцій редактора зображень, але важливо врахувати різноманітні аспекти обробки, як завантаження, видалення, знизу надано приклад кусків коду реалізації цих функцій для подальшого використання їх в інших частинах програми

```
class Gui(QtWidgets.QMainWindow):
   sliderChangeSignal = QtCore.pyqtSignal()
   def __init__(self, parent=None):
       super(Gui, self).__init__(parent)
       self.setWindowTitle('PhotoLab')
       self.setMinimumHeight(850)

       self.statusBar = QStatusBar()
       self.setStatusBar(self.statusBar)

       r_histogram = []
       g_histogram = []
       b_histogram = []

       luma_histogram = []
       self.ImageHistogramPlot = pg.plot()
       x = list(range(len(r_histogram)))
        self.ImageHistogramGraphRed = pg.PlotCurveItem(x = x, y =
r_histogram, fillLevel=2, width = 1.0, brush=(255,0,0,80))
        self.ImageHistogramGraphGreen = pg.PlotCurveItem(x = x, y
= g_histogram, fillLevel=2, width = 1.0, brush=(0,255,0,80))
      self.ImageHistogramGraphBlue = pg.PlotCurveItem(x = x, y =
b_histogram, fillLevel=2, width = 1.0, brush=(0,0,255,80))
        self.ImageHistogramGraphLuma = pg.PlotCurveItem(x = x, y
=     luma_histogram,     fillLevel=2,     width     =     1.0,
brush=(255,255,255,80))

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphRed)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphGreen)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphBlue)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphLuma)
       self.HistogramContent = None
       self.ImageHistogramPlot.hide()
       self.color_picker = None
       self.RedFactor = 100
       self.GreenFactor = 100
       self.BlueFactor = 100
       self.Temperature = 6000
```

```
        self.Color = 100
        self.Brightness = 100
        self.Contrast = 100
        self.Sharpness = 100
        self.GaussianBlurRadius = 0
        self.timer_id = -1
        self.sliderExplanationOfChange = None
        self.sliderTypeOfChange = None
        self.sliderValueOfChange = None
        self.sliderObjectOfChange = None

                                            self.OpenShortcut      =
QtGui.QShortcut(QKeySequence("Ctrl+O"), self)
        self.OpenShortcut.activated.connect(self.OnOpen)

                                           self.PasteShortcut      =
QtGui.QShortcut(QKeySequence("Ctrl+V"), self)
        self.PasteShortcut.activated.connect(self.OnPaste)

                                            self.SaveShortcut      =
QtGui.QShortcut(QKeySequence("Ctrl+S"), self)
        self.SaveShortcut.activated.connect(self.OnSaveAs)

                                         self.SaveAsShortcut       =
QtGui.QShortcut(QKeySequence("Ctrl+Shift+S"), self)
        self.SaveAsShortcut.activated.connect(self.OnSaveAs)

                                            self.UndoShortcut      =
QtGui.QShortcut(QKeySequence("Ctrl+Z"), self)
        self.UndoShortcut.activated.connect(self.OnUndo)
```

Код вище містить основний класс MainApp що є кістяком нашої програми та точкою запуску, тобто вся програма починається з цього класу, та далі ми визиваємо інші методи взаємодії з нашим підходом. Сам MainApp є підкласом QMainWindow що являється частиною бібліотеки PyQT6 для створення графічних інтерфейсів.

Так-же ми можемо побачити метод SetLang він служить для визначення локалізації нашого користувача, по замовчуванню стоїть Engl, але якщо користувач змінить на іншу мову, то буде автоматична зміна мови в залежності від мови користувача та його пристрою. Можливо помітити що в нас є перевірки на платформу користувача, якщо це Лінукс, то ми по замовчування будемо брати ту мову, що стоїть в користувача в системі.

```
if __name__ == '__main__':
   os.environ['QT_IMAGEIO_MAXALLOC'] = "1024"
   main()
```

Тут ми робимо перевірки на те що запуск нашої програми буде завжди зроблено з main.py, це потрібно для того щоб убезпечити запуск програми з інших модулів де він не повинен використовуватись, це так-же допомагає позбутись непотрібних використань модулів та методів нашого основного функціонала програми.

Таким чином невеликий код в 100-200 строк є нашим кістяком для запуска програми та використання всього іншого функціонали з максимальною безпечністю та запобіганням непотрібного функціонала за допомогою грамотного розподілення та архітектури проекту

Далі ми переходимо к конфігу та збереженню інформації нашого проекта. Класс нижче відповідальний за таку інформацію як:

1) директорія де зберігається програма;
2) назва програми;
3) опис програми;
4) версія програми;
5) версіонний тег програми;
6) повна версія програми;
7) конфіг файл з заздалегідь зазначеними даними.

```
class AppData(object):
   def __init__(self):
       self.app_frozen = appconst.APP_FROZEN
       self.app_dir = appconst.APP_DIR
       self.app_name = appconst.APP_NAME
       self.app_website_url = appconst.APP_WEBSITE_URL
       self.app_description = appconst.APP_DESCRIPTION
       self.app_version = appconst.APP_VERSION
       self.app_version_tag = appconst.APP_VERSION_TAG
       self.app_version_full = appconst.APP_VERSION_FULL
       self.app_config_file = appconst.APP_CONFIG_FILE
class AppConfiguration(AppData):
```

```
   def __init__(self, app):
       AppData.__init__(self)
       self.app = app
       self.prefs = {}
     def Config(self, key: str = None, keys: tuple = None,
value=None, default=None):
       if key is not None:
           keys = (key)
       if value is not None:
           d_key = None
           for k in keys:
               if keys.index(k) == 0:
                   d_key = self.prefs[k]
               elif keys.index(k) == len(keys) - 1:
                   d_key[k] = value
               else:
                   d_key = d_key[k]
       else:
           try:
               d_key = None
               for k in keys:
                   if keys.index(k) == 0:
                       d_key = self.prefs[k]
                   elif keys.index(k) == len(keys) - 1:
                       return d_key[k]
                   else:
                       d_key = d_key[k]
           except KeyError:
               return default
   def Load(self):
       try:
         os.makedirs(os.path.expanduser("~/.phamelab/"),
                     exist_ok=True)
           if not os.path.exists(self.app_config_file):
                                                          with
open("phamelab/datafiles/default_config.json") as f:
                   default_config = f.read()
               with open(self.app_config_file, "w+") as f:
                   f.write(default_config)
           with open(self.app_config_file, "r") as file:
               self.prefs = json.load(file)
       except IOError:
           pass
```

Код вище містить два основних класи програми, це AppData та AppConfiguration. Клас AppData відповідальний за атрибути даних, та відображення різних параметрів програми, такі як версія програми, діректорія,назва програми, тощо.

Клас AppConfiguration є підкласом AppData, та одна з його задач теж сама що й у AppConfiguration, але він виконує завдання управління конфігурацією програми. Він надає методи для завантаження конфігурації з файлу, збереженню конфігурації у файл, а також отримання та встановлення значень конфігурації з файлу за ключем або ключами. Конфігурація зберігається у вигляді словника. Щодо конфігурації, то її параметри надходять до методу Config, який призначений для доступу до значень конфігурації за ключами що були надані, якщо ключ не знадено в словнику, то ми повертаємо значення за замовчуванням.

3.3 Робота з моделями штучного інтелекту

Основним завданням при розробці цієї кваліфікаційної роботи було зробити можливість використання штучного інтелекту, точніше його доповнення за допомогою внутрішніх можливостей, а саме кастомні ноди. В нас не стояла ціль робити штучний інтелект з нуля, основна ціль це зробити фоторедактор з можливостями та функціями які доповнюються штучним інтелектом. Для цього ми обрали готові моделі, та написали функція яка займається скачуванням та створенням цих моделей в потрібних директоріях.

Приклад коду який займається завантаженням моделей та додаванням їх в директорії:

```
import requests
from tqdm import tqdm
import zipfile
import tarfile
import os
import shutil
def modified_download(url: str, filename: str, unzip=True,
unzip_path: str = None, force_download=False,
                      force_unzip=False, clean=False) -> str:
   assert url is not None, "URL cannot be None!"
    assert filename is not None, "Parameter filename cannot be
None!"
   ret_path = None
   embed = False
```

```
    # Внесення змін у URL для отримання даних з іншого джерела
    if 'iframe' in url:
        url = url.split('src="')[1].split('"')[0]
    if 'embed' in url:
        url = url.replace('embed', 'download')
        embed = True
    elif not url.endswith("?download=1"):
        url = url.split("?")[0] + "?download=1"
    try:
        response = requests.get(url, stream=True)
        if not embed:
            if 'id=' in response.url and '&' in response.url:
                                                                fname    =
response.url.split("id=")[-1].split("&")[0].split("%2F")[-1].spl
it('?')[0]
            else:
                fname = response.url.split('/')[-1].split('?')[0]
        else:
            fname = response.url.split('/')[-1].split('?')[0]
            if os.path.split(filename)[-1] == '' or '.' not in
os.path.split(filename)[-1]:
            filename = os.path.join(filename, fname)
                                          total_size_in_bytes     =
int(response.headers.get('content-length', 0))
        block_size = 1024
        ret_path = filename

        if not os.path.exists(filename) or force_download:
                    progress_bar = tqdm(total=total_size_in_bytes,
unit='iB',
                                                unit_scale=True) if
total_size_in_bytes > 1024 else None
            _create_if_not_exists(filename)
            with open(filename, 'wb') as f:
                for data in response.iter_content(block_size):
                    if progress_bar is not None:
                        progress_bar.update(len(data))
                    f.write(data)
            if progress_bar is not None:
                progress_bar.close()
                 if total_size_in_bytes != 0 and progress_bar.n !=
total_size_in_bytes:
                    raise Exception(
                              f"ERROR, something went wrong during
download!\nExpected     {total_size_in_bytes}     Bytes,     got
{progress_bar.n} Bytes.")

        if unzip:
                                if   filename.endswith(".zip")   or
filename.endswith(".tar.gz"):
                print("Unzipping file...")
```

```
                unzip_path = unzip_path if unzip_path is not None
else os.path.split(filename)[0]
                            clean_unzip_path = force_unzip and
os.path.realpath(unzip_path) not in os.path.realpath(filename)
            ret_path = unzip_path
                                _create_if_not_exists(unzip_path,
remove=clean_unzip_path)
            if force_unzip:
                print("Warning: overwriting existing files!")
            if filename.endswith(".zip"):
                        with zipfile.ZipFile(filename, 'r') as
zip_ref:
                                                    for file in
tqdm(iterable=zip_ref.namelist(), total=len(zip_ref.namelist()),
                                                  desc="Extracting
files"):
                                                        if not
os.path.exists(os.path.join(unzip_path, file)) or force_unzip:
                                        zip_ref.extract(member=file,
path=unzip_path)
            elif filename.endswith(".tar.gz"):
                        with tarfile.open(filename, 'r:gz') as
tar_ref:
                                                    for file in
tqdm(iterable=tar_ref.getnames(), total=len(tar_ref.getnames()),
                                                  desc="Extracting
files"):
                                                        if not
os.path.exists(os.path.join(unzip_path, file)) or force_unzip:
                                        tar_ref.extract(member=file,
path=unzip_path)
            if clean:
                os.remove(filename)
        return ret_path
    except Exception as e:
        print(e)
          raise Exception("ERROR, something went wrong, see error
above!")


if __name__ == '__main__':
    filename = "unique_data.zip"
    link = "https://example.com/unique_data_source"
         modified_download(link,  filename=filename,  unzip=True,
unzip_path=".")
    os.remove(filename)
```

В цілому це велика функція для завантаження файлів наших моделей з інтернету. Вона приймає URL-адресу файлу, куди файл потрібно завантажити, і ім'я файлу, під яким він буде збережений на локальному

комп'ютері. Опціонально, функція може розпакувати архів, якщо завантажений файл є zip-архівом або tar.gz-архівом. Також є можливість очистити завантажений архів після розпакування. Функція перевіряє наявність файлу на локальному комп'ютері та, в разі необхідності, завантажує його. Він потрібен для того, щоб користувач який завантажив нашу програму міг користуватись всім функціоналом.

3.4 Інші допоміжні інструменти для розробки

Один з найважливіших інструментів для розробки нашого проекту - система керування версіями Git. Я обрав Git через його надійність, ефективність та гнучкість. Git забезпечує можливість нелінійної розробки за допомогою гілок, а також забезпечує цілісність історії змін та стійкість до змін. Криптографічні методи використовуються для забезпечення цілісності історії та можливості прив'язування цифрових підписів розробників до тегів і комітів. Git - вільна та відкрита система, що дозволяє використовувати її як основу для спеціалізованих систем керування версіями або користувацьких інтерфейсів.

Наприклад, існують інтерфейси Git, такі як Cogito та StGit, які додатково розширюють можливості Git. Інтерфейси користувача, такі як gitk та git-gui, розповсюджуються разом з Git і дозволяють зручно працювати з репозиторіями. Для віддаленого доступу до репозиторіїв Git можна використовувати git-демон, SSH або HTTP сервер. Кожен з цих методів має свої переваги, і вони можуть бути використані в залежності від потреб.

Обравши Git для проекту, я врахував його популярність, надійність та гнучкість. Ця система керування версіями ідеально підходить для проектів різного масштабу, забезпечуючи ефективне керування змінами і спільною роботою над кодом та даними.

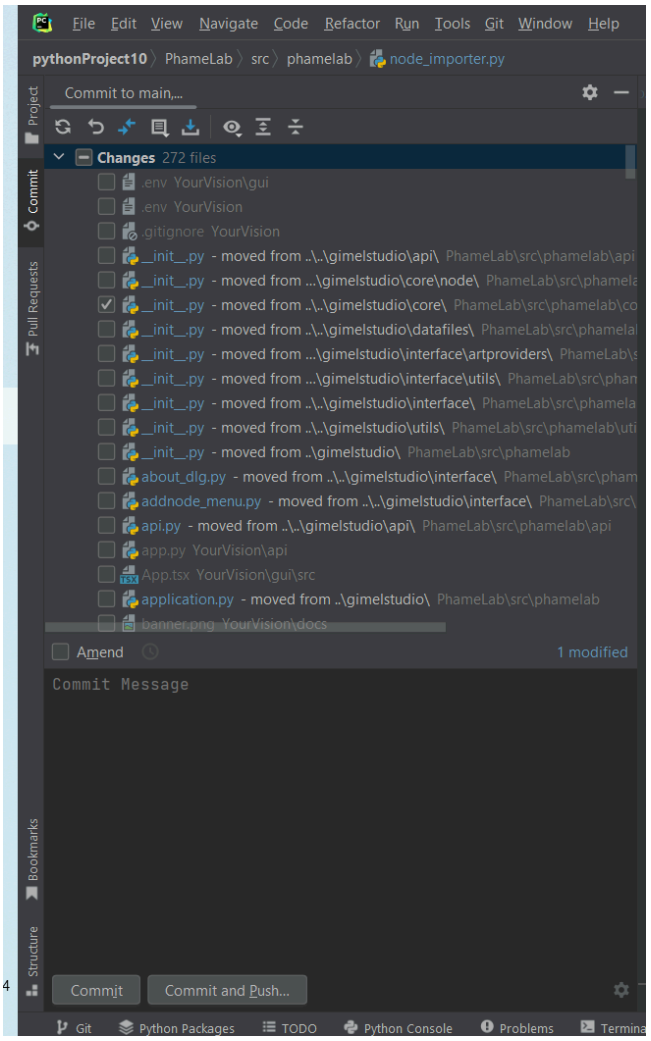

Рисунок 3.3 — Користувацький інтерфейс Git в Pycharm

Одна з переваг які ми отримали при використанні Pycharm це внутрішня Git система для користування файлами. Ми маємо можливість обрати будь які файли які ми змінили, та зберегти їх в хранилищі для подальшого використання, це допоможе нам підтримувати програму та залучити інших спеціалістів з усього світу для підтримки та модифікації програми. Так як одна з наших цілей це опенсорс напрямок, де кожен програміст ентузіаст може працювати над програмою та удосконалювати її в тому напрямку який буде його задовольняти. Однак варто уточнити, що Pycharm Git це лише одна з можливостей, майже кожна IDE для Python чи редактор має внутрішній інтерфейс для користування Git.

3.5 Функції штучного інтелекту

Було приділено багато уваги функціональним можливостям програми, що мають потенціал для використання штучного інтелекту, нижче буде описані всі функціональні можливості нашої програми:

1) масштабування та панорама;
2) вибір кольору;
3) вибір прямокутника;
4) вибір шляху;
5) обрізка;
6) малювання та стирання;
7) коригування експозиції та кольору;
8) переглядач гістограми;
9) редактор кривих;
10) інструмент видалення плям;
11) інструмент розмиття;
12) обертання ліворуч/праворуч;
13) горизонтальне/вертикальне накладання;
14) горизонтальне/вертикальне відображення;
15) зшивання панорамних зображень;
16) фільтри Instagram.

AI Функціонал:

1) коригування балансу білого;
2) видалення фону;
3) сегментація людини;
4) сірий фон;
5) режим портрету з розмиттям заднього плану;
6) інтерактивне кольорове забарвлення;
7) супер-роздільна здатність;
8) стиль аніме.

З всіх цих функцій було приділено більше уваги саме на формат виведення зображення та покращення адаптивності рівня яскравості, контрасту, враховуючи його контекст. Наш ШІ працює над всіма цими функції та в залежності від функції оптимізує процеси над зображенням. Для всіх цих процесів було використовувано бібліотеки scit-image та torch з предналаштованими моделями.

3.6 Випробування функцій

Наступним нашим етапом стало випробування функцій штучного інтелекту, для цього ми взяли просту картинку котика з інтернету, та проведемо пару функціональних тестів через ручне тестування.

Рисунок 3.4 — Інтерфейс редактора з зображенням кота

Далі ми будемо пробувати різні функції редактора на коті.

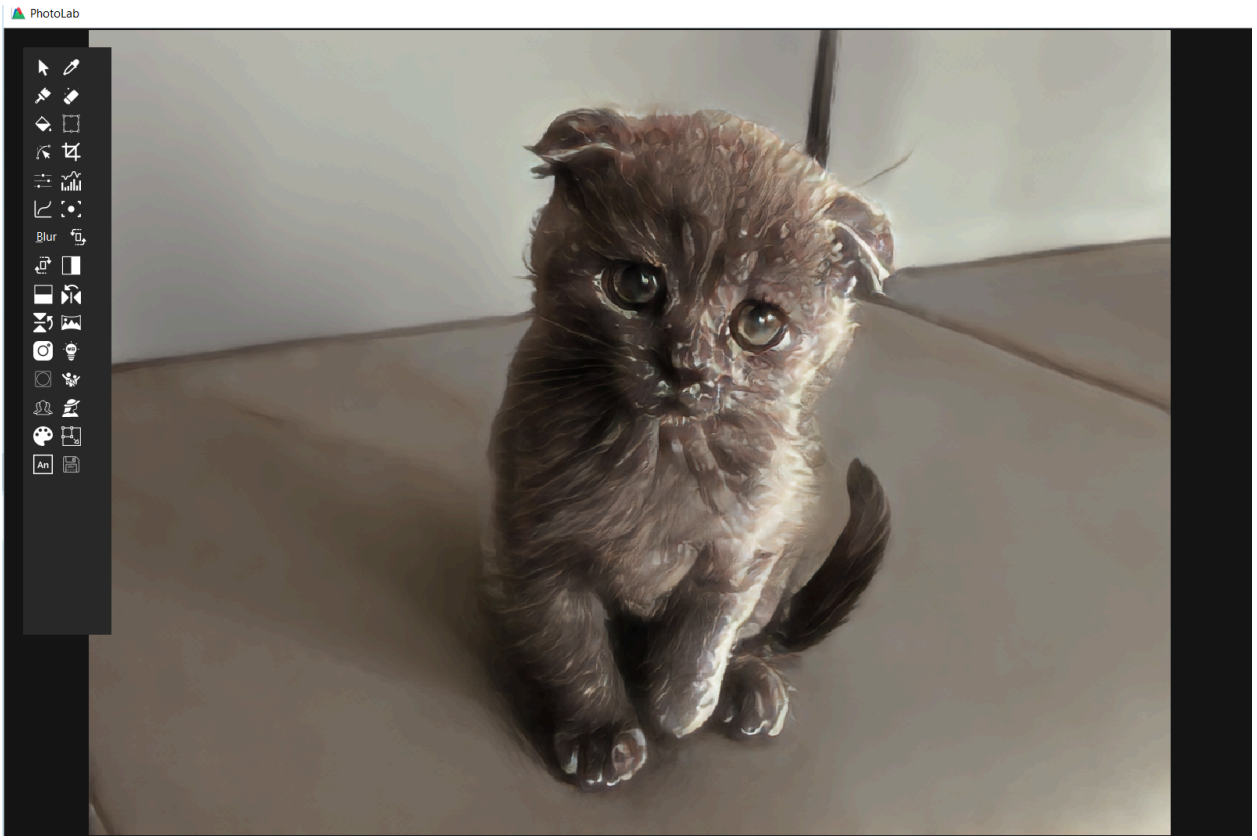

Рисунок 3.5 — Аніме стиль

На рисунку 3.5 зображено кота з використанням аніме стилю

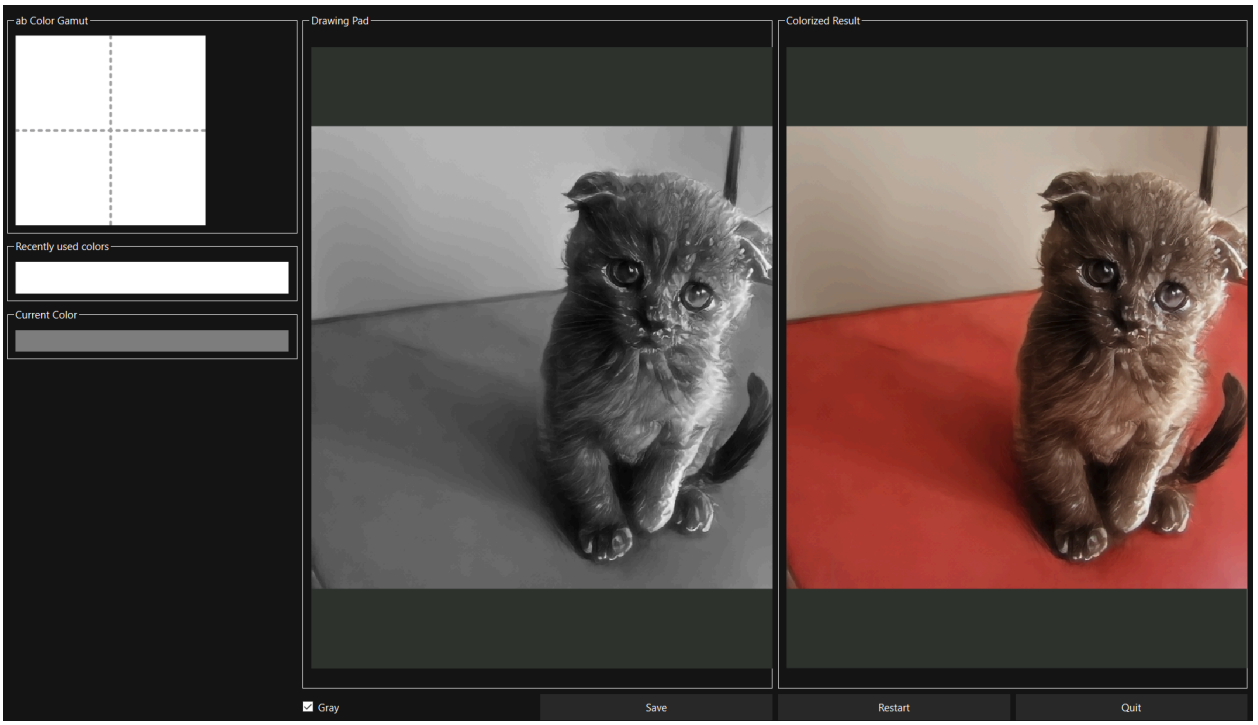

Рисунок 3.6 — Інтерактивна колоризація

На рисунку 3.6 зображено кота з використанням колоризації

Рисунок 3.7 — Налаштування сірого фону

На рисунку 3.7 зображено кота з використанням функції налаштування сірого фону.

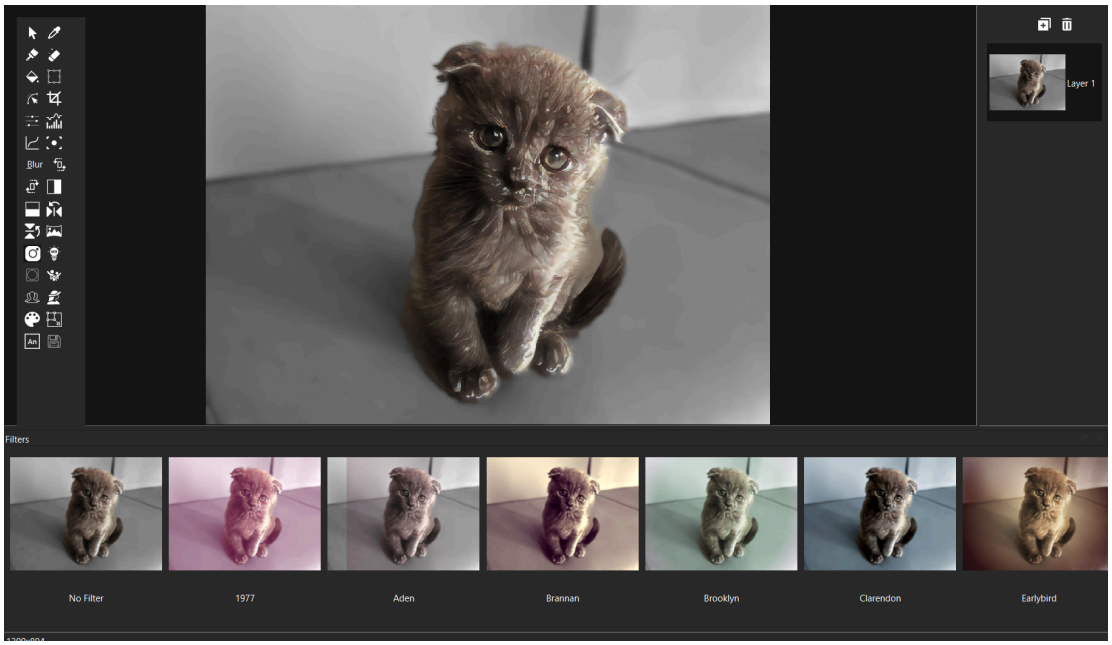

Рисунок 3.8 Інстаграм фільтри

На рисунку 3.8 — зображено кота з використанням фільтрів інстаграму

Рисунок 3.9 — Супер-роздільна здатність

На рисунку 3.9 зображено кота з використанням роздільної здатності

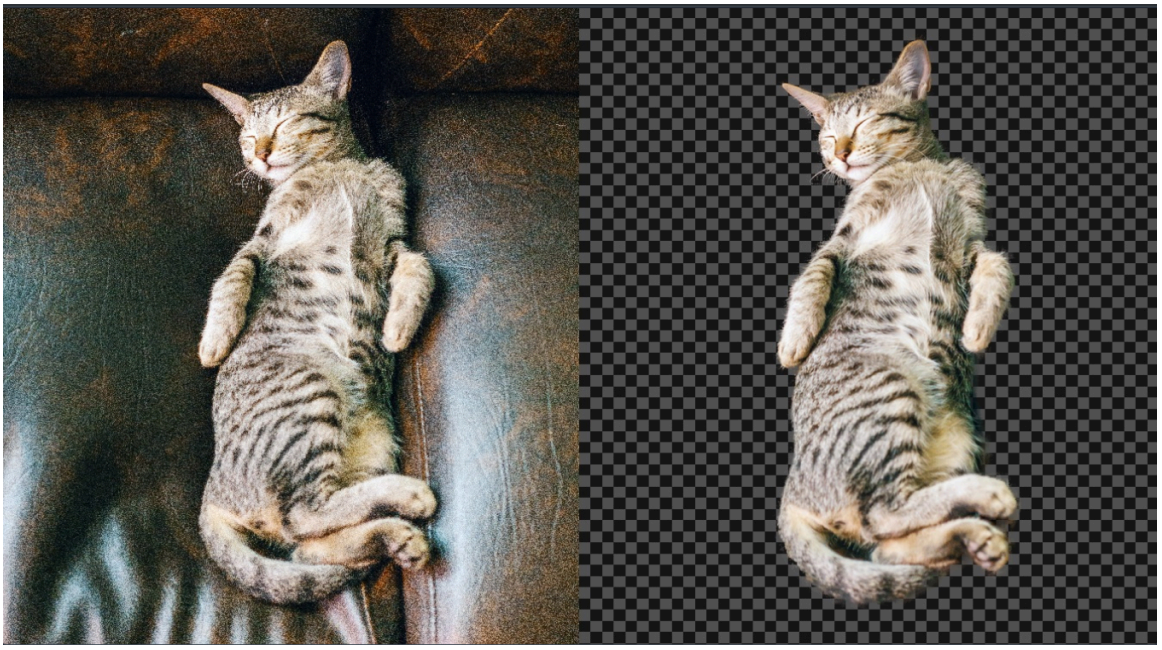

Рисунок 3.10 — Бекграунд видалення

На рисунку 3.10 зображено кота з видаленням бекграунду.

ВИСНОВОК

У даній кваліфікаційній роботі було досліджено та реалізовано фоторедактор з використанням функцій штучного інтелекту. Розроблений редактор має за мету полегшити процес обробки фотографій та підвищити якість вихідних зображень.

Актуальність теми кваліфікаційної роботи обумовлена швидким розвитком сучасних технологій, зростанням популярності соціальних мереж та потребою високоякісних зображень у сфері бізнесу та реклами. Інтеграція штучного інтелекту в обробку фотографій стає важливим напрямком розвитку програмного забезпечення для редагування зображень.

Ціллю роботи було дослідити можливості застосування штучного інтелекту в обробці фотографій та розробити програмне забезпечення, що використовує ці знання. Завданнями було вивчення функціоналу обробки зображень, розробка програмного забезпечення з необхідним набором інструментів, тестування та валідація розробленого редактора.

Результатом роботи став розроблений фоторедактор, який має ряд корисних функцій, таких як видалення об'єктів з фотографій, покращення якості зображень, автоматична ретуш та інші. Цей редактор може стати корисним інструментом для фотографів, дизайнерів та інших користувачів, що працюють з зображеннями.

У подальшому можливе розширення функціоналу редактора, вдосконалення інтерфейсу користувача та підвищення продуктивності програми. Також можливе порівняння розробленого редактора з іншими програмами на ринку для оцінки його конкурентоспроможності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bianco, S., Cusano, C., Piccoli, F., & Schettini, R. (2020). Personalized Image Enhancement Using Neural Spline Color Transforms. IEEE Transactions on Image Processing, 29, 6223-6236. .

2. Tseng, E., Zhang, Y., Jebe, L., Zhang, X., Xia, Z., Fan, Y., Heide, F., & Chen, J. (2022). Neural Photo-Finishing. ACM Transactions on Graphics (TOG), 41, 1 - 15. https://doi.org/10.1145/3550454.3555526.

3. Liu, Y., He, J., Chen, X., Zhang, Z., Zhao, H., Dong, C., & Qiao, Y. (2021). Very Lightweight Photo Retouching Network With Conditional Sequential Modulation. IEEE Transactions on Multimedia, 25, 4638-4652. https://doi.org/10.1109/TMM.2022.3179904.

4. Golchubian, A., Marques, O., & Nojoumian, M. (2021). Photo quality classification using deep learning. Multimedia Tools and Applications, 80, 22193 - 22208. https://doi.org/10.1007/s11042-021-10766-7.

5. O'Quinn, W., & Haddad, R. (2018). Image Quality Enhancement Using Machine Learning. SoutheastCon 2018, 1-5. https://doi.org/10.1109/SECON.2018.8479289.

6. Kriegeskorte, N., & Golan, T. (2019). Neural network models and deep learning. Current Biology, 29, R231-R236. https://doi.org/10.1016/j.cub.2019.02.034.

7. Brock, A., Lim, T., Ritchie, J., & Weston, N. (2016). Neural Photo Editing with Introspective Adversarial Networks. ArXiv, abs/1609.07093.

8. Oh, B., Chen, M., Dorsey, J., & Durand, F. (2001). Image-based modeling and photo editing. Proceedings of the 28th annual conference on Computer graphics and interactive techniques. https://doi.org/10.1145/383259.383310.

9. Ditrih, H., & Grgic, S. (2020). PhotoGrade – Fast and Effective Application for Digital Photo Editing. 2020 International Symposium ELMAR, 49-52. https://doi.org/10.1109/ELMAR49956.2020.9219012.

10. Bharti, S., Inani, H., & Gupta, R. (2022). Smart Photo Editor using Generative Adversarial Network: A Machine Learning Approach. 2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), 1-6. https://doi.org/10.1109/iSSSC56467.2022.10051480.

Додаток А – код програми

```
from turtle import width
import PyQt6
from PyQt6 import QtWidgets, QtCore, QtGui
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import (
    QApplication,
    QLabel,
    QSlider,
    QToolBar,
    QToolButton,
    QFileDialog,
    QStatusBar
)
from PyQt6.QtGui import QPixmap
import sys

from QImageViewer import QtImageViewer
from PyQt6.QtGui import QKeySequence
import pyqtgraph as pg
from QColorPicker import QColorPicker
import os
from QFlowLayout import QFlowLayout
from PIL import Image, ImageEnhance, ImageFilter
from QWorker import QWorker
import QCurveWidget

def free_gpu_cache():
    import torch
    from GPUtil import showUtilization as gpu_usage

    print("Initial GPU Usage")
    gpu_usage()

    torch.cuda.empty_cache()

    print("GPU Usage after emptying the cache")
    gpu_usage()

def importLibraries():
    import torch
    import numpy as np
    import cv2
    import PIL
    print("Torch version", torch.__version__)
    print("Torch CUDA available?", "YES" if
torch.cuda.is_available() else "NO")
    print("cv2 version", cv2.__version__)
    print("numpy version", np.__version__)
    print("PIl version", PIL.__version__)
```

```
class Gui(QtWidgets.QMainWindow):

    sliderChangeSignal = QtCore.pyqtSignal()

    def __init__(self, parent=None):
        super(Gui, self).__init__(parent)
        self.setWindowTitle('PhotoLab')
        self.setMinimumHeight(850)

        self.statusBar = QStatusBar()
        self.setStatusBar(self.statusBar)


        # Create Histogram
        # Compute image histogram
        r_histogram = []
        g_histogram = []
        b_histogram = []

        # ITU-R 601-2 luma transform:
        luma_histogram = []

        # Create histogram plot
        self.ImageHistogramPlot = pg.plot()
        x = list(range(len(r_histogram)))
        self.ImageHistogramGraphRed = pg.PlotCurveItem(x = x, y
= r_histogram, fillLevel=2, width = 1.0, brush=(255,0,0,80))
        self.ImageHistogramGraphGreen = pg.PlotCurveItem(x = x,
y = g_histogram, fillLevel=2, width = 1.0, brush=(0,255,0,80))
        self.ImageHistogramGraphBlue = pg.PlotCurveItem(x = x, y
= b_histogram, fillLevel=2, width = 1.0, brush=(0,0,255,80))
        self.ImageHistogramGraphLuma = pg.PlotCurveItem(x = x, y
= luma_histogram, fillLevel=2, width = 1.0,
brush=(255,255,255,80))

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphRed)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphGreen)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphBlue)

self.ImageHistogramPlot.addItem(self.ImageHistogramGraphLuma)
        self.HistogramContent = None
        self.ImageHistogramPlot.hide()
        # State of enhance sliders
        self.RedFactor = 100
        self.GreenFactor = 100
        self.BlueFactor = 100
        self.Temperature = 6000 # Kelvin, maps to (255,255,255),
direct sunlight
        self.Color = 100
```

```
        self.Brightness = 100
        self.Contrast = 100
        self.Sharpness = 100

        # State of filter sliders
        self.GaussianBlurRadius = 0

        self.timer_id = -1
        self.sliderExplanationOfChange = None
        self.sliderTypeOfChange = None
        self.sliderValueOfChange = None
        self.sliderObjectOfChange = None
def AddBlueColorSlider(self, layout):
        self.BlueColorSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.BlueColorSlider.setRange(0, 200) # 1 is original
image, 0 is black image
        layout.addRow("Blue", self.BlueColorSlider)

        # Default value of the Color slider
        self.BlueColorSlider.setValue(100)


self.BlueColorSlider.valueChanged.connect(self.OnBlueColorChange
d)

    def OnBlueColorChanged(self, value):
        self.BlueFactor = value
        self.processSliderChange("Blue", "Slider", value,
"BlueColorSlider")

    def AddTemperatureSlider(self, layout):
        self.TemperatureSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.TemperatureSlider.setRange(0, 12000)
        layout.addRow("Temperature", self.TemperatureSlider)

        # Default value of the Temperature slider
        self.TemperatureSlider.setValue(6000)


self.TemperatureSlider.valueChanged.connect(self.OnTemperatureCh
anged)

    def OnTemperatureChanged(self, value):
        self.Temperature = value
        self.processSliderChange("Temperature", "Slider", value,
"TemperatureSlider")

    def AddColorSlider(self, layout):
        self.ColorSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
```

```
        self.ColorSlider.setRange(0, 200) # 1 is original image,
0 is black image
        layout.addRow("Saturation", self.ColorSlider)

        # Default value of the Color slider
        self.ColorSlider.setValue(100)


self.ColorSlider.valueChanged.connect(self.OnColorChanged)

    def OnColorChanged(self, value):
        self.Color = value
        self.processSliderChange("Saturation", "Slider", value,
"ColorSlider")

    def AddBrightnessSlider(self, layout):
        self.BrightnessSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.BrightnessSlider.setRange(0, 200) # 1 is original
image, 0 is black image
        layout.addRow("Brightness", self.BrightnessSlider)

        # Default value of the brightness slider
        self.BrightnessSlider.setValue(100)


self.BrightnessSlider.valueChanged.connect(self.OnBrightnessChan
ged)

    def OnBrightnessChanged(self, value):
        self.Brightness = value
        self.processSliderChange("Brightness", "Slider", value,
"BrightnessSlider")

    def AddContrastSlider(self, layout):
        self.ContrastSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.ContrastSlider.setRange(0, 200) # 1 is original
image, 0 is a solid grey image
        layout.addRow("Contrast", self.ContrastSlider)

        # Default value of the brightness slider
        self.ContrastSlider.setValue(100)


self.ContrastSlider.valueChanged.connect(self.OnContrastChanged)

    def OnContrastChanged(self, value):
        self.Contrast = value
        self.processSliderChange("Contrast", "Slider", value,
"ContrastSlider")
```

```
    def AddSharpnessSlider(self, layout):
        self.SharpnessSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.SharpnessSlider.setRange(0, 200) # 1 is original
image, 0 is black image
        layout.addRow("Sharpness", self.SharpnessSlider)

        # Default value of the Sharpness slider
        self.SharpnessSlider.setValue(100)


self.SharpnessSlider.valueChanged.connect(self.OnSharpnessChange
d)

    def OnSharpnessChanged(self, value):
        self.Sharpness = value
        self.processSliderChange("Sharpness", "Slider", value,
"SharpnessSlider")

    def AddGaussianBlurSlider(self, layout):
        self.GaussianBlurSlider =
QSlider(QtCore.Qt.Orientation.Horizontal)
        self.GaussianBlurSlider.setRange(0, 2000)
        layout.addRow("Gaussian Blur", self.GaussianBlurSlider)

self.GaussianBlurSlider.valueChanged.connect(self.OnGaussianBlur
Changed)

    def OnGaussianBlurChanged(self, value):
        self.GaussianBlurRadius = value
        self.processSliderChange("Gaussian Blur", "Slider",
value, "GaussianBlurSlider")

    def UpdateHistogramPlot(self):
        # Compute image histogram
        img = self.QPixmapToImage(self.image_viewer.pixmap())
        r, g, b, a = img.split()
        r_histogram = r.histogram()
        g_histogram = g.histogram()
        b_histogram = b.histogram()

        # ITU-R 601-2 luma transform:
        luma_histogram = [sum(x) for x in zip([item *
float(299/1000) for item in r_histogram],
                                              [item *
float(587/1000) for item in g_histogram],
                                              [item *
float(114/1000) for item in b_histogram])]

        # Update histogram plot
        self.ImageHistogramGraphRed.setData(y=r_histogram)
        self.ImageHistogramGraphGreen.setData(y=g_histogram)
```

```
        self.ImageHistogramGraphBlue.setData(y=b_histogram)
        self.ImageHistogramGraphLuma.setData(y=luma_histogram)

    @QtCore.pyqtSlot()
    def onUpdateImageCompleted(self):
        if self.sliderChangedPixmap:
            self.image_viewer.setImage(self.sliderChangedPixmap,
False, self.sliderExplanationOfChange,
                                       self.sliderTypeOfChange,
self.sliderValueOfChange, self.sliderObjectOfChange)
            self.UpdateHistogramPlot()

    def timerEvent(self, event):
        self.killTimer(self.timer_id)
        self.timer_id = -1

        Pixmap = self.image_viewer.getCurrentLayerLatestPixmap()
        OriginalPixmap = Pixmap.copy()

        # TODO: If a selection is active
        # Only apply changes to the selected region
        if self.image_viewer._isSelectingRect:
            print(self.image_viewer._selectRect)
            Pixmap =
Pixmap.copy(self.image_viewer._selectRect.toRect())
        elif self.image_viewer._isSelectingPath:
            Pixmap =
self.image_viewer.getSelectedRegionAsPixmap()

        if Pixmap:
            if self.RedFactor != 100:
                Pixmap = self.UpdateReds(Pixmap,
float(self.RedFactor / 100))
            if self.GreenFactor != 100:
                Pixmap = self.UpdateGreens(Pixmap,
float(self.GreenFactor / 100))
            if self.BlueFactor != 100:
                Pixmap = self.UpdateBlues(Pixmap,
float(self.BlueFactor / 100))
            if self.Temperature != 6000:
                import AdjustTemperature
                img = self.QPixmapToImage(Pixmap)
                import numpy as np
                import cv2
                arr = np.asarray(img)
                b, g, r, a = cv2.split(arr)
                img = Image.fromarray(np.dstack((b, g, r)))

                def FindClosest(lst, K):
                    return lst[min(range(len(lst)), key = lambda
i: abs(lst[i]-K))]
```

```
                img = AdjustTemperature.convert_temp(img,
FindClosest(list(AdjustTemperature.kelvin_table.keys()),
self.Temperature))
                img = np.asarray(img)
                img = np.dstack((img, a))
                img = Image.fromarray(img)
                Pixmap = self.ImageToQPixmap(img)
            if self.Color != 100:
                Pixmap = self.EnhanceImage(Pixmap,
ImageEnhance.Color, self.Color)
            if self.Brightness != 100:
                Pixmap = self.EnhanceImage(Pixmap,
ImageEnhance.Brightness, self.Brightness)
            if self.Contrast != 100:
                Pixmap = self.EnhanceImage(Pixmap,
ImageEnhance.Contrast, self.Contrast)
            if self.Sharpness != 100:
                Pixmap = self.EnhanceImage(Pixmap,
ImageEnhance.Sharpness, self.Sharpness)
            if self.GaussianBlurRadius > 0:
                Pixmap = self.ApplyGaussianBlur(Pixmap,
float(self.GaussianBlurRadius / 100))

            if self.image_viewer._isSelectingRect:
                painter = QtGui.QPainter(OriginalPixmap)
                selectRect = self.image_viewer._selectRect
                point = QtCore.QPoint(int(selectRect.x()),
int(selectRect.y()))
                painter.drawPixmap(point, Pixmap)
                painter.end()
                Pixmap = OriginalPixmap
            elif self.image_viewer._isSelectingPath:
                painter = QtGui.QPainter(OriginalPixmap)
                painter.drawPixmap(QtCore.QPoint(), Pixmap)
                painter.end()
                Pixmap = OriginalPixmap

            self.sliderChangedPixmap = Pixmap
            self.sliderExplanationOfChange =
self.sliderExplanationOfChange
            self.sliderTypeOfChange = self.sliderTypeOfChange
            self.sliderValueOfChange = self.sliderValueOfChange
            self.sliderObjectOfChange =
self.sliderObjectOfChange
            self.sliderChangeSignal.emit()

    def RemoveRenderedCursor(self):
        # The cursor overlay is being rendered in the view
        # Remove it
        if any([self.image_viewer._isBlurring,
self.image_viewer._isRemovingSpots]):
            pixmap = self.getCurrentLayerLatestPixmap()
```

```
            self.image_viewer.setImage(pixmap, False)

    def InitTool(self):
        self.RemoveRenderedCursor()

    def OnCursorToolButton(self, checked):
        self.InitTool()
        self.EnableTool("cursor") if checked else
self.DisableTool("cursor")

    def OnColorPickerToolButton(self, checked):
        if checked:
            self.InitTool()
            class ColorPickerWidget(QtWidgets.QWidget):
                def __init__(self, parent, mainWindow):
                    QtWidgets.QWidget.__init__(self, parent)
                    self.parent = parent
                    self.closed = False
                    self.mainWindow = mainWindow

                def closeEvent(self, event):
                    self.destroyed.emit()
                    event.accept()
                    self.closed = True
                    self.mainWindow.DisableTool("color_picker")

            self.ColorPickerContent = ColorPickerWidget(None,
self)
            ColorPickerLayout =
QtWidgets.QVBoxLayout(self.ColorPickerContent)
            self.color_picker =
QColorPicker(self.ColorPickerContent, rgb=(173, 36, 207))
            self.image_viewer.ColorPicker = self.color_picker
            ColorPickerLayout.addWidget(self.color_picker)
            self.EnableTool("color_picker") if checked el
```