Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія цифрової обробки фотографії»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Данилишин В.В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. КН
Колодний В.В.
(прізвище та ініціали)
«7» 12 2023 р.

Опонент: к.т.н., доц. каф. АІІТ
Барабан М.В.
(прізвище та ініціали)
«7» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«8. 12» 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 122 – Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
_____
(підпис)
"29" 08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Данилишину Владиславу Вікторовичу

1 Тема роботи: «Інформаційна технологія цифрової обробки фотографії».
Керівник роботи: Колодний Володимир Володимирович, к.т.н., доц. каф. КН.
затверджені наказом вищого навчального закладу «18» 09 2023 року № 247

2 Строк подання студентом роботи 13.11.2023

3 Вихідні дані до роботи: формат зображення – jpeg; роздільна здатність зображення - не більше 1920*1080 піксель; кількість класів - не менше 4шт, мова програмування - об'єктно-орієнтована.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасного стану інформаційної технології цифрової обробки фотографій; розробка інформаційної технології цифрової обробки фотографії; програмна реалізація інформаційної технології цифрової обробки фотографії; економічна частина, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму цифрової обробки фотографій; структура нейронної мережі цифрової обробки фотографій; структура інформаційної технології цифрової обробки фотографій; приклад роботи програмного засобу цифрової обробки фотографій

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Колодний В.В., к.т.н., доцент каф. КН	07.09.23	21.09.23
4	Адлер О.О., к.т.н., доцент каф. ЕПВМ	16.10.23	16.10.23

7 Дата видачі завдання 29.08.23

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану інформаційної технології цифрової обробки фотографій Постановка задач дослідження	01.09.23 - 07.09.23	
2	Розробка інформаційної технології цифрової обробки фотографій	08.09.23 - 24.09.23	
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка ефективності	25.09.23 - 15.10.23	
4	Підготовка економічної частини	16.10.23 - 20.10.23	
5	Апробація та/або впровадження результатів дослідження	21.10.23 - 05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	06.11.23 - 10.11.23	

Студент _____ (підпис) Данилишин В.В. (прізвище та ініціали)

Керівник роботи _____ (підпис) Колодний В.В. (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.411

Данилишин В.В. Інформаційна технологія цифрової обробки фотографії. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 98с.

На укр. мові. Бібліогр.: 20 назв; рис.: 18; табл. 14.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології цифрової обробки фотографії. Технологія призначена для удосконалення процесів розфарбовування чорно-білих зображень.

В ході роботи проведено аналіз предметної області цифрової обробки фотографій. Розглянуто програми аналоги цифрової обробки фотографій. Удосконалено структуру нейронної мережі таким чином, щоб покращити якість цифрової обробки фотографій. Розроблено інформаційну технологію цифрової обробки фотографій на основі згорткової нейронної мережі та реалізовано на мові програмування Python програмний засіб для цифрової обробки фотографій на основі згорткової нейронної мережі.

В розділі економічної частини проведено оцінювання комерційного потенціалу розробки інформаційної технології цифрової обробки фотографії, спрогнозовано витрати на виконання наукової роботи та впровадження результатів, які склали 125123,2 грн, розраховано період окупності – 0,85 року.

Ключові слова: інформаційна технологія, фотографія, розфарбовування, обробка фотографії.

ABSTRACT

Danylyshyn V.V. Information technology of digital photo processing. Master's thesis on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 98 p.

In Ukrainian speech Bibliography: 20 titles; Fig.: 18; table 14.

This master's thesis is devoted to the development of information technology for digital photo processing. The technology is designed to improve the coloring processes of black and white images.

In the course of the work, an analysis of the subject area of digital photo processing was carried out. Analogous programs for digital photo processing are considered. The structure of the neural network has been improved in such a way as to improve the quality of digital photo processing. The information technology of digital photo processing based on a convolutional neural network was developed and a software tool for digital photo processing based on a convolutional neural network was implemented in the Python programming language.

In the section of the economic part, an assessment of the commercial potential of the development of information technology for digital photo processing was carried out, the costs for the implementation of scientific work and the implementation of the results were predicted, which amounted to UAH 125123.2, and the payback period was calculated - 0.85 years.

Keywords: information technology, photography, coloring, photo processing.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Тривалий час телебачення та фотографія були чорно-білими. Але з розвитком технологій, кольорове телебачення та фотографії, в середині XX століття, стали рости все більше та більше.

Розфарбовування – різновид технології кольорового кінематографа або фотографії, в якій вихідне чорно-біле зображення перетворюється в кольорове.

Ще в кінці XIX століття застосовувалось розфарбування кіноплівки аніліновими барвниками. У 1950-х – 1960-х роках багато старих мультфільмів були випущені в кольорі за допомогою розфарбовування. Робота полягала в перемальовуванні старих кадрів і їх ручному розфарбовуванні.

До появи цифрових технологій розфарбовування чорно-білих фотографій проводили вручну. Фотографії з паперу були намальовані аквареллю, олійними фарбами, кольоровими олівцями та пастелями. Інструменти були пензлем або спреєм. Популярність фарбування була в середині XIX - початку XX століття, поки на ринку не з’явилися багатошарові кольорові фотоматеріали.

З розвитком комп’ютерних технологій робота спростилася, але все ж вимагала кропіткої роботи художників. Основна проблема полягає в тому, що навіть при використанні комп’ютерів робота вимагає великих витрат. Кожен кадр необхідно розділити на багато сегментів вручну. Ця проблема тривалий час не могла отримати гідне автоматизоване рішення.

І лише за останні роки вдалося досягти хороших результатів у вирішенні проблеми автоматизованого розфарбовування. Причиною цього є поширення нейронних мереж, зокрема згорткових мереж.

В даний час існує кілька високоякісних реалізацій для розфарбовування чорно-білих зображень, однак покращення якості розфарбовування за допомогою нових модифікованих алгоритмів на основі штучних нейронних мереж є важливим та актуальним питанням.

Саме тому дуже доцільною є розробка інформаційної технології, яка вирішує задачу розфарбовування чорно-білих зображень.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення якості цифрофої обробки фотографій.

Для досягнення поставленої мети слід розв'язати такі завдання:

- розглянути та проаналізувати існуючі програмні реалізації розв'язання задачі цифрофої обробки фотографій;
- навести стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології цифрофої обробки фотографій;
- провести тестування програмного продукту та виконати аналіз отриманих результатів;
- здійснити економічні розрахунки доцільності розробки нової інформаційної технології.

Об'єкт дослідження – процес цифрофої обробки фотографій.

Предмет дослідження – інформаційна технологія цифрофої обробки фотографій.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрової інформації, теорія штучних нейронних мереж для реалізації інформаційної технології розфарбовування чорно-білих зображень, методи математичної статистики для обрахунків результатів отриманих за допомогою програмного засобу, програмування на мовах високого рівня.

Наукова новизна одержаних результатів полягає в наступному:

Удосконалено інформаційну технологію цифрової обробки фотографій, що відрізняється від відомих використанням згорткової нейронної мережі, що забезпечило підвищення якості цифрової обробки фотографій.

Практичне значення одержаних результатів:

1. Розроблено алгоритм цифрофої обробки фотографій.
2. Розроблено програмне забезпечення цифрофої обробки фотографій.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок здобувача. Усі результати, що наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, які написано у співавторстві, здобувачу належать: особливості цифрової обробки фотографій [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» (м. Вінниця, Україна, 2023 р.). Результати дослідження впроваджено в роботу ТОВ «ІТІ».

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано 1 тезу доповіді на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЦИФРОВОЇ ОБРОБКИ ФОТОГРАФІЇ

1.1 Аналіз задачі розфарбовування чорно-білих зображень

У даній кваліфікаційній роботі вирішується задача розфарбовування чорно-білих зображень. Актуальність завдання обумовлена тим, що у кожного в дома є старі фотоальбоми, більшість фото у яких є чорно-білим, і для того, щоб оживити їх використовують метод розфарбовування. Додавання кольору до старого фото може дати більше розуміння того, що зображено на фото. Також розфарбовування використовують різні дослідники історії, які працюють з різними архівними зображеннями, тому задача розфарбовування – актуальна в наш час.

Як зазначено раніше, розфарбовування проводили вручну до розповсюдження персональних комп'ютерів. З розвитком комп’ютерних технологій розфарбовування проводилася за допомогою простих графічних редакторів. Однак складність цієї процедури без використання автоматизованого програмного забезпечення дуже висока. Друга причина дослідження – підвищення якості розфарбовування у порівнянні з існуючим програмним забезпеченням. Все ж таки якість автоматизованої колоризації на даний час страждає, хоч і швидкість колоризації збільшилась.

Процес автоматизованої колоризації можна умовно поділити на кілька етапів:

1. Сегментація – це процес поділу цифрового зображення на декілька сегментів. Мета сегментації полягає в спрощенні і зміні представлення зображення, щоб його було простіше і легше аналізувати [5]. Сегментація зображень зазвичай використовується для того, щоб виділити об'єкти і кордони (лінії, криві, і т. д.) на зображеннях. Більш точно, сегментація зображень – це процес присвоєння таких міток кожному пікселю зображення, що пікселі з однаковими мітками мають загальні візуальні характеристики;
2. Визначення кольору кожного сегменту, враховуючи значення чорно-білого каналу і цьому сегменту;
3. Забарвлення цих сегментів відповідними кольорами.

Незважаючи на те, що технологія кольорової фотографії існує з початку XX століття (зокрема метод Адольфа Міті), дійсно популярною в нашій країні вона стала не раніше 2000-х років. Але досі у багатьох сім'ях зберігаються альбоми з чорно-білими знімками, зробленими 30-40 років тому: старі картки здатні пробудити цілу гаму почуттів та спогадів, перенести крізь час.

Втім, чорно-біле зображення ніколи не виходило з моди: навіть сьогодні, в епоху цифрової революції багато майстрів продовжують працювати в цьому жанрі. Однак буває і навпаки – через роки у людини з'являється бажання перетворити чорно-білу фотографію на кольорову. Які ж способи для цього є?

Перш за все не можна не згадати про найзвичніший для більшості представників старшого покоління спосіб, коли художник розфарбовував фотографію за допомогою олійних фарб, акварелі або спеціальних анілінових барвників, які використовувалися для професійної ретуші. Нерідко при цьому оригінал набував не лише кольорової палітри, а й нових деталей. Сьогодні така методика цікава виключно з художньої та мистецтвознавчої точок зору.

Якщо виникло бажання на якийсь час відчути себе художником 50-х, можна скористайтися веб-сервісом Colorize Photo Converter. Цей невеликий фоторедактор, що працює у вікні інтернет-браузера, дозволяє вручну розфарбовувати цифрові чорно-білі знімки, підбираючи кольори з будь-якого іншого кольорового зображення, яке завантажується в сервіс користувачем, або вибирається з наведених за замовчуванням варіантів. Цей спосіб підходить для фарбування зображень, на яких не так багато невеликих об'єктів, що потребують індивідуального підходу. Наприклад, за допомогою сервісу можна швидко розфарбувати чорно-білий портрет.

Програми, що працюють на базі таких технологій, як штучний інтелект, нейронні мережі та машинне навчання, все частіше стають частиною нашого життя. Ряд інтернет-сайтів та мобільних програм пропонують широкий спектр інструментів і для прикладної фотообробки. Однак у переважній більшості випадків йдеться про фільтри, ефект яких найбільш помітний у випадку кольорових знімків. А ось програм, що дозволяють автоматично перетворити чорно-біле фото на кольорове, значно менше. Принцип роботи більшості з них схожий: йдеться про штучний інтелект (ШІ),

якому «надають» значну кількість фотографій, як кольорових, так і чорно-білих, він вивчає їх, коригує алгоритм колоризації чорно-білих зображень і пізніше застосовує до користувачів, що завантажуються. фотографіям.

Цей сервіс, розроблений групою вчених Каліфорнійського університету в Берклі (The University of California, Berkele), мабуть, на сьогоднішній день найвідоміший. Але хоча йому вже кілька років, за які їм проаналізовано мільйони фотографій, він все ще працює в демонстраційному режимі. Принаймні такий висновок можна зробити, оцінивши його URL-адресу. Спочатку проект був запущений після вивчення II мільйона зображень із бази даних ImageNet.

Працювати з Algorithmia Colorize Photos дуже просто: вставити URL-адресу зображення в спеціальне поле або завантажити оцифровану фотографію за допомогою кнопки Upload, після чого натиснути Colorize It. Іноді процес колоризації може розпочатися автоматично. Приблизно через 5–25 секунд сервіс представить результат: інтерактивне зображення, розділене навпіл фіолетовою лінією, ліворуч від якої розташована частина оригінального фото, а справа – оброблена.

Крім того, тут зазначено обмеження на розмір файлу зображення, що завантажується – 15 Мбайт. Також можна впливати на обробку фото, змінюючи пункт 'Render Factor', значення якого варіюються від 3 до 45 (хим нижче, тим більше кольорова насиченість кінцевого зображення). За моїми суб'єктивними відчуттями саме цей сервіс видає найбільш наближену до реальності кольорообробку.

Свого часу цей сервіс запустила студія Артемія Лебедєва. Як і попередні, він автоматично розфарбовує монохромні зображення, використовуючи алгоритми машинного навчання. Результати роботи приблизно на одному рівні з Algorithmia.

Додатки, що працюють на базі нейромереж, машинного навчання та ШІ, мають низку недоліків. Так, інтерфейс, пропонований користувачеві, зазвичай відрізняється різноманіттям параметрів, дозволяють впливати на результат. Крім того, кінцевий результат часто далекий від ідеалу: наприклад, фото може бути пофарбоване не повністю, якісь кольори визначені правильно, а якісь менш. А буває, що фото після обробки майже не відрізняється від оригіналу, особливо часто таке трапляється з старими знімками, що вицвіли.

Якщо користувача не влаштовують автоматичні програми на базі нейромереж, варто поекспериментувати з одним із графічних фоторедакторів. У більшості серйозних програм, будь то Photoshop або GIMP, процес колоризації складається приблизно з тих самих етапів. Способів розфарбовування досить багато, але в основі більшості лежить робота з шарами. Алгоритм приблизно такий. Створюється новий шар з копією оригінального зображення, на якому відбувається робота з фарбування окремої області (наприклад, шкіри обличчя або тканини сорочки, якщо йдеться про портрет). Кожен наступний шар відповідає певній частині оригіналу і фарбується по-різному, після чого об'єднуються всі шари. Однак це дуже короткий опис, з усіма нюансами процесу краще познайомитись за допомогою спеціальних відеоінструкцій на YouTube. По суті, йдеться про ту саму ручну ретушу, за допомогою якої розмальовували старі фотографії, лише в цифровому вигляді.

Якщо немає бажання занурюватися у вивчення професійного ПЗ, можна звернутися до відповідного фахівця. Зокрема, на сервісі Reddit Colorization Requests люди викладають чорно-білі фото, які хотіли б побачити у кольорі, та чекають на реакцію спільноти. У заголовку повідомлення слід вказати, чи готовий користувач платити за роботу (Free або Paid), а в тексті вказати, що він хоче отримати на виході. Зазвичай за кілька днів хтось із добровольців береться за таку роботу.

Варто визнати, що цифрова ретуш і колоризація не надто далеко відійшли від ручного розфарбовування фізичних фотографій – адже нейромережні алгоритми поки що вчаться вгадувати, як виглядали ті чи інші об'єкти в реальності. Так, іноді виходить чудовий результат, проте викладають, як завжди, тільки хороше, а десятки, а то й сотні провалів ніхто описувати не поспішає.

Професійна робота з серйозним програмним забезпеченням вимагає довгих годин навчання та експериментів – далеко не кожен готовий на це. Перетворення чорно-білих фотографій на кольорові як було мистецтвом у 50-ті роки, так їм залишається. Цифрова колоризація – найскладніша робота із сотнями і навіть тисячами верств. Доки будь-яка, навіть невелика деталь не буде ідеально поєднуватися з іншими (по суті, для кожного важливого елемента створюється свій шар). При цьому нерідко доводиться підключати знання з інших областей, наприклад,

ретельно вивчати історичний підтекст фотографії. Необхідно також адекватно оцінити рівень пошкодження вихідної фотографії (подряпини, пил, що в'ївся, поразка пліснявою і т. п.), щоб грамотно його усунути при обробці.

В рамках первинної реконструкції відбувається робота з фізичними вадами оригіналу, коригуються контрастність, яскравість та світлотіні. Далі йде найскладніше - горезвісна робота з кольоровими шарами. Необхідно вгадати, який колір і де використовувати, оскільки на даному етапі розвитку технологій ми все ще можемо лише припускати (якщо не маємо знань чи документованих свідчень), якого кольору, наприклад, були очі нашого прадіда і т.д. Про це можна судити завдяки тому, що якісь предмети з фотографії збереглися (наприклад, одяг чи годинник), а щодо інших залишилися достовірні відомості в архівах тощо.

Отже, проаналізувавши схожі дослідницькі роботи було зроблено висновок, що основною проблемою, яка впливає на якість колоризації, на даний момент є неправдоподібне визначення кольору деяких сегментів. Для вирішення цієї проблеми доцільно використати підказки користувача, в якості кольорових точок.

1.2 Аналіз програм аналогів цифрової обробки фотографій

Розфарбовування чорно-білих зображень є досить нетривіальним завданням. На сьогоднішній день для автоматизації таких дій розроблено не багато додатків. Найчастіше колоризація проводиться вручну - для кожної області зображення функція яскравості змінюється відповідно до потрібного кольору. Це дуже трудомістка та займає багато часу процедура. Ось чому в сучасних ринкових умовах колоризація чорно-білих зображень дорожчає, але користується великою популярністю [6].

На мій погляд, можна виділити такі програмні продукти для розфарбовування чорно-білих зображень: Adobe Photoshop, веб-сервіс ColorizePhoto, веб-сервіс Algorithmia colorize-photos, Akvis Coloriage. Перші 2 варіанти працюють без штучного інтелекту, на відміну від двох останніх.

Кожен з цих технічних засобів має свої плюси і мінуси, але процес забарвлення скрізь однаковий – площа однотонної монохромної частини виділяється і обробляється окремо. У рамках цьої кваліфікаційної роботи ця дія автоматизована.

Adobe Photoshop – графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe [7]. Photoshop головним чином призначений для редагування цифрових фотографій та створення растрової графіки, головне вікно якого зображено на рисунку 1.1. Але якщо розглядати це програмне забезпечення, як інструмент для колоризації, то варто зазначити, що використовуючи його потрібно мати великий досвід в обробці зображень, високого рівня художні навики. Якщо усі ці фактори об’єднати, то результат колоризації буде дуже якісним. Проте цей позитивний фактор нівелюється, так як цей процес займе великий проміжок часу, якщо порівнювати з іншими програмними продуктами для колоризації.

Рисунок 1.1 – головне вікно програми Adobe Photoshop CS6

Веб-сервіс ColorizePhoto – веб-додаток для колоризації без використання штучного інтелекту [8]. Механізм колоризації у ньому ідентичний до того механізму, що і Adobe Photoshop. Проте його функціонал призначений суто для колоризації. Високої якості розфарбування можна добитись, але як і в попередньому програмному

забезпеченні, але це трудомісткий і досить довгий процес. Вікно цього веб-додатку зображено на рис. 1.2. Ще одним програмним забезпеченням є веб-сервіс Algorithmia Colorize Photos. Сервіс Colorize Photos створений групою дослідників, що займаються нейронними мережами, для демонстрації одного з варіантів практичного застосування штучного інтелекту. В даному випадку він використовується для розфарбовування чорно-білих фотографій. Будь-який відвідувач сайту може ввести в поле посилання на який-небудь старий знімок і натисненням однієї кнопки перетворити його в кольоровий [9]. Незважаючи на те, що знімки обробляються всього за кілька секунд, обсяг роботи, що виконує штучний інтелект, дуже великий. Адже за цей час йому потрібно проаналізувати вміст картинки, розпізнати на ній знайомі об'єкти, витягти зі своєї бази даних приблизні кольори цих об'єктів і, нарешті, застосувати їх до фотографії.

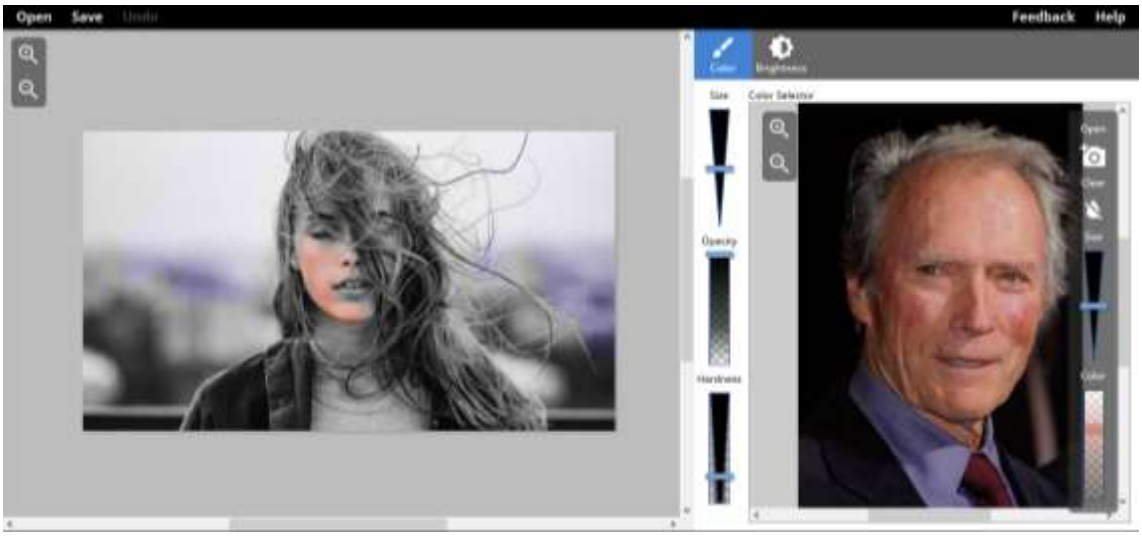

Рисунок 1.2 – вікно веб-додатку ColorizePhoto

Результати роботи Colorize Photos досить сильно залежать від пропонованого до обробки знімка. Деякі фотографії просто дивують правдоподібністю і розмаїттям кольорів, інші виглядають, скажімо прямо, не дуже. Після декількох експериментів стає зрозуміло, що краще за все Colorize Photos справляється з кольорами людського тіла, води, дерев, автомобілів, будинків.

Аналізуючи роботу цього сервісу можна дійти до висновку, що на відміну від попередніх програмних продуктів, даний проводить колоризацію за лічені секунди, проте якість цієї колоризації залишає бажати кращого. Вікно з прикладом колоризації за допомогою Algorithmia Colorize Photos зображено на рисунку 1.3.

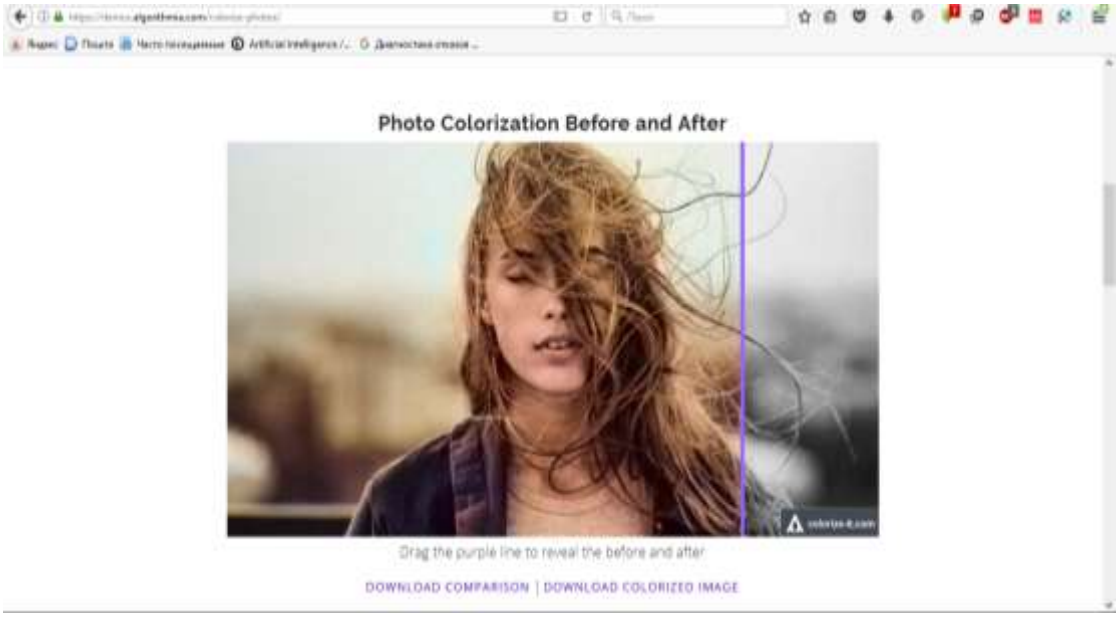

Рисунок 1.3 – вікно колоризації Algorithmia Colorize Photos

Також розглянемо Coloriage від компанії Akvis. AKVIS Coloriage – програма для розфарбовування чорно-білих фотографій і заміни кольору на кольорових зображеннях [10].

Програма однаково добре працює для обробки портретних, пейзажних фотографій, натюрмортів. Coloriage дозволяє додати колір в старі відсканові чорно-білі знімки, замінити деякі кольори на кольорових цифрових зображеннях, розфарбувати олівцеві чорно-білі малюнки, комікси, карикатури. Coloriage допомагає при підборі колірної гами інтер'єру або одягу, дає можливість поекспериментувати із зовнішністю. Колоризацію за допомогою AKVIS Coloriage зображенно на рисунку 1.4.

AKVIS Coloriage є потужним інструментом для колоризації зображень, який може бути використаний як окрема програма (standalone), так і як плагін (plugin) до

графічного редактора. Використання програми у двох режимах надає користувачам гнучкість у виборі оптимального підходу до редагування їх зображень.

На відміну від інших програм у даному переліку, AKVIS Coloriage виділяється своєю вражаючою функціональністю та надзвичайно зручним інтерфейсом. Вона пропонує ряд інструментів, що спрощують процес колоризації та роблять його доступним навіть для тих, хто не має глибоких знань у графічному дизайні.

Незважаючи на свої очевидні переваги, слід відзначити, що деякі користувачі можуть висловлювати певні зауваження щодо якості колоризації. Сучасні стандарти високої якості можуть ставити певні вимоги, які вимагають постійного вдосконалення програмного забезпечення для досягнення оптимальних результатів. Таким чином, важливо слідкувати за оновленнями програми та можливими вдосконаленнями в якості колоризації зображень.

Рисунок 1.4 – Приклад колоризації в AKVIS Coloriage

1.3 Постановка задачі

У ході виконання магістерської роботи необхідно вивчити предметну область процесу колоризації. Далі необхідно ознайомитися з існуючим станом справ в області колоризації, вивчити чи є аналоги і зробити висновок про необхідність розробки

нового продукту, розробити для нього технічне завдання, необхідно вибрати засоби розробки: операційну систему, мову і середовище програмування.

Отже, потрібно розробити інформаційну технологію колоризації чорно-білих зображень. Функціонально розроблений програмний додаток має дозволити користувачеві завантажити чорно-біле зображення в додаток, відобразити його на екрані і розфарбувати або автоматично, або з використанням підказок, у вигляді кольорових точок на чорно-білому зображенні. Для кожної точки-підказки програма повинна генеруватиме набір кольорів, які найбільш імовірні для розфарбування обраного сегмента зображення. Якщо ж ці кольори не задовольняють користувача, повинна бути панель з палітрою кольорів, на якій користувач зможе вибрати колір, який більше підходить для розфарбування обраного сегменту. Поруч з чорно-білим зображенням має відображатись результуюче зображення, яке буде оновлюватись після кожного запуску колоризації. Також програмна повинна зберігати готові кольорові зображення.

Чорно-біле зображення можна уявити як матрицю G з H рядків і W стовпців, де кожен елемент матриці відображає інтенсивність відповідного пікселя на зображенні. Ці значення зазвичай виражені у вигляді чисел в діапазоні від 0 до 1, де 0 вказує на повну відсутність світла (чорний колір), а 1 відповідає повній присутності світла (білий колір).

Важливо відзначити, що інтенсивність може бути представлена як цілим, так і чи дійсним число будь-якої розмірності на практиці. Зазвичай використовуються дійсні числа у вигляді десяткових дробів від 0 до 1, що полегшує обробку та аналіз зображень. Збільшення розмірності діапазону інтенсивності дозволяє отримати більше відтінків сірого на зображенні, що поліпшує його якість та дозволяє передавати більше деталей. Наприклад, перші монітори могли показувати тільки 16 різних відтінків сірого, використовуючи лише 4 біта [11].

Пікселі кольорового зображення, як правило, представляють у вигляді кортежу, як правило трійки, чисел з використанням одного з колірних просторів. Таким чином, кольорове зображення висотою H і шириною W пікселів можна представити у вигляді тензора C з розмірністю $3 * H * W$.

Задачу колоризації можна сформулювати як задачу про обчислення відповідних компонентів тензора C по відомим компонентах матриці G, що відповідає чорно-білому зображенню. Таким чином потрібно побудувати відображення

$$F: R^{W*H} \rightarrow R^{3*W*H} \quad (1.1)$$

з такими властивостям:

1. Інтенсивності пікселів одержуваного зображення повинні бути такими ж, як інтенсивності пікселів вихідного зображення.

2. Розфарбоване зображення повинно виглядати реалістично з точки зору людини.

Перша вимога може бути легко формалізована, в той час як друге виразити математично неможливо. Ця вимога суб'єктивна по своїй суті. У цьому полягає велика складність вирішення даного завдання і оцінки якості цього рішення.

1.4 Висновок до розділу 1

В даному розділі проаналізовано предметну область, зазначено актуальність дослідження, визначено основну проблему при колоризації чорно-білих зображень, що полягає в неможливості користувача впливати на результат колоризації.

Виконано аналітичний огляд аналогів вирішення технічної проблеми. Аналіз показав, не існує програмної реалізації, яка водночас дозволяє проводити швидку і якісну колоризацію чорно-білих зображень. Одним із найкращих в цьому плані можна вважати програмне забезпечення AKVIS Coloriage, яке дозволяє проводити колоризацію відносно швидко. Якість колоризації також задовільна, але на обробленому зображенні наявні артефакти, через погану сегментацію зображень, а також в певні області мають неприроднє забарвлення.

Враховуючи недоліки існуючих програмних засобів розфарбовування зображень виконано постановку задачі, виконання якої призведе підвищення якості розфарбовування зображень.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЦИФРОВОЇ ОБРОБКИ ФОТОГРАФІЇ

2.1 Обґрунтування вибору нейромережевого методу обробки зображень в задачі розфарбовування чорно-білих знімків

Завдання розфарбовування - виділення областей чорно-білих зображень одного колірного тону. Це підзавдання можна інтерпретувати як задача розпізнавання зображень. Існує три основні групи методів розпізнавання:

1. Статистичні методи.
2. Структурні методи.
3. Нейронні мережі.

Статистичний підхід базується на правилах математичної класифікації, які сформульовані і виведені з точки зору математичної статистики. Цей метод забезпечує класифікатор, коли відомі щільності розподілу для всіх наборів зображень і ймовірність зображення для кожного класу. У розпізнаванні образів невідомий об’єкт, що підлягає класифікації, представляється у вигляді вектора елементарних ознак.

У структурному розпізнаванні символів сутність представляється як сукупність елементарних частин, їх атрибутів і взаємозв’язків разом із глобальними якостями сутності. Ключовими моментами цього підходу є вибір непохідних елементів зображення, інтеграція цих елементів і пов’язаних з ними зв’язків у граматиці зображень, а також відповідне виконання відповідних процесів аналізу та розпізнавання мови.

Нейронні мережі є перспективною альтернативою традиційним методам вирішення проблем розпізнавання зображень. Сьогодні цей напрямок активно розвивається. Нейронні мережі різноманітні за обсягом, з’являються нові моделі нейронних мереж, адаптуються існуючі моделі для вирішення нових проблем тощо. Нейронна мережа - математична модель, а також її програмна або апаратна реалізація, побудована за принципом організації та функціонування біологічних нейронних мереж - мереж нервових клітин у живих організмах. Ця концепція була розроблена

під час вивчення процесів, що відбуваються в мозку, і спроб змоделювати ці процеси. Першою такою спробою були нейронні мережі W. McCulloch та W. Pitts [12].

Основні переваги нейронних мереж.

1. Усунення невідомих шаблонів. Використовуючи здатність вчитися на багатьох прикладах, нейронна мережа здатна вирішувати задачі, у яких закономірності ситуацій і зв'язок між входом і виходом невідомі. Традиційні математичні методи та експертні системи в таких випадках неприйнятні.

2. Стійкість до порушень у вхідних даних. Можливість роботи при наявності великої кількості неінформаційних вхідних шумових сигналів. Початковий скринінг не потрібен, нейронна мережа сама визначить його як невідповідне завдання і відхилить.

3. Пристосування до змін навколишнього середовища. Нейронні мережі здатні пристосовуватися до змін навколишнього середовища. Зокрема, нейронні мережі, які навчаються працювати в конкретному середовищі, можна легко перекваліфікувати на роботу в умовах невеликих коливань параметрів середовища. Більше того, в заочному середовищі (де статистика змінюється з часом) ви можете створювати нейронні мережі, які навчаються в режимі реального часу. Чим вище адаптивна здатність системи, тим вона буде стабільнішою в нестаціонарному середовищі. Слід зазначити, що адаптивність не завжди призводить до опору, іноді вона призводить до прямо протилежного результату. Наприклад, адаптивна система, де параметри швидко змінюються, також може швидко реагувати на зовнішні збудження, що призводить до втрати продуктивності.Для повного використання можливостей адаптації основні параметри системи повинні бути достатньо стабільними, щоб не враховувати зовнішні перешкоди, і достатньо гнучкими, щоб реагувати на значні зміни зовнішнього середовища [12].

4. Потенційно дуже висока швидкість. Нейронні мережі мають потенціал для швидкої продуктивності завдяки використанню масивної паралельної обробки інформації [12].

5. Стійкість до збоїв у реалізації апаратної нейронної мережі. Нейронні мережі потенційно відмовостійкі. Це означає, що за несприятливих умов їх ефективність

дещо знижується. Наприклад, якщо пошкоджено нейрон або його зв’язок, отримати збережену інформацію важко. Однак, беручи до уваги розсіяний характер зберігання інформації в нейронній мережі, можна стверджувати, що лише серйозне пошкодження структури нейронної мережі істотно вплине на її продуктивність. Тому зниження якості нейронної мережі відбувається повільно [12].

Для реалізації розфарбування чорно-білих зображень було обрано метод на основі нейронної мережі.

Нейронні мережі – це система зв’язаних і взаємодіючих нейронів. Такі нейрони зазвичай досить прості. Кожен нейрон у такій мережі взаємодіє лише з сигналами, які він періодично отримує або передає іншим нейронам. І все ж, будучи з’єднані в досить велику мережу з керованою взаємодією, такі нейрони, незважаючи на простоту кожного з них окремо, разом здатні виконувати досить складні завдання.

Нейронні мережі не програмуються в звичному розумінні цього слова, а навчаються. Навчання є однією з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно кажучи, навчання полягає у знаходженні ваги зв’язків між нейронами. Під час навчання нейронна мережа здатна знаходити складні зв’язки між входом і виходом, а також узагальнювати. Іншими словами, якщо навчання пройшло успішно, мережа зможе повернути правильний результат на основі даних, які були відсутні у навчальній вибірці, а також неповних або частково спотворених даних.

Структурно нейронна мережа являє собою сукупність нейронів, з'єднаних у шари. Кожен нейрон являє собою одиницю, яка отримує дані з багатьох різних входів з попереднього шару через синапси (рис. 2.1, 2.2). Зважена сума S всіх входів обчислюється в нейроні, а значення функції активації надається на вихід

$$Y = F(S) \tag{2.1}$$

Таке значення і є значенням аксона (рис.2.1) [12].

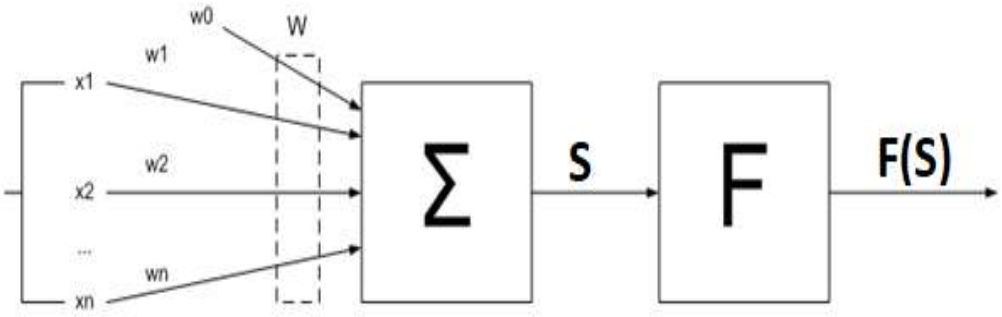

Рисунок 2.1 – Внутрішня будова штучного нейрона в нейронній мережі

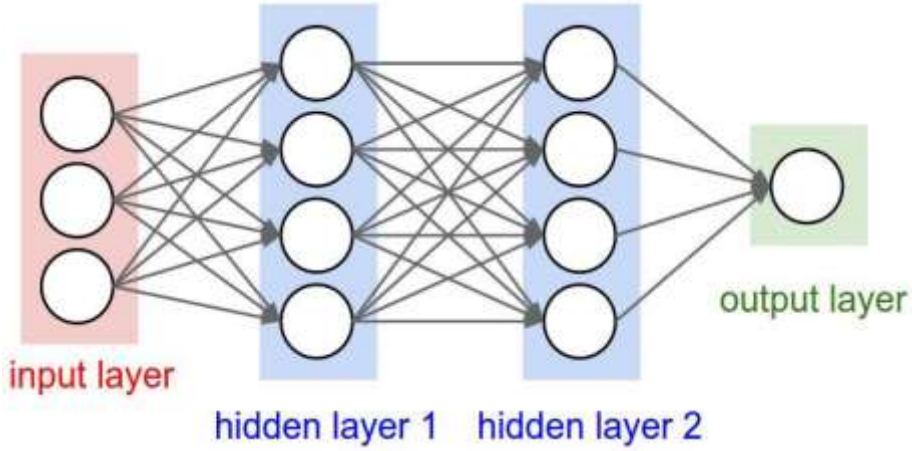

Рисунок 2.2 – З’єднання шарів в нейронній мережі

Функцію активації можна розглядати як будь-яку функцію, але часто використовуються лише деякі, оскільки існує ряд вимог, які вказують на те, що їх робота позитивно впливає на навчання нейронних мереж:

Нелінійність – якщо функція активації нелінійна, можна показати, що двошарова нейронна мережа є т.зв. універсальний апроксиматор функцій. Інакше це не має сенсу для кількох шарів, оскільки лінійна комбінація лінійних функцій є лінійною функцією. Вище було сказано, що проблему розфарбовування можна сформулювати як задачу пошуку складної карти F. Здатність нейронних мереж будувати універсальні наближення є однією з причин використання їх у цій роботі для розв’язання задачі [11].

Безперервне диференціювання – ця властивість просто необхідна для можливості застосування методу оптимізації нейронної мережі на основі розрахунку градієнтів [12].

Обмеження. Методи оптимізації схильні до нестабільності в навчанні нейронних мереж, коли функції активації мережі необмежені.

Монотонність – доведено, що якщо функція активації монотонна, то функція помилки на кожному окремому шарі є опуклою, що значно збільшує можливості розв’язування задач оптимізації, оскільки опуклі задачі оптимізації вивчені набагато краще [11].

Похідна монотонність – функції активації з цією властивістю, як було показано, значно краще узагальнені [11].

Апроксимація ідентичного відображення поблизу нуля - якщо функція активації має таку властивість, нейронна мережа буде ефективно навчатися, якщо її масштаби спочатку ініціюються з малих випадкових масштабів. В іншому випадку потрібні додаткові дослідження, щоб зрозуміти, які початкові варіанти необхідні для початку навчання [11].

2.2 Обгрунтування вибору типу нейронної мережі для роботи з зображеннями

Класичні нейронні мережі використовували лише повністю взаємопов’язані шари. Такий підхід підходить для невеликих завдань, але при роботі з зображеннями кількість параметрів в такій мережі величезна навіть для двох шарів. Тому для вирішення проблеми розфарбовування чорно-білих зображень була обрана згортка нейронна мережа.

Згорткові нейронні мережі - це клас глибоких штучних нейронних мереж прямого поширення, створених Яном Лекуном з метою ефективного вирішення проблем з розпізнаванням зображень [2].

Назва мережевої архітектури виникла тому, що вона заснована на операції згортки, суть якої полягає в тому, що кожен піксель зображення множиться на матрицю згортки (ядро) і результат підсумовується і надсилається у відповідну позицію оригінальне зображення.

Роботу згорткової нейронної мережі зазвичай інтерпретують як перехід від більш простих ознак зображення до більш абстрактних, підкреслюючи концепцію більш високого рівня. У цьому випадку мережа точно налаштовує і виробляє необхідну ієрархію абстрактних функцій (послідовності карт характеристик), відфільтровуючи дрібні деталі та виділяючи важливі.

Ця інтерпретація досить метафорична чи ілюстративна. Насправді функції, створювані складною мережею, незрозумілі та складні для інтерпретації, тому в практичних системах рекомендується не намагатися зрозуміти зміст цих функцій або намагатися налаштувати їх вручну, а покращити структуру та архітектуру системи. мережі для кращих результатів. У повністю взаємопов’язаній нейронній мережі кожен нейрон у кожному шарі з’єднаний з кожним нейроном попереднього шару, і кожному з’єднанню присвоюється особистий ваговий коефіцієнт. У згортковій нейронній мережі операція згортки використовує обмежену матрицю малої ваги, яка відображається в кожній позиції обробленого шару і генерує сигнал активації для нейрона наступного шару у відповідній позиції після кожного зсуву. Це означає, що одна і та ж матриця ваги використовується для різних нейронів вихідного шару, який також називають ядром сплетіння. Це інтерпретується як графічне кодування будь-якого символу, наприклад наявність косої риски під певним кутом. Потім наступний шар, отриманий в результаті операції згортання такої матриці ваг, показує наявність цієї ознаки в обробленому шарі та його координатах, створюючи т.зв. карта об’єктів. У згортковій нейронній мережі набір ваг — це не окремий елемент, а цілий набір елементів, які кодують зображення (наприклад, лінії та дуги під різними кутами). При цьому такі згорткові ядра не визначаються заздалегідь дослідником, а формуються самостійно, навчаючи мережу класичного методу зворотного поширення помилки. Перехід кожного набору ваг створює власну копію карти ознак, роблячи нейронну мережу багатоканальною. При перегляді шару з ваговою матрицею він зазвичай зміщується не на розмір цієї матриці, а на меншу відстань. Наприклад, коли розмірність матриці з вагами 4 × 4 зміщується на один або два пікселі замість 5. Приклад згорткової операції показано на рисунку 2.3. У цьому прикладі згортковий фільтр має розмір 3 × 3 [12].

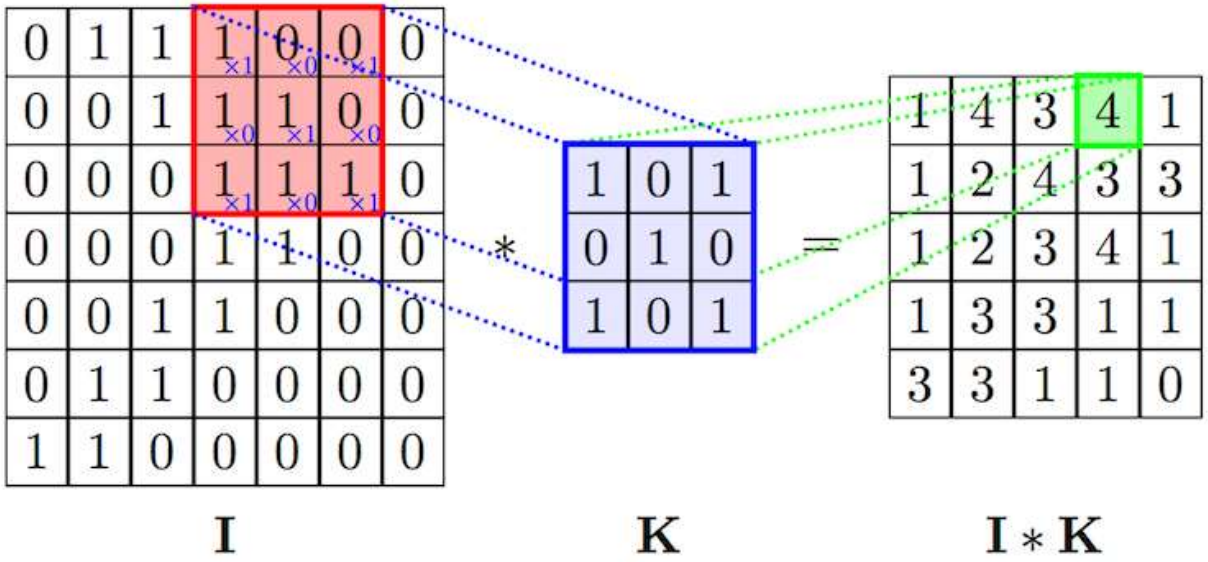

Рисунок 2.3 – Приклад Згорткової операції

Субдискретизація або пулінг зменшує розмірність об’єктів, отриманих із шару згортки карт. У цій архітектурі мережі вважається, що інформація про наявність потрібного об’єкта важливіша за точне знання його координат, тому з кількох сусідніх значень карти об’єктів вибирається одне – максимальне, і обробляється як одне із значень більш щільної карти ознак. Завдяки цій операції, крім прискорення подальших обчислень, мережа знаходить незмінність у масштабі вхідного зображення [2].

Розглянемо типову структуру згорткової нейронної мережі більш детально. Мережа складається з багатьох шарів. Після початкового шару (вхідного зображення) сигнал проходить через серію згорткових шарів, які чергуються між фактичною згорткою та підвибіркою (витягуванням). Чергування шарів дозволяє будувати більш складні функції, оскільки глибина мережі збільшується, карта зменшується з кожним наступним шаром, але збільшується кількість каналів. Найчастіше, після переміщення крізь кілька шарів, карта стає векторною або навіть скалярною, але таких карток об’єктів сотні. Як правило, на виході шарів згортки мережі встановлюється кілька додаткових шарів повністю пов’язаної між собою нейронної мережі, на вхід якої подаються кінцеві карти ознак.

Шар згортки є основним блоком згорткової нейронної мережі. Згортковий шар містить власний фільтр для кожного каналу, згорткове ядро, яке обробляє попередній шар за фрагментами, підсумовуючи результати матричного продукту для кожного фрагмента. У випадку згорткового ядра ваги спочатку невідомі і визначаються процесом навчання [2] Особливістю згорткового шару є відносно невелика кількість параметрів, які задаються під час навчання. Наприклад, якщо вихідне зображення має розмір 100×100 пікселів через три канали (тобто 30 000 вхідних нейронів), а згортковий шар використовує фільтри з ядром 3 × 3 пікселів з 6-канальним виводом, процес навчання визначає лише 9 вагових ядра. для всіх комбінацій каналів, тобто 3×3×3×6 = 162, в цьому випадку для цього шару потрібно знайти лише 162 параметри, що набагато менше, ніж кількість необхідних параметрів повнозв’язаної нейронної мережі близько 100 мільйонів.

Шар субдискретизації являє собою нелінійне ущільнення карти ознак, з групою пікселів (зазвичай 2 × 2), ущільненими до одного пікселя, які піддаються нелінійному перетворенню. Найбільш часто використовуваною функцією є максимум (рис. 2.4) [23]. На перетворення впливають непересічені прямокутники або квадрати, кожен з яких усічений на один піксель, з вибраним пікселем, який має максимальне значення. Операція пулінга дозволяє значно зменшити просторовий об’єм зображення. Пулінг інтерпретується так: якщо деякі особливості вже були виявлені в попередній операції згортання, таке детальне зображення більше не потрібно для подальшої обробки і ущільнюється до менш деталізованого. Крім того, фільтрація непотрібних деталей допомагає не перетренуватися. Пулінговий шар зазвичай розміщують після шару згортки перед наступним шаром згортки (рис. 2.4).

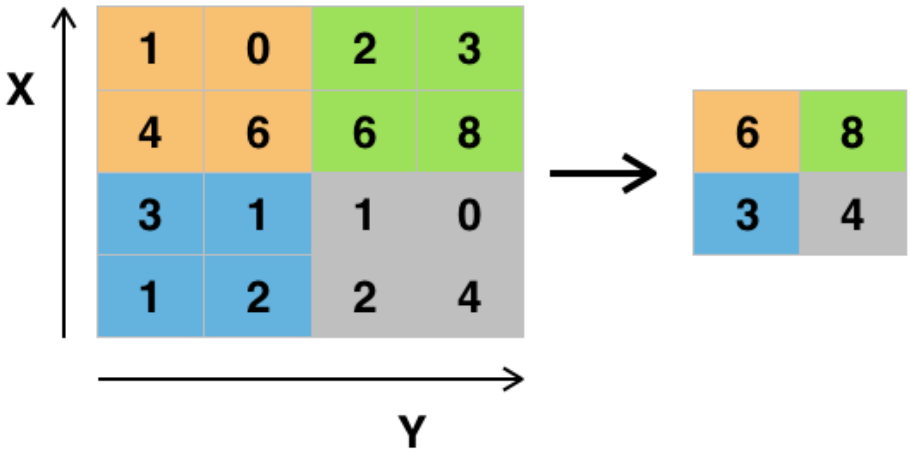

Рисунок 2.4 – Субдискретизація з кроком 2 і розміром вікна 2×2

Особливої уваги заслуговує дія т. зв хгортки транспонування або деконволюція. У певному сенсі ця операція протилежна операції згортання. Якщо ви використовуєте операцію згортки на крок, більший за одиницю, отримане зображення буде меншим за розміром, а якщо ви використовуєте операцію транспонованої згортки, то розмір результату буде більшим (рис. 2.5) [4].

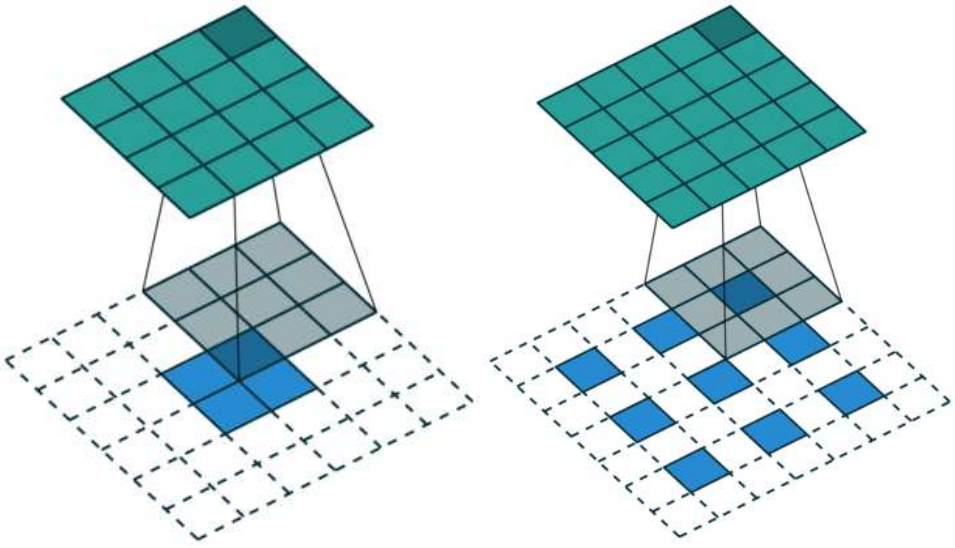

Рисунок 2.5 –Операція простої згортки й транспонованої згортки

На практиці операція транспонованої згортки часто використовується для відновлення початкового розміру зображення після кількох шарів підвибірки або звичайної згортки. Цей прийом особливо часто використовується в генеративних

моделях, де нейронна мережа повинна генерувати щось відмінне від вихідного зображення.

2.3 Організація навчання нейронної мережі для цифрової обробки фотографії

Правило навчання або алгоритм навчання — це метод або математична модель, яка підвищує продуктивність штучної нейронної мережі, і, як правило, правило застосовується знову і знову по всій мережі.

Відомі три парадигми навчання нейронних мереж, які базуються на характеристиках машинного навчання:

- supervised learning - навчання під керівництвом учителя;
- unsupervised - навчання без учителя;
- reinforcement leaming – навчання з підкріпленням.

У випадку згорткових нейронних мереж найкраще використовувати навчання з учителем, як було підтверджено в більшості досліджень, і альтернативи практично немає. Навчання з учителем – передбачається, що для кожного вхідного вектора є вихідний вектор. Разом ці два вектори називають навчальною парою, а набір навчальних пар називають навчальною вибіркою. Процес навчання зводиться до почергової відправки навчальних пар на вхід нейронної мережі, обчислення похибки між реальним і бажаним значенням нейронного сигналу та коригування параметрів мережі для зменшення цієї помилки.

Продуктивність нейронної мережі можна виміряти за допомогою функції втрат, яка оптимізується під час навчання мережі. Якщо функція втрат добре узгоджена, загалом, чим нижче її значення, тим краще мережа навчилася вирішувати свою проблему. Таким чином, завдання навчання нейронної мережі фактично є завданням мінімізації деякої функції втрат, пов’язаної з мережею, тобто завдання оптимізації. Тому кожен метод оптимізації можна використовувати з різним ступенем успіху [2].

Нейронні мережі були винайдені на початку 1900-х років, але почали використовуватися лише нещодавно. Причиною цього є винахід методу зворотного

поширення помилки [6]. Ця подія здійснила революцію в машинному навчанні, оскільки ефективність цього методу набагато вища, ніж у всіх методів, які були винайдені раніше. Саме тому цей метод був обраний для нашої нейронної мережі.

Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів - у напрямку, протилежному прямому поширенню сигналів у звичайному режимі. На кожній ітерації алгоритму поширення помилки обчислюється функція втрат і змінюються ваги нейронної мережі, щоб покращити рішення на отриманих прикладах.

Щоб визначити, як змінити вагу, також розраховується градієнт функції втрат для всіх параметрів мережі, тому для можливості застосування цього методу необхідно диференціювати функцію активації нейрона та знати її похідні. Метод є модифікацією класичного методу градієнтного спуску [2].

Мінімізація функції втрат є складною проблемою: є багато параметрів (для стандартних прикладів, реалізованих на ПК, їх кількість досягає мільйонів), місцевість адаптації (графік оцінки як функція від регульованих параметрів) є складною, він може містити багато локальних мінімумів.

Щоб зрозуміти алгоритм зворотного поширення помилки розглянемо ще раз будову нейронної мережі. Припустимо, що на вхід нейронної мережі надходить вектор x. Після застосування функцій активації і множення результату на матрицю ваг буде отримано вектор y, який надійде на вхід другого шару. Таким чином, можна сказати, що перший шар нейронної мережі задає деяке відображення

$$F1: F1(x) = y \tag{2.2}$$

Аналогічно виходом для другого шару буде вектор z, який ми отримаємо після перетворення вектора y. Введемо ще одне відображення

$$F2: F2(y) = F2\big(F1\ (x)\big) = z \tag{2.3}$$

Аналогічно можна побудувати n відображень

$$F_i, i = 1,2,\dots,n \tag{2.4}$$

де n – число шарів мережі

Звідси, на виході мережі отримаємо вектор:

$$F(x) = F_n\left(F_n - 1(\dots F1\ (x)\dots)\right) \tag{2.5}$$

Таким чином, нейронну мережу можна представити у вигляді композиції функцій і кожен новий шар додає в ній ще одну функцію.

Нехай θ – вектор параметрів нейронної мережі. При навчанні для обчислення градієнта функції втрат, потрібно обчислити градієнт функції F. Для обчислення градієнта $\nabla\theta F(x)$ потрібно скористатися правилом диференціювання складної функції:

$$\frac{dF}{dt} = \frac{dF_n}{dF_{n-1}} \times \frac{dF_{n-1}}{dF_{n-2}} \times \dots \times \frac{dF_1}{dt} \tag{2.6}$$

В тому випадку, коли похідні функцій активації є відомими, кожна з похідних може бути легко обчислена. При проході через мережу одного об'єкта, спершу будуть обчислені значення для останнього шару, потім – передостаннього і т.д. Це означає, що обчислення градієнта йтиме в напряку, зворотному порядку шарів. Тому метод зворотного поширення помилки і отримав таку назву [27].

До появи методів колоризації на основі нейроних мереж використовували багато інших підходів. Один з фундаментальних підходів, який надихнув багато інших розробок, – це підхід, в якому проблема колоризації ставиться як умовна задача оптимізації, за допомогою якої знаходять значення каналів a і b отриманого зображення, поки канал L відомий і використовує натомість чорно-біле вхідне зображення.

Кожен піксель має дві координати (x, y) . Таким чином, алгоритм розфарбовування на вхід отримує значення каналу $L = L(x, y)$, а на виході віддає $a(x, y)$, $b(x, y)$. Надалі (x, y) позначимо як r.

Алгоритм заснований на припущенні, що два сусідніх пікселя повинні мати схожий колір, якщо їх інтенсивність також схожа. Формально можна сказати, що для кожного пікселя необхідно мінімізувати різницю між каналами $a(r)$ і $b(r)$ і зваженими середніми кольорів його сусідніх пікселів. Опишемо цю різницю на прикладі каналу a:

$$J(a) = \sum_{r} \left(a(r) - \sum_{s \in N(r)} w_{rs} a(s) \right)^2 \tag{2.7}$$

де w_{rs} – функція ваги, сума значень якої дорівнює одиниці, і яка приймає великі значення, якщо $L(r)$ близько до $L(s)$, і менші – в іншому випадку;

$N(r)$ – множина сусідів пікселі з координатою r.

Це й же спосіб використаємо в даній магістерській кваліфікаційній роботі.

2.4 Вибір способу оптимізації навчання нейронної мережі для цифрової обробки фотографії

Навчання нейронних мереж вимагає обчислення градієнта функції втрат. Найчастіше функція втрат містить в собі суму втрат для всіх об'єктів навчальної вибірки. У загальному випадку навчальна вибірка може мати дуже великий розмір, в силу чого обчислення точного значення градієнта є складним і нестабільним завданням [26]. Одним із способів адаптувати градієнтний спуск для оптимізації функцій такого роду є стохастичний градієнтний спуск.

Суть методу полягає в припущенні, що для оптимізації функцій цього типу не потрібно обчислювати точний градієнт при кожній ітерації, достатньо взяти його стохастичну оцінку, а потім зробити градієнтний крок у своєму напрямку. Оцінка дуже проста - градієнт обчислюється не за всією вибіркою, а лише за її підмножиною.

Коли розміром підмножини обирається вся вибірка, в результаті отримаємо класичний градієнтний спуск.

Перевага методу стохастичного градієнтного спуску полягає в тому, що у багатьох випадках швидкість його збіжності значно вище, ніж у класичного градієнтного спуску. Ще одна перевага полягає в тому, що цей спосіб полегшує вихід з локальних мінімумів, ніж у випадку класичного підходу. На рисунку 2.6 показані приклади стохастичного та звичайного градієнтного спуску. Видно, що градієнтний спуск робить менше кроків і рухається більш точно в напрямку до оптимального, і робить це з меншою кількістю ітерацій, але кожен з ітерацій стохастичного градієнтного спуску вимагає значно менших обчислень, тому швидше сходиться.

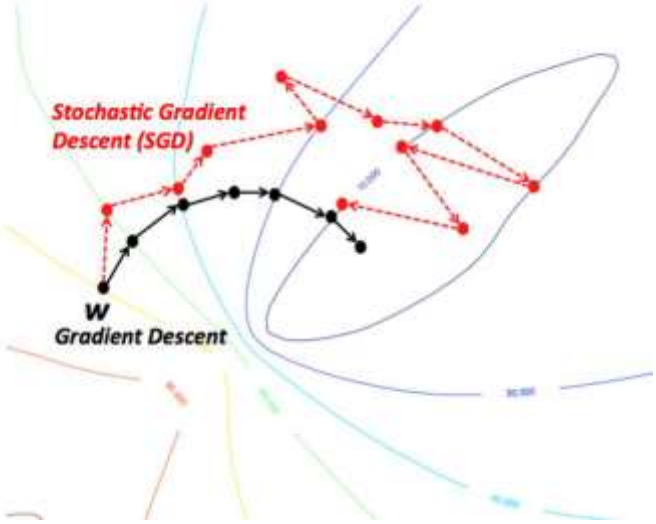

Рисунок 2.6 – Приклад стохастичного і звичайного гардієнтного спуску

Однією з модифікацій методу стохастичного градієнтного спуску є метод ADAM [6]. Хоча ця модифікація вносить багато різних змін, найважливішим можна назвати використання так званих імпульсів – на кожній ітерації до отриманої оцінки градієнта додається зважене середнє деякого числа оцінок з попередніх ітерацій. Така модифікація дозволяє методу рухатися в напрямку до оптимального, піддаючись коливанням менше, ніж їм піддається метод стохастичного градієнтного спуску. ADAM добре зарекомендував себе на практиці і використовується в більшості сучасних нейронних мереж. З цієї причини він буде використаний у цій роботі.

Перенавчання – явище в машинному навчанні, коли модель добре поводиться на прикладах з навчального зразка, але не працює на тесті. Причиною цього явища є те, що навчання моделей зазвичай є завданням оптимізації функції втрат у навчальному зразку, тоді як справжнє завдання – мінімізувати її у всіх можливих прикладах. Через це при занадто сильній оптимізації модель по суті "підганяє" свої параметри під приклади з тренувальної вибірки, втрачаючи генералізацію і узагальнення на весь простір прикладів [9].

На рисунку 2.7 червоною лінією показаний графік помилки на тестовій вибірці, а синьою – на тренувальній. Видно, що в якийсь момент похибка в тестовому зразку перестає зменшуватися і починає зростати. Це один типовий приклад перенавчання, коли модель починає адаптуватися до навчального зразка замість узагальнення.

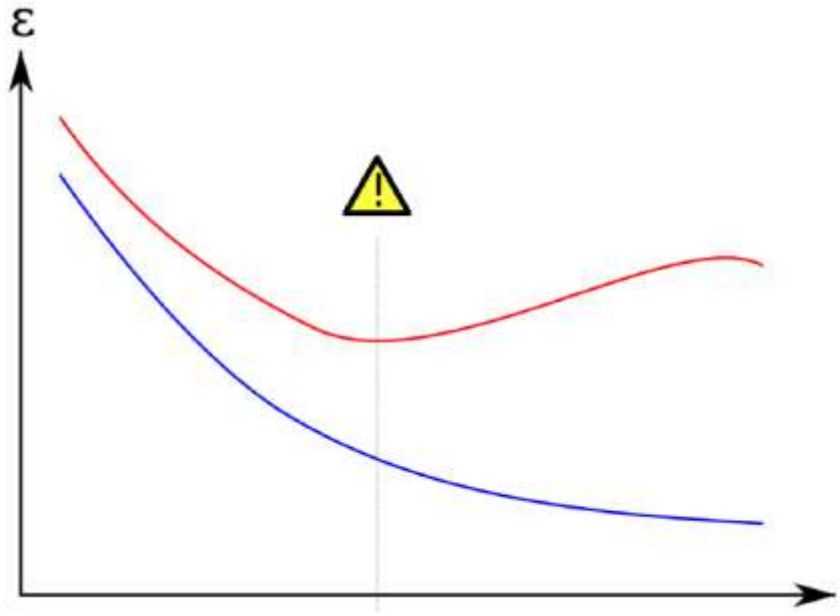

Рисунок 2.7 – Приклад перенавчання нейронної мережі

Боротьба з перенавчанням – одна з актуальних задач в області машинного навчання і розроблено безліч способів її рішення. Одним з них є регуляризація.

Регуляризація – метод додавання деякої додаткової інформації до умови з метою вирішити некоректно поставлену задачу або запобігти перенавчання [9]. На практиці часто реалізується в такий спосіб. Замість вирішення завдання

$$L \to min \tag{2.8}$$

Вирішується задача

$$L + \lambda * ||w|| \to min \tag{2.9}$$

де L – функція втрат;

λ – коефіцієнт регуляризації;

w – вектор параметрів моделі.

Як норми, зазвичай, використовується L_1 або L_2 норми. Ці способи називаються L_1 і L_2 регуляризації відповідно. Однією з причин цього методу є те, що норма вектора параметрів корелює зі складністю моделі, а якщо модель занадто складна, то вона може адаптуватися до навчального зразка, а не узагальнювати по всьому простору.

На рисунку 2.8 показаний приклад такої ситуації. Розглянуто модельну задачу поділу точок на площині на два класи, у цьому прикладі - на червоний та синій. Хоча зелена лінія є оптимальним рішенням у навчальному зразку, її форма дуже складна і, швидше за все, чорна лінія на тестовому зразку дасть найкращий результат.

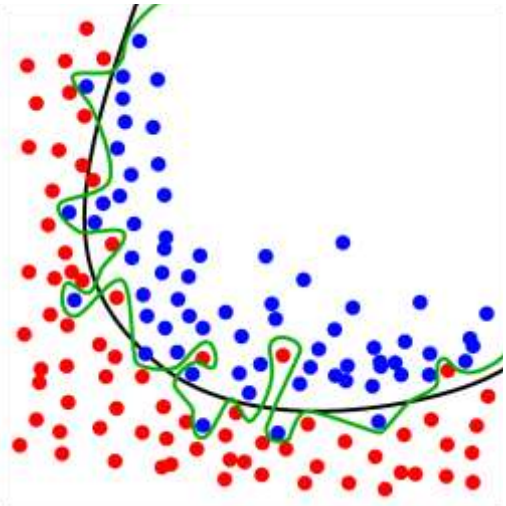

Рисунок 2.8 – приклад навчання на надто складній моделі

Проте в згорткових нейронних мережах доцільніше використовувати метод виключень для регуляризації.

Припустимо, у нас є нескінченна обчислювальна потужність. Тоді чудовим методом регуляризації було б навчання всіх підмножин побудованої мережі та взяття середнього зваженого рівня їх результатів. Такий підхід у машинному навчанні називається ансамблем [30].

Це можливо безпосередньо для дуже маленьких мереж, але, очевидно, неприйнятний для великих.

Виключення (dropout) – метод, який ефективно виконує описані вище операції, дозволяє уникнути перенавчання і певним чином поєднує експоненціально багато різних мереж. Виключаючи мережевий нейрон, ми тимчасово видаляємо його з мережі разом з усіма вхідними та вихідними з'єднаннями. Вибір нейрона який виключають відбувається випадковим чином. Кожна отримана таким чином мережа тренується. В результаті тренування мережі з виключеннями в деякому роді є тренування 2^n мереж, де n – число нейронів в ній. Приклад мережі до і після виключення нейронів показаний на рисунку 2.9.

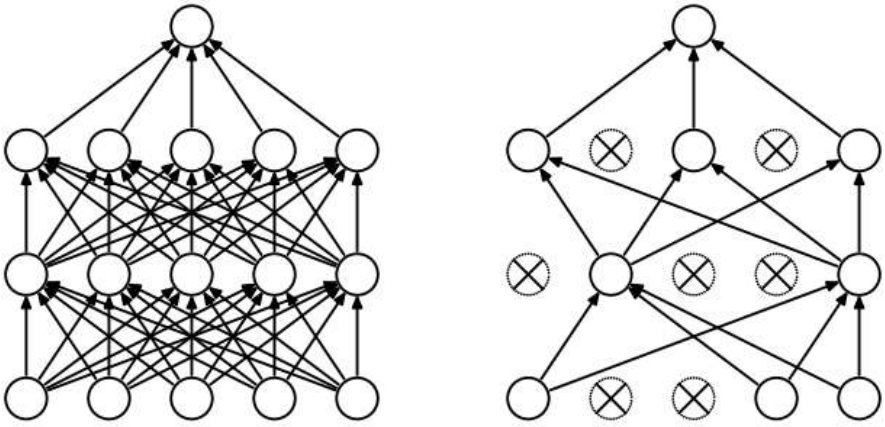

Рисунок 2.9 – Приклад методу виключення

Використовуючи таку мережу, неможливо обчислити всі значення експоненціально. Натомість використовується більш проста схема. Жоден нейрон не виключається. Всі ваги в мережі нормалізуються до ймовірності, з якою вони були виключені під час тренувань. Таким чином, вихід кожного нейрона є математичним

очікуванням його виходу, як якщо б її застосували 2^n раз для всіх підмножин. Нормування ваг мережі показана на рисунку 2.10.

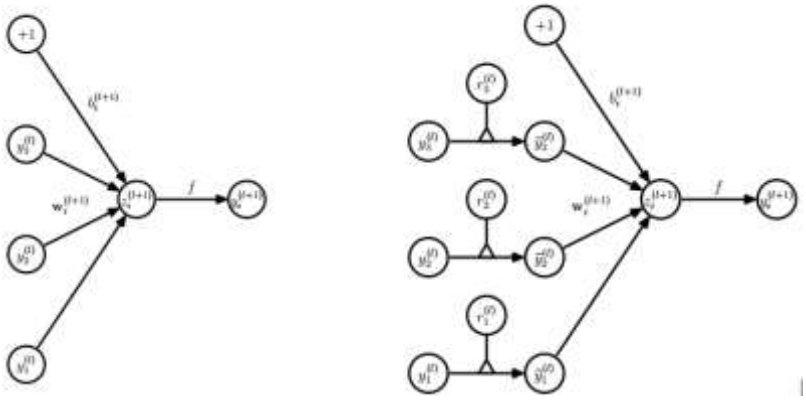

Рисунок 2.10 – Приклад ваг нейронної мережі при застосуванні методу виключень

Мережі такого типу навчаються точно так само, як і звичайні. Єдиною відмінністю, наприклад, для стохастичного градієнтного спуску, є те, що в процесі навчання кожен батч проходить не через всю мережу, а лише через її підмножину.

Можна також показати, що використання методики виключення є своєрідним доповненням шуму до навчальних даних у навчальному процесі, що є одним із прийомів підвищення стабільності моделі.

Показано, що такі нейронні мережі набагато стійкіші та менш схильні до перекваліфікації, ніж звичайні нейронні мережі. На даний момент важко натрапити на серйозні роботи або промислові мережі, які не використовують цю техніку. Тому наша робота використовуватиме цю методику.

Навчання глибоких нейронних мереж сильно ускладнюється тим, що розподіл входів кожного шару значно змінюється під час навчання, оскільки параметри попереднього шару так само змінюються. Це значно знижує швидкість навчання, вимагаючи ретельної ініціалізації параметрів і зменшення параметра швидкості навчання. Це явище називається внутрішнім коваріаційним зсувом. Батчева нормалізація є частиною мережі, що дозволяє використовувати більш високі швидкості навчання і не звертати уваги на початкові налаштування мережі,

застосовуючи нормалізацію до даних у кожній групі. Показано також, що такий підхід є додатковою регуляризацією [7].

Зазвичай нейронні мережі навчаються методом стохастичного градієнтного спуску або його модифікаціями. Метод полягає в тому, що на кожній ітерації градієнт рахується не для всіх об'єктів навчальної вибірки, а лише для певної її підмножини – батчу. Градієнт уздовж такого батчу дає оцінку на справжній градієнт, дозволяючи з набагато меншими обчислювальними затратами виробляти черговий крок спуску.

Проблема полягає в тому, що після оновлення мережевих налаштувань після проходження наступних батчів, дані, що надходять на вхід кожного шару, можуть різко змінитися. Алгоритм навчання повинен робити багато непотрібних дій, щоб змусити мережу щоразу адаптуватися до таких змін. Проблема стає більш значущою, чим більша глибина мережі.

Нормалізація може бути проведена дуже просто. Нехай кожен об'єкт батчу є вектором з d параметрами, $X = \left(x\,(1), \ldots, x\,(d)\right)$. Усередині батчу порахуємо для кожного параметра середнє і дисперсію по всіх об'єктах [30]. Після чого просто припустимо, що

$$\hat{x}^{(k)} = \frac{x^{(k)} - E\left[x^{(k)}\right]}{\sqrt{D\left[x^{(k)}\right]}} \tag{2.10}$$

де математичне очікування для кожного параметра і його дисперсія просто пораховані статистично по тренувальній вибірці.

На виході такого шару ми завжди маємо дані з нульовим математичним очікуванням і одиничною дисперсією. Навіть цього досить, щоб вирішити багато описаних вище проблем, але не застосовується з деякими функціями активації. Щоб позбутися і від цієї проблеми, застосовується ще одна нескладна техніка. Виходом шару буде не число $x^{(k)}$, а

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{2.11}$$

де γ і β – ті параметри мережі, які піддаються навчанню. Таким чином, ми дозволяємо мережі самостійно в процесі навчання визначити, з даними якої дисперсії і математичного очікування відповідний шар може працювати ефективніше.

Неважко побачити, що при

$$y^{(k)} = \sqrt{D[x^{(k)}]}, \tag{2.12}$$

$$\beta^{(k)} = E\left[x^{(k)}\right] \tag{2.13}$$

виходом мережі буде оригінальне значення параметра $x^{(k)}$.

Ще одним покращенням є те, що у випадку зі стохастичними методами навчання (в загальному – стохастичний градієнтний спуск), математичне сподівання і дисперсія обчислюються не для всієї вибірки, а лише для поточного батча. Це також призводить до того, що в такому випадку компоненти градієнта, що відповідають параметрам β і γ, природним чином обчислюються за допомогою алгоритму зворотного поширення помилки [30].

В результаті якість навчання істотно підвищується. Було показано, що наприклад, для мережі Inception, що є переможцем змагання за класифікацією зображень ImageNet 2014 року, вдалося отримати якість оригінальної мережі за в 14 разів менше число ітерацій. На рисунку 2.11 зображено приклад графу обчислення операцій батчевої нормалізації.

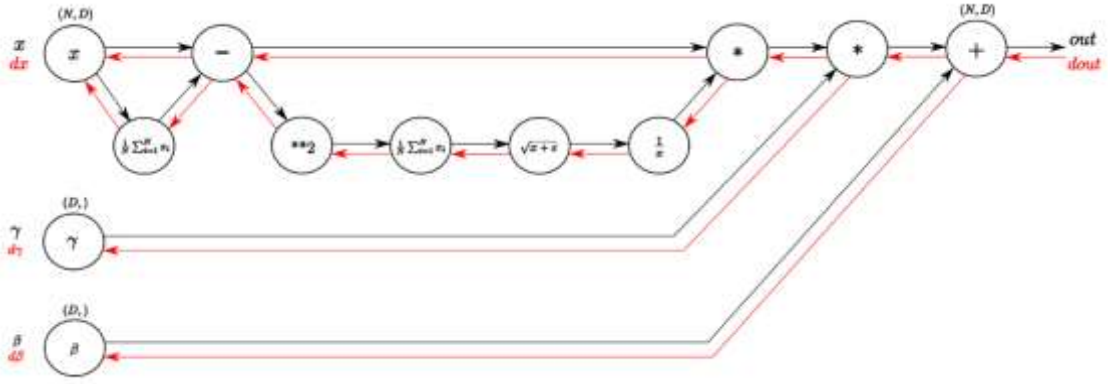

Рисунок 2.11 – Приклад графу обчислення операцій батчевої нормалізації

2.5 Обгрунтування вибору колірної моделі

Колірна модель - математична модель, яка описує представлення кольорів у вигляді наборів чисел (зазвичай трьох, рідше чотирьох значень), званих компонентами кольору або колірними координатами. Усі можливі значення кольору, задані моделлю, визначають колірний простір [3].

Спосіб налаштування кольору істотно впливає на вирішення проблеми. Наприклад, стандартна модель RGB набагато менш підходить для більшості застосувань, ніж деякі менш популярні моделі.

RGB (червоний, зелений, синій) - адитивна колірна модель. Вибір кольорів залежить від особливостей будови і сприйняття оком людини. Він є адитивним, оскільки встановлює кольори, додаючи їх до чорного (рис. 2.12).

Рисунок 2.12 – Адитивність колірної моделі RGB

Колір можна представити у вигляді трьох чисел, кожне від 0 до 1, тому весь колірний простір можна представити у вигляді куба 1 * 1 * 1 (рис. 2.13).

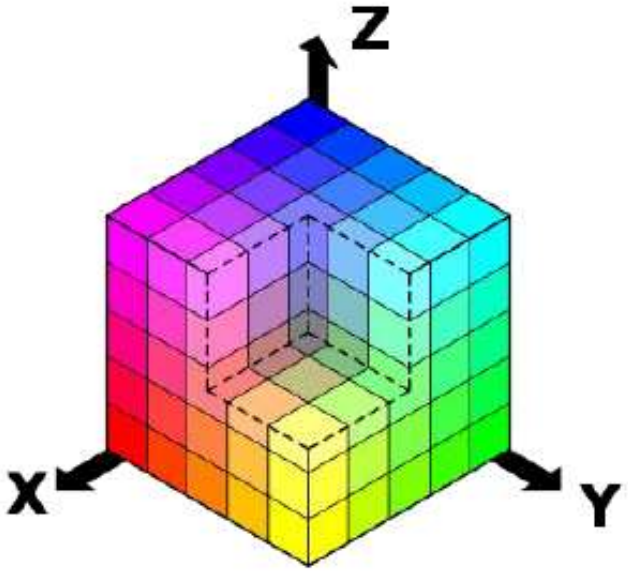

Рисунок 2.13 – Модель RGB у вигляді одиничного куба

Недоліки RGB-моделі:

1. Залежність обладнання. Колір, який утворюється в результаті змішування кольорових компонентів елемента RGB, залежить від типу люмінофора.

2. Обмеження асортименту. Колірна гама – це діапазон кольорів, які можна розрізнити або відтворити на пристрої, незалежно від колірної гами.

HSV (Hue, Saturation, Value) - колірна модель, де координати кольору:

- Tint - відтінок кольору. Варіюється від 0 до 360 градусів, іноді може бути представлено як від 0 до 1 або до 100;
- Насичення – насичення. Він коливається від 0 до 100 або від 0 до 1. Чим вище цей параметр, тим «чистіший» колір, тому його іноді називають чистотою кольору. І чим ближче цей параметр до нуля, тим ближче колір до нейтрального сірого;
- Значення або яскравість. Він також встановлюється від 0 до 100 або від 0 до 1 [3].

Модель HSV представлена в циліндричній системі координат або у вигляді конуса (рис. 2.14).

Недолік колірної моделі HSV:

- схожа на попередню колірну модель;

- обмежений колірний простір.

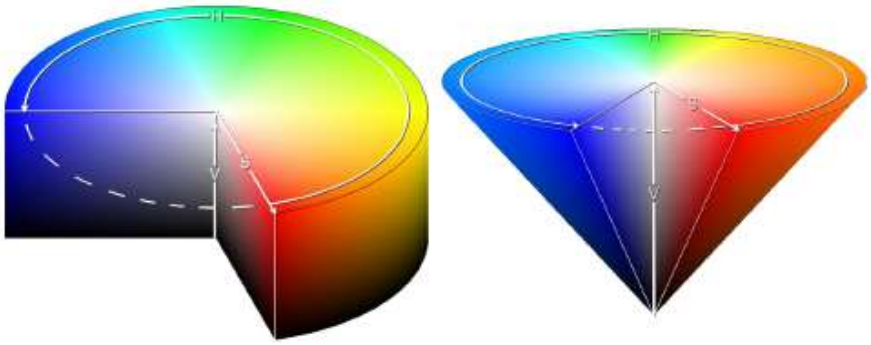

Рисунок 2.14 – Колірна модель HSV

Колірна модель CIELAB (лабораторія) була розроблена для створення світлового простору, де зміна кольору буде лінійною для людського сприйняття. Тобто, внесення однакової зміни значень колірних координат у різних частинах колірного простору створює враження однакової зміни кольору [34]. Використання такої моделі покращило б нелінійність колірного сприйняття людиною. Канал L (яскравість, яскравість) має те саме значення, що й канал Y в моделі YUV, координата a відповідає позиції кольору від зеленого до червоного, а координата b - від синього до жовтого (рис. 2.15).

На відміну від багатьох інших моделей, де кольоровий дисплей залежить від апаратних даних, CIELAB однозначно визначає колір. Тому CIELAB широко використовується в програмному забезпеченні обробки зображень.

Колірна модель CIE Lab, порівняно з RGB, дозволяє точніше визначити колір. Таким чином, CIE Lab є абсолютною моделлю, а RGB — відносною моделлю. Щоразу, коли зображення перетворюється з однієї колірної моделі в іншу, спочатку виконується внутрішнє перетворення в модель CIE Lab. Великою перевагою моделі CIE Lab є її незалежність від пристрою та той факт, що її колірний діапазон є найбільшим (тому вона містить широкий діапазон кольорів), а також повна незалежність яскравості L від кольору a, b. компонентів.

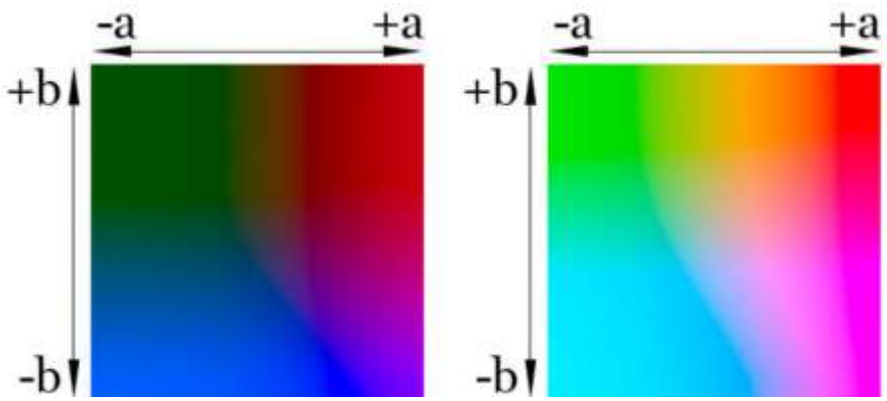

Рисунок 2.15 − Залежність кольорової моделі CIELAB від значень a і b, при L рівному 0.25 і 0.75

Переваги моделі CIELAB [3]:

1. Апаратна незалежність;

2. Більш широкий діапазон кольорів у порівнянні з моделями RGV та HSV;

3. На основі параметрів цієї колірної системи можна визначити параметри інших колірних моделей.

Завдяки наведеним вище функціям ми будемо використовувати цю модель, щоб додати кольорові підказки до чорно-білого зображення.

2.6 Розробка структури нейронної мережі для розфарбовування цифрових фотографій

Класична архітектура розфарбовування мережі являє собою структуру encoder-decoder - кілька шарів згортки підряд, за якими слідують відповідні транспоновані шари згортки [3]. Ця архітектура дозволяє мережі розпізнавати складні шаблони аж до найвужчої частини. Ця частина мережі називається «encoder».

Інша частина мережі робить навпаки - вона генерує необхідну вихідну потужність, у нашому випадку - кольоровий малюнок. Цю частину мережі називають «decoder».

Але коли справа доходить до задачі розфарбовування, цього недостатньо. У такій архітектурі втрачається багато низькорівневої інформації, наприклад, про межі або об’єкти більш складної форми. Замість цього було запропоновано новий підхід

до побудови такої архітектури, який називається U-net. Його особливість полягає в тому, що шари декодера отримують дані не тільки з попереднього шару, але і з відповідного шару кодера (рис. 2.16).

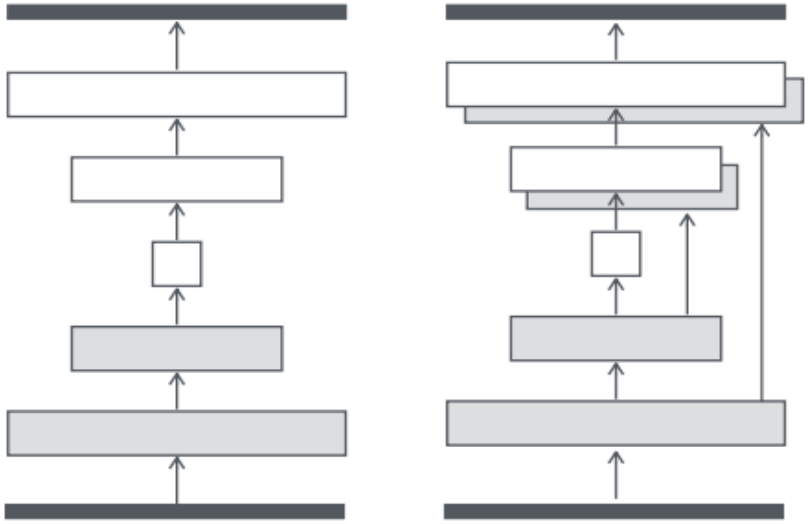

Рисунок 2.16 – Структура мережі encoder-decoder та U-net

Також, на якість навчання суттєво впливає використання шарів нормалізації. Тут їх використовують після кожного шару згортки.

Варто згадати про регуляризації L_1 і L_2. Вище було сказано, що їх використання недоцільно, а все тому, що їх використання призводить до генерації розмитих зображень з розмитими краями і плавною зміною кольору. З цієї причини регуляризація приймає лише форму партійної нормалізації та винятків.

Перевагою продуктивності є також створення зображень не в просторі RGB, а в лабораторному просторі, оскільки L-канал по суті є чорно-білим зображенням. У цьому випадку моделям потрібно генерувати лише два канали замість трьох. Також розглядалося використання простору HSV з використанням чорно-білих значень зображення як V-каналу, але було досліджено, що це матиме негативний вплив на якість навчання моделей. При S = 0 значення каналу H не впливає на отримане зображення.

Недоліком більшості моделей, які автоматично розфарбовують зображення, є неможливість користувача певним чином вплинути на результат, навіть при наявності

апріорного знання кольорів у певних частинах зображення. В результаті колір, наприклад, одягу часто відрізняється від оригіналу. Так само, з цим знанням, багато різних кольорів можуть здаватися такими ж правдоподібними. Тому в кваліфікаційній роботі цього майстра буде застосовуватися метод розмальовки шляхом розповсюдження нестандартних підказок.

Архітектура згорткової нейронної мережі U-net, розроблена в рамках магістерської кваліфікаційної роботи, модифікована підмережею підказок користувача (рис. 2.17).

Вхідними параметрами є чорно-біле зображення (L-канал у кольоровій моделі CIE Lab), маска з одиниць і нулів, де одиниці представляють пікселі, де розташована визначена користувачем підказка, та два канали a і b - самі підказки.

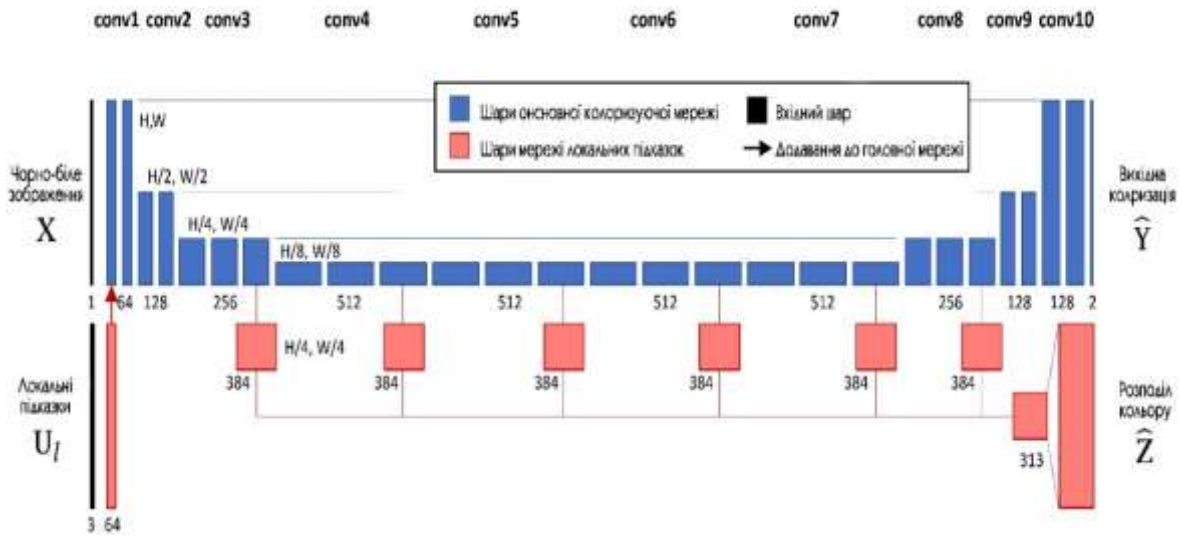

Рисунок 2.17 – Структура нейронної мережі розфарбовування цифрових фотографій

На відміну від класичних методів, поряд з каналами a та b, мережа також генерує розподіл кольору для кожного пікселя, який потім використовується в інтерфейсі.

Нейронна мережа складається з двох частин: основної мережі розфарбовування за архітектурою U-net (10 згорткових блоків) і мережі локальних підказок.

2.7 Розробка алгоритму функціонування інформаційної технології цифрової обробки фотографії

Розробимо алгоритм функціонування інформаційної технології цифрової обробки фотографії (рис. 2.18).

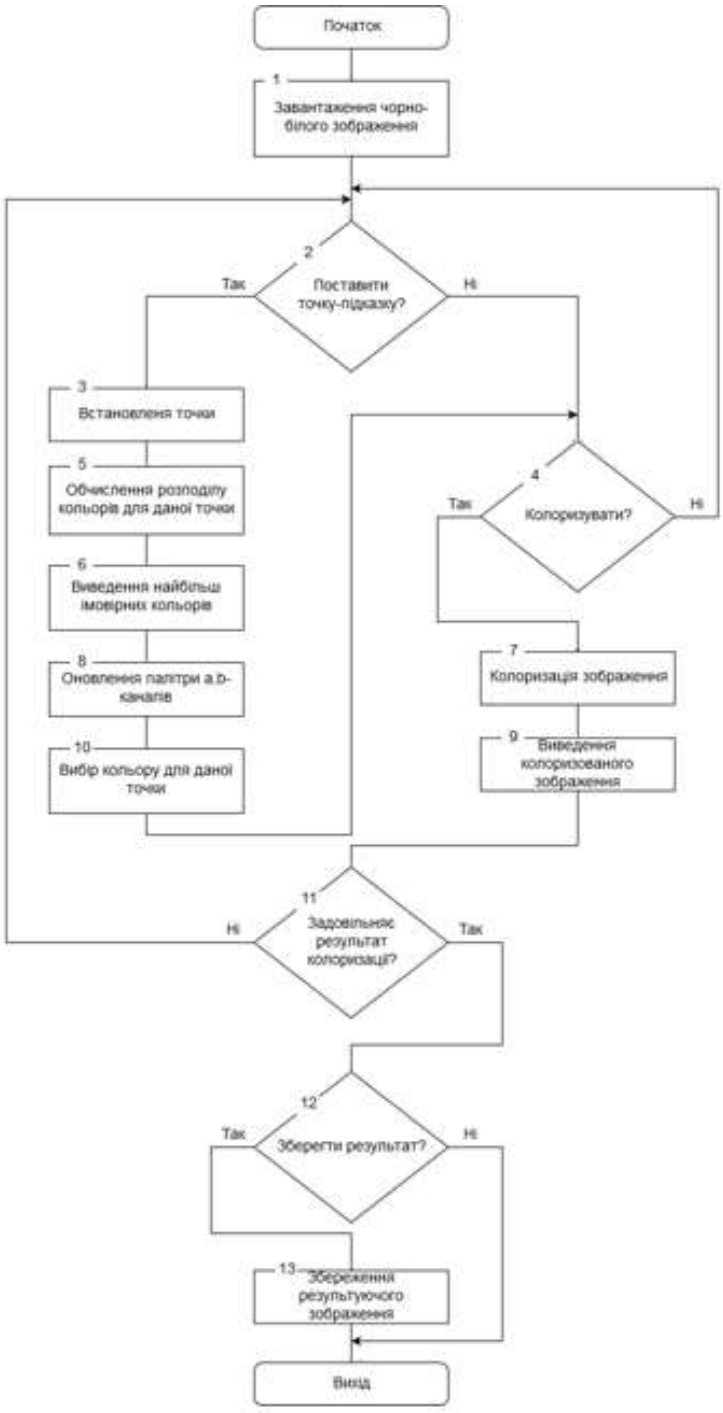

Рисунок 2.18 – Алгоритм функціонування інформаційної технології цифрової обробки фотографії

Для сегментації зображення для забезпечення можливості малювання за номерам поперше, необхідно зменшити кількість кольорів, а по-друге в отриманому зображенні виділити контури областей, що сформувалися в процесі обробки.

Серед особливостей запропонованого алгоритму слід відмітити поєднання в алгоритмі попередньої обробки, фільтрації, перетворень та правил взаємодії між характерними точками на різних етапах роботи

Цей алгоритм можна застосовувати разом із такими алгоритмами як фільтрація шумів та розмиття зображення [8]. Ці алгоритми можна використовувати як до зменшення кількості кольорів так і після цієї процедури. В результаті використання цих алгоритмів вихідне зображення не буде мати дуже маленьких областей.

2.8 Висновок до розділу 2

У другому розділі обгрунтовано вибір колірної моделі для розфарбовування зображень. Обгрунтовано вибір згорткової нейронної мережі, як один з найефективніших для вирішення задачі розфарбовування. Обгрунтовано вибір методу навчання нейронної мережі.

Розроблено архітектуру нейронної мережі для розфарбовування цифрових фотографій, що грунтується на архітектурі U-net.

Розроблено загальну схему алгоритму функціонування інформаційної технології цифрової обробки фотографії.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЦИФРОВОЇ ОБРОБКИ ФОТОГРАФІЇ

3.1 Обґрунтування вибору мови програмування та середовища розробки

При виборі засобів розробки програмного забезпечення необхідно враховувати надзвичайно велику кількість різних аспектів, найважливішим з яких є мова програмування, оскільки вона значною мірою визначає інші доступні інструменти. Наприклад, для розробки графічного інтерфейсу користувача розробникам потрібна сумісна з мовою бібліотека графічного інтерфейсу, яка надає готові елементи інтерфейсу, такі як кнопки та меню. При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність програми та продуктивність пам’яті. Важливим фактором при виборі мови, в даному випадку, є її універсальність.

Для створення програмних систем такого рівня використовуються три мови, які відповідають вимогам універсальності: Java, C# і Python.

Java — це об'єктно-орієнтована мова програмування, випущена Sun Microsystems у 1995 році як основний компонент платформи Java. Oracle, яка придбала Sun Microsystems у 2009 році, зараз працює над мовою. Синтаксис мови значною мірою походить від C і C ++. В офіційній реалізації програми Java компілюються в байт-код, який при інтерпретації інтерпретується віртуальною машиною для певної платформи. Oracle надає компілятор Java та віртуальну машину Java, які відповідають специфікаціям Java Community Process під загальною суспільною ліцензією GNU.

C# — це об'єктно-орієнтована мова програмування із безпечною системою запису для .NET. Розроблено Андерсом Галесбергом, Скоттом Вілтамутом і Пітером Голдом під егідою Microsoft Research (під керівництвом Microsoft).

Синтаксис C# схожий на C++ і Java.Мова має суворо статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени

класу, атрибути, події, властивості, винятки, коментарі у форматі XML. Перенявши багато чого від своїх попередників - C ++, Delphi, Module і Smalltalk - C #, на основі їхньої практики їх застосування, виключає деякі моделі, які виявилися проблематичними при розробці програмного забезпечення, наприклад, множинне наслідування класів (на відміну від C). ++).

Python — це інтерпретована високорівнева об’єктно-орієнтована мова програмування з динамічною семантикою. Розроблено в 1990 році Гвідо ван Россумом. Високорівневі структури даних, поряд з динамічною семантикою та динамічним зв’язуванням, роблять його привабливим для швидкої розробки програмного забезпечення, а також засобом зв’язування існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python і стандартні бібліотеки доступні як у скомпільованому, так і в вихідному вигляді на всіх основних платформах. Мова програмування Python підтримує декілька парадигм програмування, включаючи об’єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану [37].

Основним недоліком C# в порівнянні з Java і Python є його власна ліцензія і можливість повноцінно використовувати всі його функції тільки в Windows. Оскільки розробка системи бажана, повна підтримка UNIX-подібних операційних систем, включаючи GNU / Linux. Тому справжній вибір – між Java та Python.Швидкість розробки програмного забезпечення Python на порядок вища, ніж у Java, завдяки динамічному запису. Крім того, Python — це інтерпретована мова, яка дозволяє легко вносити зміни та налаштовувати програмну систему на етапі впровадження.

Однак головною перевагою при виборі Python було те, що для цієї мови програмування існує більшість бібліотек нейронних мереж, таких як: Theano, Pylearn2, Deepnet, Caffe, Keras.

Тому основною мовою програмування є Python.

Інтегроване середовище розробки (IDE) — це програмна система, яка використовується програмістами для створення програмного забезпечення [40].

Як правило, середовище розробки включає:

• текстовий редактор,

• компілятор або інтерпретатор,

• заходи автоматизації стягнення боргів,

• налагоджувач.

Іноді він також включає інструменти для інтеграції з системами контролю версій і різні інструменти для спрощення дизайну графічного інтерфейсу. Багато сучасних середовищ розробки також включають засіб перегляду класів, інспектор об'єктів та діаграму ієрархії класів для використання в об'єктно-орієнтованому програмуванні. Хоча існують інтернет-провайдери розроблений для багатьох мов програмування, таких як Eclipse, Embarcadero RAD Studio, Qt Creator, останні версії NetBeans, Xcode і Microsoft Visual Studio, але ISR зазвичай призначений для однієї конкретної мови програмування, наприклад Visual Basic, Delphi, Dev- C ++.

Інтегровані середовища розробки були створені для максимальної продуктивності розробників завдяки тісно пов’язаним компонентам із простими користувацькими інтерфейсами. Це дозволяє розробнику робити менше кроків для перемикання між різними режимами, на відміну від окремих програм розробника.

RRI зазвичай є єдиною програмою, де вся розробка відбулася. Зазвичай він містить багато функцій для створення, модифікації, компіляції, розгортання та налагодження програмного забезпечення. Метою середовища розробки є вилучення конфігурації, необхідної для об’єднання інструментів командного рядка в єдиний модуль, що зменшить час вивчення мови та підвищить продуктивність розробника. Також вважається, що комплексна інтеграція цілей розвитку може ще більше підвищити продуктивність. Наприклад, ISR дозволяє аналізувати код і таким чином надавати негайний зворотний зв’язок і повідомляти про синтаксичні помилки.

На даний момент існує багато постачальників веб-послуг для Python: Boa Constructor, Eclipse, PyDev.

3.2 Обгрунтування вибору нейромережевої бібліотеки Caffe

Як згадувалося раніше, Python є найбагатшою на програмування нейронних бібліотек. Бібліотека Caffe була обрана для впровадження інформаційних технологій розфарбовування чорно-білих зображень.

Caffe розробляється з вересня 2013 року. Янцин Цзя почав свій розвиток під час навчання в Каліфорнійському університеті Берклі. Відтоді Caffe активно підтримується The Berkeley Vision and Learning Center (BVLC) і спільнотою розробників GitHub. Бібліотека поширюється за ліцензією BSD 2-Clause.

Caffe реалізовано на мові програмування C++, є обгортки в Python і MATLAB. Офіційно підтримувані операційні системи - Linux і OS X, є також неофіційний порт для Windows. Для векторних і матричних обчислень Caffe використовує бібліотеку BLAS (ATLAS, Intel MKL, OpenBLAS). Крім того, до числа зовнішніх залежностей входять glog, gflags, OpenCV, protoBuf, boost, leveldb, nappy, hdf5, lmdb. Щоб прискорити обробку, Caffe може працювати на графічному процесорі з використанням основних можливостей CUDA або примітивної бібліотеки глибокого навчання cuDNN [8].

Розробники Caffe підтримують можливість створювати, навчати та тестувати повні та складні нейронні мережі.

На завершення хотілося б ще раз підкреслити, що Caffe має ряд переваг перед аналогами, а саме:

- «чиста» архітектура дозволяє миттєво реалізувати. Мережі визначаються за допомогою простих конфігураційних файлів, без жорсткого «вшивання» параметрів даних у код. Це робить перемикання між ЦП і ГП простим і швидким, дозволяючи навчати моделі на ПК з потужним графічним процесором, а потім використовувати його на будь-якій кластерній машині.

- відкритий код дозволяє не тільки контролювати впровадження, але й модифікувати його відповідно до ваших потреб. За перші 6 місяців активного використання Caffe понад 300 незалежних розробників зробили внесок у розвиток бібліотеки, просто налаштувавши її для себе.

- Швидкість роботи робить Caffe незамінним інструментом для комерційного використання. Використовуючи лише один графічний процесор NVIDIA K40, бібліотека може обробляти понад 400 мільйонів зображень на день. Це еквівалентно швидкості навчання 5 мілісекунд на зображення та швидкості тестування 2

мілісекунд на зображення. Автори справедливо підкреслюють, що Caffe є лідером швидкості серед згорткових нейронних мереж [9].

Caffe – бібліотека, сконцентрована ефективної реалізації алгоритмів глибокого навчання, має відкритий вихідний код. До переваг бібліотеки належить наявність великої кількості передвибраних моделей та прикладів, що у поєднанні з іншими характеристиками робить бібліотеку найпростішою для старту роботи серед перерахованих вище. Бібліотека Caffe надає досить простий та зручний інтерфейс, дозволяючи легко конфігурувати та навчати нейронні мережі. Для роботи з бібліотекою потрібно створити опис мережі у форматі prototxt (англ., protocol buffer definition file, мова опису даних, створена компанією Google), яка дещо схожа на формат JSON, добре структурована і зрозуміла для людини. Опис мережі є почергове опис кожного з її шарів. Як вхідні дані бібліотека може працювати з базою даних (leveldb або lmdb), даними з пам'яті, файлами HDF5 і зображеннями. Також є можливість використовувати для цілей розробки та тестування спеціальний вид даних, званий DummyData [1].

Deeplearning4j – бібліотека з відкритим вихідним кодом для реалізації нейронних мереж та алгоритмів глибокого навчання, написана мовою Java. Можливе використання мов Java, Scala та Closure, підтримується інтеграція з Hadoop, Apache Spark, Akka та AWS. Бібліотека розвивається та підтримується компанією Skymind, яка також надає комерційну підтримку для цієї бібліотеки. У бібліотеці використовується підбібліотека для швидкої роботи з n-мірними масивами ND4J розробки тієї ж компанії.Deeplearning4j підтримує безліч типів мереж, включаючи багатошаровий персептрон, згорткові мережі, рекурентні мережі та деякі інші. Важливою особливістю цієї бібліотеки є її здатність працювати у кластері. Також бібліотека підтримує навчання мереж із використанням GPU. До недоліків бібліотеки Deeplearning4j, виявлених у процесі роботи з нею, можна віднести складність установки, а також помилки в демонстраційних прикладах, що постачаються разом з бібліотекою, що викликало певні питання щодо надійності бібліотеки і вкрай утруднило її подальше вивчення.

TensorFlow – це повністю відкрита програмна бібліотека, розроблена спеціально для машинного навчання цілого ряду завдань. Спочатку її створила команда Google Brain для своїх власних потреб, а саме для систем, здатних створювати та розвивати штучні нейронні мережі для виявлення та розшифровки образів та кореляцій за аналогією з навчанням та розумінням, що використовуються людьми. Крім основної функціональності машинного навчання, TensorFlow також включає власну систему логування, власний інтерактивний візуалізатор логів і навіть потужну архітектуру доставки даних. Першу версію було випущено публічно в листопаді 2015 року. Працює на будь-яких 64-розрядних операційних системах, у тому числі мобільних. До переваг бібліотеки віднесемо можливість роботи на кількох процесорах, підтримка колекцій потокових графів, черг та доповнень образів для оболонок високого рівня, висока швидкість та гнучкість API для різних мов: Python, C++, Go та Java.Як недолік бібліотеки TensorFlow можна виділити складність роботи з нею для новачків.

Theano – бібліотека мовою Python з відкритим вихідним кодом, яка дозволяє ефективно створювати, обчислювати та оптимізувати математичні вирази з використанням багатовимірних масивів. Для представлення багатовимірних масивів та дій над ними використовується бібліотека NumPy, створена групою вчених з університету Монреаля та призначена насамперед для наукових досліджень. Можливості Theano досить широкі, причому робота з нейронними мережами – це лише одна з її невеликих частин. При цьому саме ця бібліотека є найпопулярнішою і найчастіше згадується, коли йдеться про роботу з Deep Learning. Бібліотека Teano розвивається, щоб стати стандартною бібліотекою для роботи з алгоритмами глибокого навчання. Це накладає такі вимоги до самої бібліотеки: швидкість роботи, легкість використання, легкий перехід із відомих інструментів. Також реалізовано кодогенерацію мовою C для поліпшення продуктивності. Крім цього, розвивається підтримка обчислень на GPU (англ., graphics processing unit, графічний процесор), що є актуальним завданням у наш час, прискорюючи навчання глибоких нейронних мереж у десятки разів у порівнянні з CPU (англ. central processing unit, центральний

3.3 Розробка схеми алгоритму тренування і валідації та тестування нейронної мережі цифрової обробки фотографій

Основними етапами застосування методів машинного навчання є навчання, валідація та тестування. Фаза навчання циклічна, кожен цикл займає одну епоху (повна вибірка навчального набору даних), фаза перевірки (валідаці) слідує за кожною епохою навчання і використовується для перевірки покращення моделі на невідомих даних (з набору даних перевірки) (рис. 3.1).

Послідовність наступних кроків описує алгоритм навчання та перевірки:

1. Отримати архітектуру та підмодулі нейронних мереж.
2. Ініціалізація ваг (випадкова)
3. Налаштування функції втрат.
4. Ініціалізація алгоритму оптимізації.
5. Ініціалізація набору даних для навчання та перевірки.
6. Ініціалізація генератора даних для набору даних тестування та перевірки.
7. Епоха навчання:
    - Для кожного міні-лоту в генераторі навчальних даних:
    - Градієнтне очищення.
    - Крок вперед.
    - Розрахунок втрат.
    - Крок назад
    - Крок оптимізації.
8. Епоха підтвердження:
    - Для кожного міні-лоту з генератора даних перевірки:
    - Крок вперед.
    - Розрахунок втрат.
9. Збереження моделі.
10. Тюнінг моделі.
11. Збереження отриманої моделі.

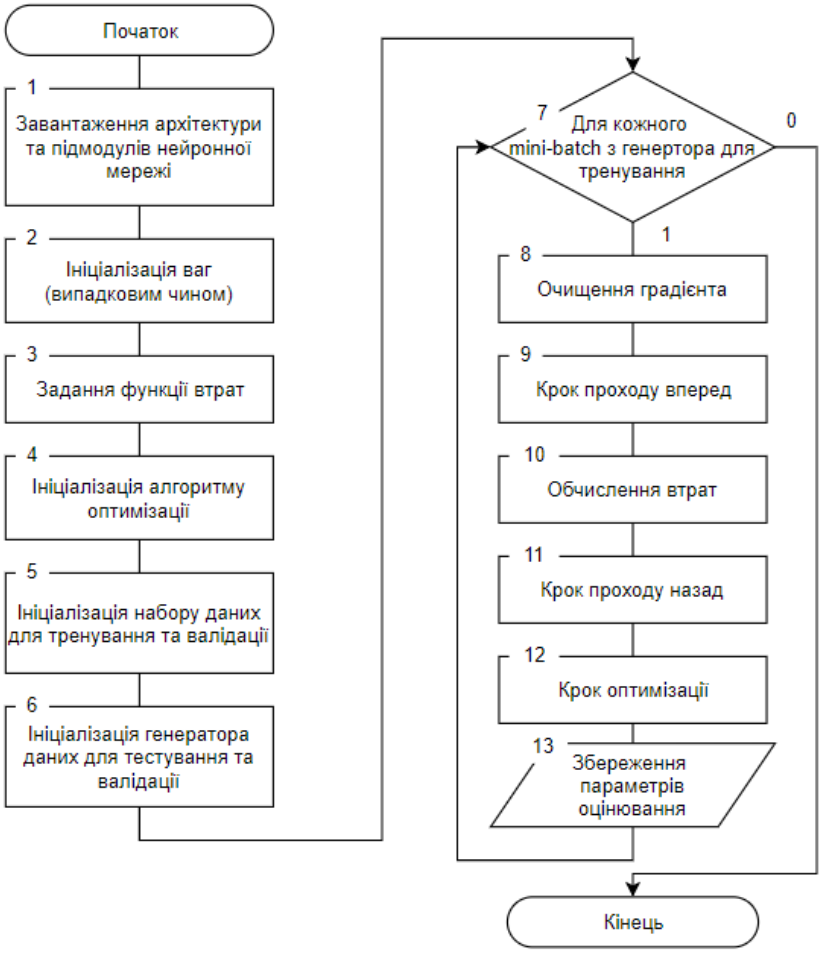

Рисунок 3.1 – Алгоритм тренування й валідації нейронної мережі цифрової обробки фотографій

Алгоритм тестування мережі можна описати так (рис. 3.2):

1. Завантажити параметри моделі з файлу.
2. Отримати архітектуру та підмодулі нейронних мереж.
3. Застосувати параметри моделі
4. Ініціалізувати функції втрати.
5. Ініціалізувати набір даних для тестування.
6. Ініціалізувати завантажувач даних для набору даних для перевірки.
7. Епоха тестування:
    - Для кожного міні-лоту із завантажувачем даних для тестування:
    - Крок вперед.
    - Розрахунок втрат.
    - Зберегти модель.

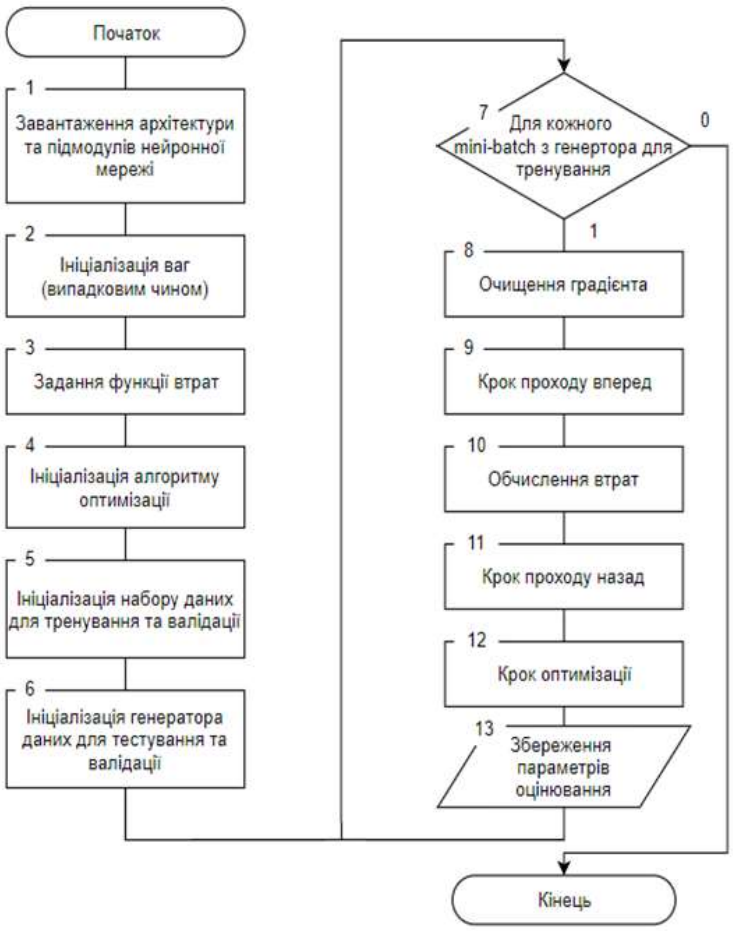

Рисунок 3.2 – Алгоритм тестування нейронної мережі цифрової обробки фотографій

3.4 Побудова UML-діаграми класів програмного забезпечення цифрової обробки фотографій

Для програмної реалізації інформаіційної технології цифрової обробки фотографій було розроблено класи Main, ColorizeImage, LabGamut , GUIDesign (рис. 3.3).

Основний клас (Main) - початкова програмна діяльність. Викликає клас GUIDesign.

Клас GUIDesign – керує відображенням головного меню програми та взаємодією з користувачем.

Клас ColorizeImage - основна логіка розфарбовування цифрових фотографій.

За логіку програми відповідає клас LabGamut, в якому вибирається колір пропонованих підказок.

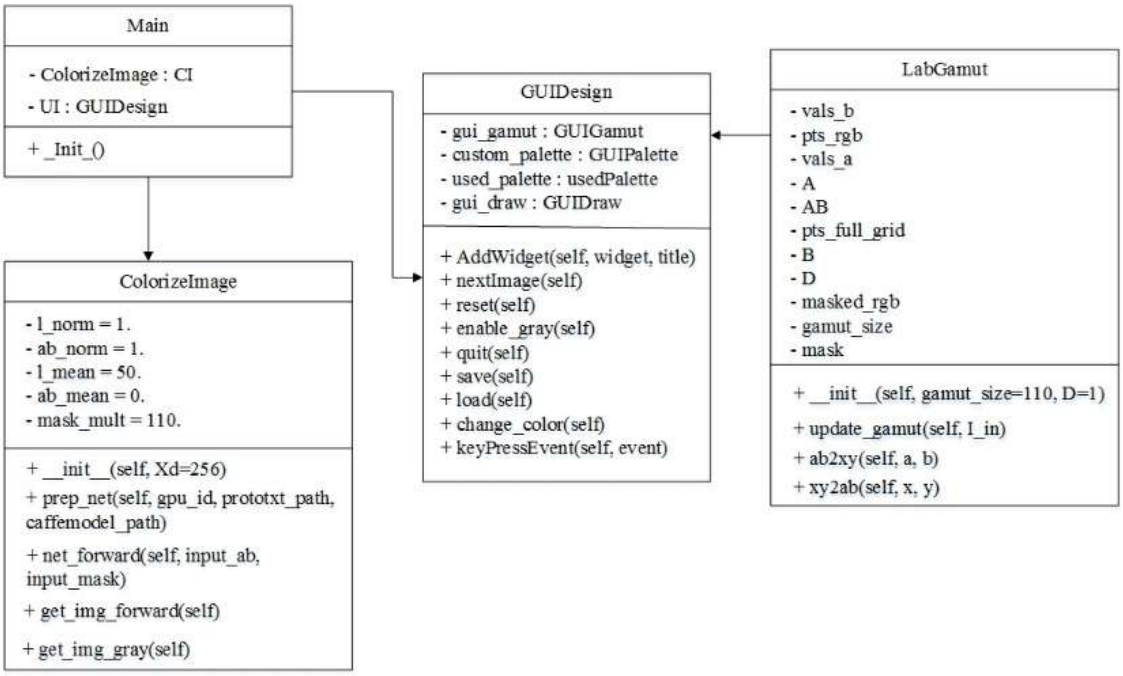

Рисунок 3.3 – UML-діаграма класів програмного забезпечення цифрової обробки фотографій

3.5 Тестування розробленого програмного забезпечення цифрової обробки фотографій та аналіз результатів його роботи

Розроблене програмне забезпечення цифрової обробки фотографій було протестовано, що підтвердило коректність його роботи.

Було проведено 100 запусків програми, протестовано можливості її роботи, що дало можливість адекватно оцінити результати.

Після запуску програми відкриється головне вікно (рис. 3.4 - 3.6).

Рисунок 3.4 – Інтерфейс програмного забезпечення цифрової обробки фотографій без використання користувацьких підказок

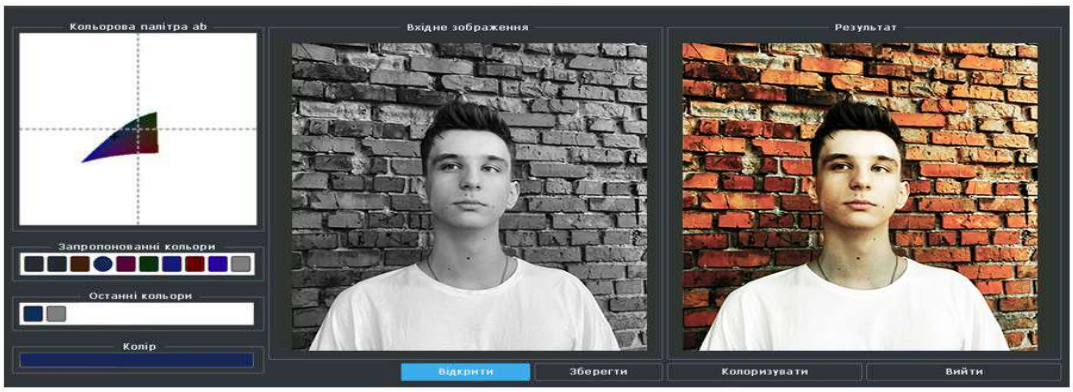

Рисунок 3.5 – Інтерфейс програмного забезпечення цифрової обробки фотографій з користувацькими підказками

Головне вікно розробленої програми цифрової обробки фотографій містить такі елементи взаємодії з користувачем:

• Кнопка «Відкрити» - для відкриття чорно-білого зображення, яке потрібно обробити;

• Панель введення відображає відкрите чорно-біле зображення. Ця панель також використовується користувачем для вставки кольорових підказок;

• Панель паралельної палітри Ab – відображає колірну гаму a, b каналів CIELAB для кожної точки, визначеної користувачем. Ця гама виводиться із значення L-каналу в даній точці зображення. На ній можна вибрати колір точки підказки;

• Панель «Пропоновані кольори» – відображає набір найбільш підходящих кольорів для кожної точки, запропонованої нейронною мережею. На ній можна вибрати колір точки підказки;

• Панель «Останні кольори» - відображає останні використані кольори. На ній можна вибрати колір точки підказки;

• Колірна панель - відображає поточний колір;

• Кнопка «Колір» - для початку розфарбовування;

• Панель «Результат» містить кольорове зображення. Зображення оновлюється щоразу, коли починається розфарбовування;

• кнопка «Зберегти» - для збереження кольорової фотографії на комп'ютері;

• Кнопка «Вийти» - для закриття програми.

Як видно, результат розфарбовування без спеціальної підказки відрізняється від результату з 1 підказкою. У табл. 3.1 наведено результати порівняння роботи розробленого програмного забезпечення з іншими програмами для обробки чорно-білих зображень. На рисунку 3.6 показано порівняння результатів розробленої програми з аналогом.

Рисунок 3.6 – Приклад обробки програмним забезпеченням і програмою аналогом

Отже, після виконання тестових запусків програми можна зробити наступні висновки: програма розфарбовує зображення з підказкою кольору і без підказки. Було виявлено, що зображення, отримане після застосування підказки, суб'єктивно краще розфарбовується , на відміну від методу без підказки.

Оцінка якості проводилась так: ми зробили кольорову фотографію, з неї зробили чорно-білу версію. Потім це чорно-біле зображення було розфарбовано в програмному забезпеченні та програмою Akvis Coloriage. Потім ми розрахували індекс структурної схожості DSSIM, який вимірює подібність двох зображень [4]. Він знаходиться в межах від -1 до 1. Ми розрахували цей індекс для наступних пар зображень: оригінал - зображення розфарбоване розробленим програмним забезпеченням; оригінал - кольорове зображення Akvis Coloriage.

Індекс SSIM розраховується для різних вікон зображення. Міра між двома вікнами x і y загального розміру [4] є:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.1}$$

де μ_x - середнє значення x;

μ_y - середнє значення y;

σ^2_x - дисперсія x;

σ^2_y - дисперсія y;

σ_{xy} - коваріантність x і y;

$c_1=(k_1L)^2$, $c_2=(k_2L)^2$ - дві змінні для стабілізації ділення зі слабким знаменником;

L - динамічний діапазон значень пікселів ($2^{\#\text{bits per pixel}}$-1);

k_1=0.01, k_2=0.03.

Структурна несхожість (DSSIM) може бути отримана з SSIM:

$$\mathrm{DSSIM}(x,y) = \frac{1 - \mathrm{SSIM}(x,y)}{2} \tag{3.2}$$

Таблиця 3.1 – Порівняльний аналіз якості цифрової обробки фотографії

Програма	DSSIM
Розроблена програма без підказок	0,094
Розроблена програма з 5 підказками	0,102
Розроблена програма з 10 підказками	0,106
AKVIS Coloriage	0,087

Розроблене програмне забезпечення характеризується кращою якістю цифрової обробки в порівнянні з програмною аналогом. Якість у порівнянні з прототипом зросла майже на 2,8%.

3.6 Висновок до розділу 3

У даному розділі обгрунтовано вибір мови програмування, інтегрованого середовища розробки, використання нейромережевої бібліотеки. В роботі використано мову програмування Python та бібліотеку Caffe, які є найпотужнішими інструментами для роботи зі згортковими нейронними мережами.

Розроблено алгоритми тренування та валідації й тестування нейронної мережі розфарбовування цифрових фотографій. Розроблено UML-діаграму класів програмного хабезпечення розфарбовування цифрових фотографій. Здійснено його програмну реалізацію.

Проведено тестування розробленого програмного забезпечення, яке підтвердило коректність його роботи. Також було проведено порівняльний аналіз створеного програмного забезпечення з існуючим аналогом з визначенням критерія якості розфарбовування. Доведено, що якість йифрової обробки в порівнянні з прототипом зросла майже на 2,8%.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Проведення комерційного та технологічного аудиту має на меті багатоаспектну оцінку науково-технічного рівня та комерційного потенціалу розробок, які виникають в результаті науково-технічної діяльності, зокрема, під час виконання магістерської кваліфікаційної роботи. Цей вид аудиту є ключовим етапом у визначенні ефективності та перспективності розробок з точки зору їхнього впровадження на ринку та взаємодії з бізнес-середовищем.

Науково-технічний рівень оцінюється з точки зору інноваційності, наявності унікальних технічних рішень та їхнього практичного застосування. Також важливо враховувати ступінь науково-технічного освоєння проблеми, над якою працювала дослідницька група.

Комерційний потенціал включає в себе аналіз ринкових можливостей, потенційних конкурентів, стратегії маркетингу та планів впровадження на ринок. Економічна вартість та стійкість до ризиків також є важливими аспектами оцінки комерційного потенціалу.

Важливо враховувати, що результати комерційного та технологічного аудиту можуть слугувати основою для удосконалення розробок, розробки стратегії впровадження та взаємодії з іншими суб'єктами ринку.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу засобу поляриметричного аналізу оптично активних рідни

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	4	3	4
Ринкові переваги (наявність аналогів)	3	4	4
Ринкові переваги (ціна продукту)	3	4	3
Ринкові переваги (технічні властивості)	4	3	2
Ринкові переваги (експлуатаційні витрати)	3	2	2
Ринкові перспективи (розмір ринку)	4	3	3
Ринкові перспективи (конкуренція)	3	4	4
Практична здійсненність (наявність фахівців)	3	3	3
Практична здійсненність (наявність фінансів)	4	2	3
Практична здійсненність (необхідність нових матеріалів)	3	3	4
Практична здійсненність (термін реалізації)	4	3	3
Практична здійсненність (розробка документів)	3	3	2
Сума балів	43	37	37
Середньоарифметична сума балів, СБ	39		

За результатами розрахунків, наведених в табл. 4.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології цифрової обробки фотографії – вище середнього.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам,

копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	10000	455	30	13650
Розробник	8000	364	30	10920
Консультанти	6700	305	2	610
Всього:	25180			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_і$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; $t_і$ – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2023 році $М_м$=6700 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Проведені розрахунки зведемо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмування	12	1	38,1	1,0	457,2
Налагодження	25	3	44,9	1,18	1122,5
Тестування	10	2	41,5	1,09	415
Всього					1994,7

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (25180 + 1994{,}7) = 2717{,}5 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (25180 + 1994{,}7 + 2717{,}5) = 6576 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Розрахунок витрат на матеріали. Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$М = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де $Н_і$ – кількість матеріалів і-го виду, шт.; $Ц_і$ – ціна матеріалів і-го виду, грн.; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів матеріалів.

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Матеріали, що використані на розробку

Найменування матеріали	Ціна за одиницю, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Офісний папір А4	250	1	250
Тонер для принтера	200	1	200
Всього, з врахуванням коефіцієнта транспортних витрат			495

Спецустаткування для наукових (експериментальних) робіт. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами.

$$В_{спец} = \sum_{1}^{к} Ц_{і} \cdot С_{пр.і} \cdot К_{і}, \quad (4.5)$$

де $Ц_{і}$ – ціна придбання спецустаткування і-го виду, грн.;

$С_{пр.і}$ – кількість одиниць спецустаткування відповідного виду, шт.; $К_{і}$ – коефіцієнт транспортних витрат,

$К_{і}$ = (1,1…1,15); n – кількість видів спецустаткування.

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на придбання спецустаткування

Найменування спецустаткування	Ціна за одиницю, грн.	Витрачено	Вартість спецустаткування, грн.
Проектор	12500	1	12500
Екран для проектора	3000	1	3000
Стіл	2500	1	2500
Всього, з врахуванням коефіцієнта транспортних витрат			19800

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і}, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_{і}$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_{і}$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
PyCharm	9500	1	9500
Miccrosoft Office	4800	1	4800
Всього, з врахуванням коефіцієнта інсталяції та налагодження			15730

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.8)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук (DELL Vostro 15 3000)	12500	5	2	417
Принтер Epson 3010	5000	5	2	167
Всього	584			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$\text{Ве} = \sum \frac{W_i \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_\text{впі}}{\text{ККД}} = \frac{0{,}03 \cdot 300 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} + \frac{0{,}4 \cdot 120 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 414 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук (DELL Vostro 15 3000)	0,03	300
Принтер Epson 3010	0,4	120

де W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Ів} = \left(З_\text{о} + З_\text{р}\right) \cdot \frac{\text{Н}_\text{ів}}{100\%} = (25180 + 1994{,}7) \cdot \frac{75}{100} = 20381 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (25180 + 1994{,}7) \cdot \frac{115}{100} = 31251 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} = 25180 + 1994{,}7 + 2717{,}5 + 6576 + 495 + 19800 + 15730 + 584$$
$$+ 414 + 20381 + 31251 = 125123{,}2 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{125123{,}2}{0{,}9} = 139026 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії впровадження, то η=0,9.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість

споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.9)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного

продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((2500 - 1500) \cdot 1000 - (1000 - 1000) \cdot 1500) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 219120 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{219120}{(1+0{,}1)^1} = 199200 \text{ грн.},$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо T=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 1 \cdot 139026 = 139026 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=1...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 199200 - 139026 = 60174 \text{ грн.},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[1]{1 + \frac{60174}{139026}} = 1{,}18,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар’єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Далі розраховуємо період окупності інвестицій T_o, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_o = \frac{1}{E_в} = \frac{1}{1{,}18} = 0{,}85 \text{ року.}$$

Оскільки T_o=0,85<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія цифрової обробки фотографії» становить 42 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,8 рази.

Також термін окупності становить 0,85 року, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія цифрової обробки фотографії».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено інформаційну технологію цифрової обробки фотографій.

В роботі проаналізовано основні технології розфарбовування фотографій, доведено актуальність дослідження, виявлено основні недоліки при розфарбовування чорно-білих фотографій, які полягають в неможливості користувача впливати на результат розфарбовування. Аналіз програм аналогів показав, не існує такого програмного рішення, яке б одночасно дозволяло здійснювати швидко і якісно розфарбовування чорно-білих фотографій. Здійснено постановку задачі. Обгрунтовано вибір колірної моделі для розфарбовування зображень. Обгрунтовано вибір згорткової нейронної мережі, як один з найефективніших для вирішення задачі розфарбовування. Обгрунтовано вибір методу навчання нейронної мережі. Розроблено архітектуру нейронної мережі для розфарбовування цифрових фотографій, що грунтується на архітектурі U-net. Розроблено загальну схему алгоритму функціонування інформаційної технології цифрової обробки фотографій.

В роботі було обгрунтовано вибір мови програмування, інтегрованого середовища розробки, використання нейромережевої бібліотеки. В роботі використано мову програмування Python та бібліотеку Caffe, які є найпотужнішими інструментами для роботи зі згортковими нейронними мережами. Розроблено алгоритми тренування та валідації й тестування нейронної мережі розфарбовування цифрових фотографій. Розроблено UML-діаграму класів програмного забезпечення розфарбовування цифрових фотографій. Проведено тестування розробленої інформаційної технології, яке підтвердило коректність її роботи. Також було проведено порівняльний аналіз створеного програмного забезпечення з існуючим аналогом з визначенням критерія якості розфарбовування. Доведено, що якість цифрової обробки в порівнянні з прототипом зросла майже на 2,8%.

Здійснено економічні розрахунки, які доводять доцільність розробки нової інформаційної технології цифрової обробки фотографій.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія цифрової обробки фотографії» становить 42 бали,

що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,8 рази.

Також термін окупності становить 0,85 року, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок. Спрогнозовано витрати на виконання наукової роботи та впровадження результатів, які склали 125123,2 грн, розраховано період окупності – 0,85 року.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія цифрової обробки фотографії».

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Данилишин В.В., Колодний В.В. Особливості цифрової обробки фотографій в Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» − [Електронний ресурс]. – https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/19942
2. Automatic Colorization [Electronic resource] / Tinyclouds – Mode of access: http://tinyclouds.org/colorize/
3. Adobe Photoshop [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Adobe_Photoshop
4. Colorize Photo [Електроний ресурс] – Режим доступу: https://www.colorizephoto.com/
5. Algorithmia Colorize Photos [Electronic resource]] – Mode of access: https://demos.algorithmia.com/colorize-photos/
6. AKVIS Coloriage [Електроний ресурс] – Режим доступу: http://akvis.com/ru/coloriage/index.php
7. Doyle, W. Operations useful for similarity invariant pattern recognition. // Journal ACM. [Текст]/ Doyle, W.. – 1962 Том. 9, № 2. С. 259-267.
8. Paul J. Werbos , The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting. / New York, NY: John Wiley & Sons, Inc. – 1994.
9. Alona, What is Backpropagation / Alona // Personal page [Electronic resource] – Mode of access: https://alonalj.github.io/2016/12/10/What-is-Backpropagation/
10. Gashler Michael S., Training Deep Fourier Neural Networks to Fit Time-Series Data / Gashler, Michael S., and Stephen C. Ashmore in Intelligent Computing in Bioinformatics. Springer International Publishing, 2014. 48-55 // arXiv [Electronic resource]. – 2014 – Mode of access: https://arxiv.org/pdf/1405.2262.pdf

11. D. Sussillo, Random walks: Training very deep nonlinear feed-forward networks with smart initialization / Sussillo, David, and L. F. Abbott in CoRR, 2014. // arXiv [Electronic resource] – 2014 – Mode of access: https://arxiv.org/pdf/1412.6558.pdf
12. Колірна модель [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Колірна_модель
13. HSV [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/HSV
14. CIELAB [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/CIELAB
15. U-Net: Convolutional Networks [Electronic resource]. – Mode of access: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
16. Caffe – Deep Learning Framework [Electronic resource] – Mode of access: http://caffe.berkeleyvision.org/
17. Машинне навчання з використанням мікрокомп’ютерів – [Електронний ресурс] – Режим доступу: http://man.gov.ua/files/49/Machine_Nav4ann_Mogilniy.pdf
18. Python documentation – [Електронний ресурс] – https://www.python.org/doc/
19. Image Classification on ImageNet – [Електронний ресурс] – Режим доступу: https://paperswithcode.com/sota/image-classification-on-imagenet
20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія цифрової обробки фотографії

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 87,3% Схожість 12,7%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Данилишин В.В.

Керівник роботи Колодний В.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from skimage import color
from sklearn.cluster import KMeans
import os
from scipy.ndimage.interpolation import zoom


def create_temp_directory(path_template, N=1e8):
    print(path_template)
    cur_path = path_template % np.random.randint(0, N)
    while(os.path.exists(cur_path)):
        cur_path = path_template % np.random.randint(0, N)
    print('Creating directory: %s' % cur_path)
    os.mkdir(cur_path)
    return cur_path


def lab2rgb_transpose(img_l, img_ab):
    ''' INPUTS
            img_l     1xXxX     [0,100]
            img_ab     2xXxX     [-100,100]
        OUTPUTS
            returned value is XxXx3 '''
    pred_lab = np.concatenate((img_l, img_ab), axis=0).transpose((1, 2, 0))
    pred_rgb = (np.clip(color.lab2rgb(pred_lab), 0, 1) * 255).astype('uint8')
    return pred_rgb


def rgb2lab_transpose(img_rgb):
    ''' INPUTS
            img_rgb XxXx3
        OUTPUTS
            returned value is 3xXxX '''
    return color.rgb2lab(img_rgb).transpose((2, 0, 1))


class ColorizeImageBase():
    def __init__(self, Xd=256, Xfullres_max=10000):
        self.Xd = Xd
        self.img_l_set = False
        self.net_set = False
        self.Xfullres_max = Xfullres_max  # maximum size of maximum dimension
        self.img_just_set = False  # this will be true whenever image is just
loaded
        # net_forward can set this to False if they want

    def prep_net(self):
        raise Exception("Should be implemented by base class")

    # ***** Image prepping *****
    def load_image(self, input_path):
        # rgb image [CxXdxXd]
        im = cv2.cvtColor(cv2.imread(input_path, 1), cv2.COLOR_BGR2RGB)
        self.img_rgb_fullres = im.copy()
        self._set_img_lab_fullres_()
```

```
            im = cv2.resize(im, (self.Xd, self.Xd))
            self.img_rgb = im.copy()
            #                               self.img_rgb                                    =
sp.misc.imresize(plt.imread(input_path),(self.Xd,self.Xd)).transpose((2,0,1))

            self.img_l_set = True

            # convert into lab space
            self._set_img_lab_()
            self._set_img_lab_mc_()

        def set_image(self, input_image):
            self.img_rgb_fullres = input_image.copy()
            self._set_img_lab_fullres_()

            self.img_l_set = True

            self.img_rgb = input_image
            # convert into lab space
            self._set_img_lab_()
            self._set_img_lab_mc_()

        def net_forward(self, input_ab, input_mask):
            # INPUTS
            #     ab         2xXxX     input color patches (non-normalized)
            #     mask      1xXxX    input mask, indicating which points have been
provided
            # assumes self.img_l_mc has been set

            if(not self.img_l_set):
                print('I need to have an image!')
                return -1
            if(not self.net_set):
                print('I need to have a net!')
                return -1

            self.input_ab = input_ab
            self.input_ab_mc = (input_ab - self.ab_mean) / self.ab_norm
            self.input_mask = input_mask
            self.input_mask_mult = input_mask * self.mask_mult
            return 0

        def get_result_PSNR(self, result=-1, return_SE_map=False):
            if np.array((result)).flatten()[0] == -1:
                cur_result = self.get_img_forward()
            else:
                cur_result = result.copy()
            SE_map = (1. * self.img_rgb - cur_result)**2
            cur_MSE = np.mean(SE_map)
            cur_PSNR = 20 * np.log10(255. / np.sqrt(cur_MSE))
            if return_SE_map:
                return(cur_PSNR, SE_map)
            else:
                return cur_PSNR

        def get_img_forward(self):
            # get image with point estimate
            return self.output_rgb

        def get_img_gray(self):
            # Get black and white image
            return lab2rgb_transpose(self.img_l, np.zeros((2, self.Xd, self.Xd)))
```

```
        def get_img_gray_fullres(self):
            # Get black and white image
            return        lab2rgb_transpose(self.img_l_fullres,        np.zeros((2,
self.img_l_fullres.shape[1], self.img_l_fullres.shape[2])))

        def get_img_fullres(self):
            # This assumes self.img_l_fullres, self.output_ab are set.
            # Typically, this means that set_image() and net_forward()
            # have been called.
            # bilinear upsample
            zoom_factor    =    (1,    1.    *    self.img_l_fullres.shape[1]    /
self.output_ab.shape[1], 1. * self.img_l_fullres.shape[2] / self.output_ab.shape[2])
            output_ab_fullres = zoom(self.output_ab, zoom_factor, order=1)

            return lab2rgb_transpose(self.img_l_fullres, output_ab_fullres)

        def get_input_img_fullres(self):
            zoom_factor    =    (1,    1.    *    self.img_l_fullres.shape[1]    /
self.input_ab.shape[1], 1. * self.img_l_fullres.shape[2] / self.input_ab.shape[2])
            input_ab_fullres = zoom(self.input_ab, zoom_factor, order=1)
            return lab2rgb_transpose(self.img_l_fullres, input_ab_fullres)

        def get_input_img(self):
            return lab2rgb_transpose(self.img_l, self.input_ab)

        def get_img_mask(self):
            # Get black and white image
            return  lab2rgb_transpose(100.  *  (1  -  self.input_mask),  np.zeros((2,
self.Xd, self.Xd)))

        def get_img_mask_fullres(self):
            # Get black and white image
            zoom_factor    =    (1,    1.    *    self.img_l_fullres.shape[1]    /
self.input_ab.shape[1], 1. * self.img_l_fullres.shape[2] / self.input_ab.shape[2])
            input_mask_fullres = zoom(self.input_mask, zoom_factor, order=0)
            return lab2rgb_transpose(100. * (1 - input_mask_fullres), np.zeros((2,
input_mask_fullres.shape[1], input_mask_fullres.shape[2])))

        def get_sup_img(self):
            return lab2rgb_transpose(50 * self.input_mask, self.input_ab)

        def get_sup_fullres(self):
            zoom_factor    =    (1,    1.    *    self.img_l_fullres.shape[1]    /
self.output_ab.shape[1], 1. * self.img_l_fullres.shape[2] / self.output_ab.shape[2])
            input_mask_fullres = zoom(self.input_mask, zoom_factor, order=0)
            input_ab_fullres = zoom(self.input_ab, zoom_factor, order=0)
            return lab2rgb_transpose(50 * input_mask_fullres, input_ab_fullres)

        # ***** Private functions *****
        def _set_img_lab_fullres_(self):
            # adjust full resolution image to be within maximum dimension is within
Xfullres_max
            Xfullres = self.img_rgb_fullres.shape[0]
            Yfullres = self.img_rgb_fullres.shape[1]
            if Xfullres > self.Xfullres_max or Yfullres > self.Xfullres_max:
                if Xfullres > Yfullres:
                    zoom_factor = 1. * self.Xfullres_max / Xfullres
                else:
                    zoom_factor = 1. * self.Xfullres_max / Yfullres
                self.img_rgb_fullres  =  zoom(self.img_rgb_fullres,  (zoom_factor,
zoom_factor, 1), order=1)

            self.img_lab_fullres                                                =
color.rgb2lab(self.img_rgb_fullres).transpose((2, 0, 1))
```

```
            self.img_l_fullres = self.img_lab_fullres[[0], :, :]
            self.img_ab_fullres = self.img_lab_fullres[1:, :, :]

        def _set_img_lab_(self):
            # set self.img_lab from self.im_rgb
            self.img_lab = color.rgb2lab(self.img_rgb).transpose((2, 0, 1))
            self.img_l = self.img_lab[[0], :, :]
            self.img_ab = self.img_lab[1:, :, :]

        def _set_img_lab_mc_(self):
            # set self.img_lab_mc from self.img_lab
            # lab image, mean centered [XxYxX]
            self.img_lab_mc = self.img_lab / np.array((self.l_norm, self.ab_norm,
self.ab_norm))[:, np.newaxis, np.newaxis] - np.array(
                (self.l_mean / self.l_norm, self.ab_mean / self.ab_norm,
self.ab_mean / self.ab_norm))[:, np.newaxis, np.newaxis]
            self._set_img_l_()

        def _set_img_l_(self):
            self.img_l_mc = self.img_lab_mc[[0], :, :]
            self.img_l_set = True

        def _set_img_ab_(self):
            self.img_ab_mc = self.img_lab_mc[[1, 2], :, :]

        def _set_out_ab_(self):
            self.output_lab = rgb2lab_transpose(self.output_rgb)
            self.output_ab = self.output_lab[1:, :, :]


    class ColorizeImageTorch(ColorizeImageBase):
        def __init__(self, Xd=256, maskcent=False):
            print('ColorizeImageTorch instantiated')
            ColorizeImageBase.__init__(self, Xd)
            self.l_norm = 1.
            self.ab_norm = 1.
            self.l_mean = 50.
            self.ab_mean = 0.
            self.mask_mult = 1.
            self.mask_cent = .5 if maskcent else 0

            # Load grid properties
            self.pts_in_hull = np.array(np.meshgrid(np.arange(-110, 120, 10),
np.arange(-110, 120, 10))).reshape((2, 529)).T

        # ***** Net preparation *****
        def prep_net(self, gpu_id=None, path='', dist=False):
            import torch
            import models.pytorch.model as model
            print('path = %s' % path)
            print('Model set! dist mode? ', dist)
            self.net = model.SIGGRAPHGenerator(dist=dist)
            state_dict = torch.load(path)
            if hasattr(state_dict, '_metadata'):
                del state_dict._metadata

            # patch InstanceNorm checkpoints prior to 0.4
            for key in list(state_dict.keys()):  # need to copy keys here because we
mutate in loop
                self.__patch_instance_norm_state_dict(state_dict, self.net,
key.split('.'))
            self.net.load_state_dict(state_dict)
            if gpu_id != None:
                self.net.cuda()
```

```
            self.net.eval()
            self.net_set = True

        def __patch_instance_norm_state_dict(self, state_dict, module, keys, i=0):
            key = keys[i]
            if i + 1 == len(keys):  # at the end, pointing to a parameter/buffer
                if module.__class__.__name__.startswith('InstanceNorm') and \
                        (key == 'running_mean' or key == 'running_var'):
                    if getattr(module, key) is None:
                        state_dict.pop('.'.join(keys))
                if module.__class__.__name__.startswith('InstanceNorm') and \
                   (key == 'num_batches_tracked'):
                    state_dict.pop('.'.join(keys))
            else:
                self.__patch_instance_norm_state_dict(state_dict,   getattr(module,
key), keys, i + 1)

        # ***** Call forward *****
        def net_forward(self, input_ab, input_mask):
            # INPUTS
            #     ab         2xXxX     input color patches (non-normalized)
            #     mask     1xXxX    input mask, indicating which points have been
provided
            # assumes self.img_l_mc has been set

            if ColorizeImageBase.net_forward(self, input_ab, input_mask) == -1:
                return -1

            # net_input_prepped = np.concatenate((self.img_l_mc, self.input_ab_mc,
self.input_mask_mult), axis=0)

            # return prediction
            # self.net.blobs['data_l_ab_mask'].data[...] = net_input_prepped
            # embed()
            output_ab    =    self.net.forward(self.img_l_mc,    self.input_ab_mc,
self.input_mask_mult, self.mask_cent)[0, :, :, :].cpu().data.numpy()
            self.output_rgb = lab2rgb_transpose(self.img_l, output_ab)
            #        self.output_rgb        =        lab2rgb_transpose(self.img_l,
self.net.blobs[self.pred_ab_layer].data[0, :, :, :])

            self._set_out_ab_()
            return self.output_rgb

        def get_img_forward(self):
            # get image with point estimate
            return self.output_rgb

        def get_img_gray(self):
            # Get black and white image
            return lab2rgb_transpose(self.img_l, np.zeros((2, self.Xd, self.Xd)))


    class ColorizeImageTorchDist(ColorizeImageTorch):
        def __init__(self, Xd=256, maskcent=False):
            ColorizeImageTorch.__init__(self, Xd)
            self.dist_ab_set = False
            self.pts_grid   =   np.array(np.meshgrid(np.arange(-110,   120,   10),
np.arange(-110, 120, 10))).reshape((2, 529)).T
            self.in_hull = np.ones(529, dtype=bool)
            self.AB = self.pts_grid.shape[0]  # 529
            self.A = int(np.sqrt(self.AB))  # 23
            self.B = int(np.sqrt(self.AB))  # 23
            self.dist_ab_full = np.zeros((self.AB, self.Xd, self.Xd))
            self.dist_ab_grid = np.zeros((self.A, self.B, self.Xd, self.Xd))
```

```
            self.dist_entropy = np.zeros((self.Xd, self.Xd))
            self.mask_cent = .5 if maskcent else 0

        def prep_net(self, gpu_id=None, path='', dist=True, S=.2):
            ColorizeImageTorch.prep_net(self, gpu_id=gpu_id, path=path, dist=dist)
            # set S somehow

        def net_forward(self, input_ab, input_mask):
            # INPUTS
            #     ab         2xXxX     input color patches (non-normalized)
            #     mask     1xXxX    input mask, indicating which points have been
provided
            # assumes self.img_l_mc has been set

            # embed()
            if ColorizeImageBase.net_forward(self, input_ab, input_mask) == -1:
                return -1

            # set distribution
            (function_return,   self.dist_ab)   =   self.net.forward(self.img_l_mc,
self.input_ab_mc, self.input_mask_mult, self.mask_cent)
            function_return = function_return[0, :, :, :].cpu().data.numpy()
            self.dist_ab = self.dist_ab[0, :, :, :].cpu().data.numpy()
            self.dist_ab_set = True

            # full grid, ABxXxX, AB = 529
            self.dist_ab_full[self.in_hull, :, :] = self.dist_ab

            # gridded, AxBxXxX, A = 23
            self.dist_ab_grid = self.dist_ab_full.reshape((self.A, self.B, self.Xd,
self.Xd))

            # return
            return function_return

        def get_ab_reccs(self, h, w, K=5, N=25000, return_conf=False):
            ''' Recommended colors at point (h,w)
            Call this after calling net_forward
            '''
            if not self.dist_ab_set:
                print('Need to set prediction first')
                return 0

            # randomly sample from pdf
            cmf = np.cumsum(self.dist_ab[:, h, w])  # CMF
            cmf = cmf / cmf[-1]
            cmf_bins = cmf

            # randomly sample N points
            rnd_pts = np.random.uniform(low=0, high=1.0, size=N)
            inds = np.digitize(rnd_pts, bins=cmf_bins)
            rnd_pts_ab = self.pts_in_hull[inds, :]

            # run k-means
            kmeans = KMeans(n_clusters=K).fit(rnd_pts_ab)

            # sort by cluster occupancy
            k_label_cnt = np.histogram(kmeans.labels_, np.arange(0, K + 1))[0]
            k_inds = np.argsort(k_label_cnt, axis=0)[::-1]

            cluster_per = 1. * k_label_cnt[k_inds] / N  # percentage of points within
cluster
            cluster_centers = kmeans.cluster_centers_[k_inds, :]  # cluster centers
```

```
            # cluster_centers = np.random.uniform(low=-100,high=100,size=(N,2))
            if return_conf:
                return cluster_centers, cluster_per
            else:
                return cluster_centers

        def compute_entropy(self):
            # compute the distribution entropy (really slow right now)
            self.dist_entropy = np.sum(self.dist_ab * np.log(self.dist_ab), axis=0)

        def plot_dist_grid(self, h, w):
            # Plots distribution at a given point
            plt.figure()
            plt.imshow(self.dist_ab_grid[:, :, h, w], extent=[-110, 110, 110, -110],
interpolation='nearest')
            plt.colorbar()
            plt.ylabel('a')
            plt.xlabel('b')

        def plot_dist_entropy(self):
            # Plots distribution at a given point
            plt.figure()
            plt.imshow(-self.dist_entropy, interpolation='nearest')
            plt.colorbar()


    class ColorizeImageCaffe(ColorizeImageBase):
        def __init__(self, Xd=256):
            print('ColorizeImageCaffe instantiated')
            ColorizeImageBase.__init__(self, Xd)
            self.l_norm = 1.
            self.ab_norm = 1.
            self.l_mean = 50.
            self.ab_mean = 0.
            self.mask_mult = 110.

            self.pred_ab_layer = 'pred_ab'  # predicted ab layer

            # Load grid properties
            self.pts_in_hull_path = './data/color_bins/pts_in_hull.npy'
            self.pts_in_hull = np.load(self.pts_in_hull_path)  # 313x2, in-gamut

        # ***** Net preparation *****
        def prep_net(self, gpu_id, prototxt_path='', caffemodel_path=''):
            import caffe
            print('gpu_id = %d, net_path = %s, model_path = %s' % (gpu_id,
prototxt_path, caffemodel_path))
            if gpu_id == -1:
                caffe.set_mode_cpu()
            else:
                caffe.set_device(gpu_id)
                caffe.set_mode_gpu()
            self.gpu_id = gpu_id
            self.net = caffe.Net(prototxt_path, caffemodel_path, caffe.TEST)
            self.net_set = True

            # automatically set cluster centers
            if len(self.net.params[self.pred_ab_layer][0].data[...].shape) == 4 and
self.net.params[self.pred_ab_layer][0].data[...].shape[1] == 313:
                print('Setting ab cluster centers in layer: %s' % self.pred_ab_layer)
                self.net.params[self.pred_ab_layer][0].data[:,   :,   0,   0]   =
self.pts_in_hull.T

            # automatically set upsampling kernel
```

```
            for layer in self.net._layer_names:
                if layer[-3:] == '_us':
                    print('Setting upsampling layer kernel: %s' % layer)
                    self.net.params[layer][0].data[:, 0, :, :] = np.array(((.25, .5,
.25, 0), (.5, 1., .5, 0), (.25, .5, .25, 0), (0, 0, 0, 0)))[np.newaxis, :, :]

        # ***** Call forward *****
        def net_forward(self, input_ab, input_mask):
            # INPUTS
            #     ab         2xXxX     input color patches (non-normalized)
            #     mask     1xXxX    input mask, indicating which points have been
provided
            # assumes self.img_l_mc has been set

            if ColorizeImageBase.net_forward(self, input_ab, input_mask) == -1:
                return -1

            net_input_prepped  =  np.concatenate((self.img_l_mc,  self.input_ab_mc,
self.input_mask_mult), axis=0)

            self.net.blobs['data_l_ab_mask'].data[...] = net_input_prepped
            self.net.forward()

            # return prediction
            self.output_rgb             =             lab2rgb_transpose(self.img_l,
self.net.blobs[self.pred_ab_layer].data[0, :, :, :])

            self._set_out_ab_()
            return self.output_rgb

        def get_img_forward(self):
            # get image with point estimate
            return self.output_rgb

        def get_img_gray(self):
            # Get black and white image
            return lab2rgb_transpose(self.img_l, np.zeros((2, self.Xd, self.Xd)))


    class ColorizeImageCaffeGlobDist(ColorizeImageCaffe):
        # Caffe colorization, with additional global histogram as input
        def __init__(self, Xd=256):
            ColorizeImageCaffe.__init__(self, Xd)
            self.glob_mask_mult = 1.
            self.glob_layer = 'glob_ab_313_mask'

        def net_forward(self, input_ab, input_mask, glob_dist=-1):
            # glob_dist is 313 array, or -1
            if np.array(glob_dist).flatten()[0] == -1:  # run without this, zero it
out
                self.net.blobs[self.glob_layer].data[0, :-1, 0, 0] = 0.
                self.net.blobs[self.glob_layer].data[0, -1, 0, 0] = 0.
            else:  # run conditioned on global histogram
                self.net.blobs[self.glob_layer].data[0, :-1, 0, 0] = glob_dist
                self.net.blobs[self.glob_layer].data[0,     -1,     0,     0]     =
self.glob_mask_mult

            self.output_rgb   =   ColorizeImageCaffe.net_forward(self,   input_ab,
input_mask)
            self._set_out_ab_()
            return self.output_rgb


    class ColorizeImageCaffeDist(ColorizeImageCaffe):
```

```
        # caffe model which includes distribution prediction
        def __init__(self, Xd=256):
            ColorizeImageCaffe.__init__(self, Xd)
            self.dist_ab_set = False
            self.scale_S_layer = 'scale_S'
            self.dist_ab_S_layer = 'dist_ab_S'  # softened distribution layer
            self.pts_grid = np.load('./data/color_bins/pts_grid.npy')  # 529x2, all
points
            self.in_hull = np.load('./data/color_bins/in_hull.npy')  # 529 bool
            self.AB = self.pts_grid.shape[0]  # 529
            self.A = int(np.sqrt(self.AB))  # 23
            self.B = int(np.sqrt(self.AB))  # 23
            self.dist_ab_full = np.zeros((self.AB, self.Xd, self.Xd))
            self.dist_ab_grid = np.zeros((self.A, self.B, self.Xd, self.Xd))
            self.dist_entropy = np.zeros((self.Xd, self.Xd))

        def prep_net(self, gpu_id, prototxt_path='', caffemodel_path='', S=.2):
            ColorizeImageCaffe.prep_net(self,  gpu_id,  prototxt_path=prototxt_path,
caffemodel_path=caffemodel_path)
            self.S = S
            self.net.params[self.scale_S_layer][0].data[...] = S

        def net_forward(self, input_ab, input_mask):
            # INPUTS
            #     ab         2xXxX     input color patches (non-normalized)
            #     mask     1xXxX    input mask, indicating which points have been
provided
            # assumes self.img_l_mc has been set

            function_return   =   ColorizeImageCaffe.net_forward(self,   input_ab,
input_mask)
            if np.array(function_return).flatten()[0] == -1:  # errored out
                return -1

            # set distribution
            # in-gamut, CxXxX, C = 313
            self.dist_ab = self.net.blobs[self.dist_ab_S_layer].data[0, :, :, :]
            self.dist_ab_set = True

            # full grid, ABxXxX, AB = 529
            self.dist_ab_full[self.in_hull, :, :] = self.dist_ab

            # gridded, AxBxXxX, A = 23
            self.dist_ab_grid = self.dist_ab_full.reshape((self.A, self.B, self.Xd,
self.Xd))

            # return
            return function_return

        def get_ab_reccs(self, h, w, K=5, N=25000, return_conf=False):
            ''' Recommended colors at point (h,w)
            Call this after calling net_forward
            '''
            if not self.dist_ab_set:
                print('Need to set prediction first')
                return 0

            # randomly sample from pdf
            cmf = np.cumsum(self.dist_ab[:, h, w])  # CMF
            cmf = cmf / cmf[-1]
            cmf_bins = cmf

            # randomly sample N points
            rnd_pts = np.random.uniform(low=0, high=1.0, size=N)
```

```
            inds = np.digitize(rnd_pts, bins=cmf_bins)
            rnd_pts_ab = self.pts_in_hull[inds, :]

            # run k-means
            kmeans = KMeans(n_clusters=K).fit(rnd_pts_ab)

            # sort by cluster occupancy
            k_label_cnt = np.histogram(kmeans.labels_, np.arange(0, K + 1))[0]
            k_inds = np.argsort(k_label_cnt, axis=0)[::-1]

            cluster_per = 1. * k_label_cnt[k_inds] / N  # percentage of points within
cluster
            cluster_centers = kmeans.cluster_centers_[k_inds, :]  # cluster centers

            # cluster_centers = np.random.uniform(low=-100,high=100,size=(N,2))
            if return_conf:
                return cluster_centers, cluster_per
            else:
                return cluster_centers

        def compute_entropy(self):
            # compute the distribution entropy (really slow right now)
            self.dist_entropy = np.sum(self.dist_ab * np.log(self.dist_ab), axis=0)

        def plot_dist_grid(self, h, w):
            # Plots distribution at a given point
            plt.figure()
            plt.imshow(self.dist_ab_grid[:, :, h, w], extent=[-110, 110, 110, -110],
interpolation='nearest')
            plt.colorbar()
            plt.ylabel('a')
            plt.xlabel('b')

        def plot_dist_entropy(self):
            # Plots distribution at a given point
            plt.figure()
            plt.imshow(-self.dist_entropy, interpolation='nearest')
            plt.colorbar()
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ЦИФРОВОЇ ОБРОБКИ ФОТОГРАФІЇ

Виконав: студент 2 курсу, групи 1КН-22м

спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Данилишин В.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Колодний В.В.
(прізвище та ініціали)

«7.» 12 2023 р.

Вінниця ВНТУ – 2023 рік

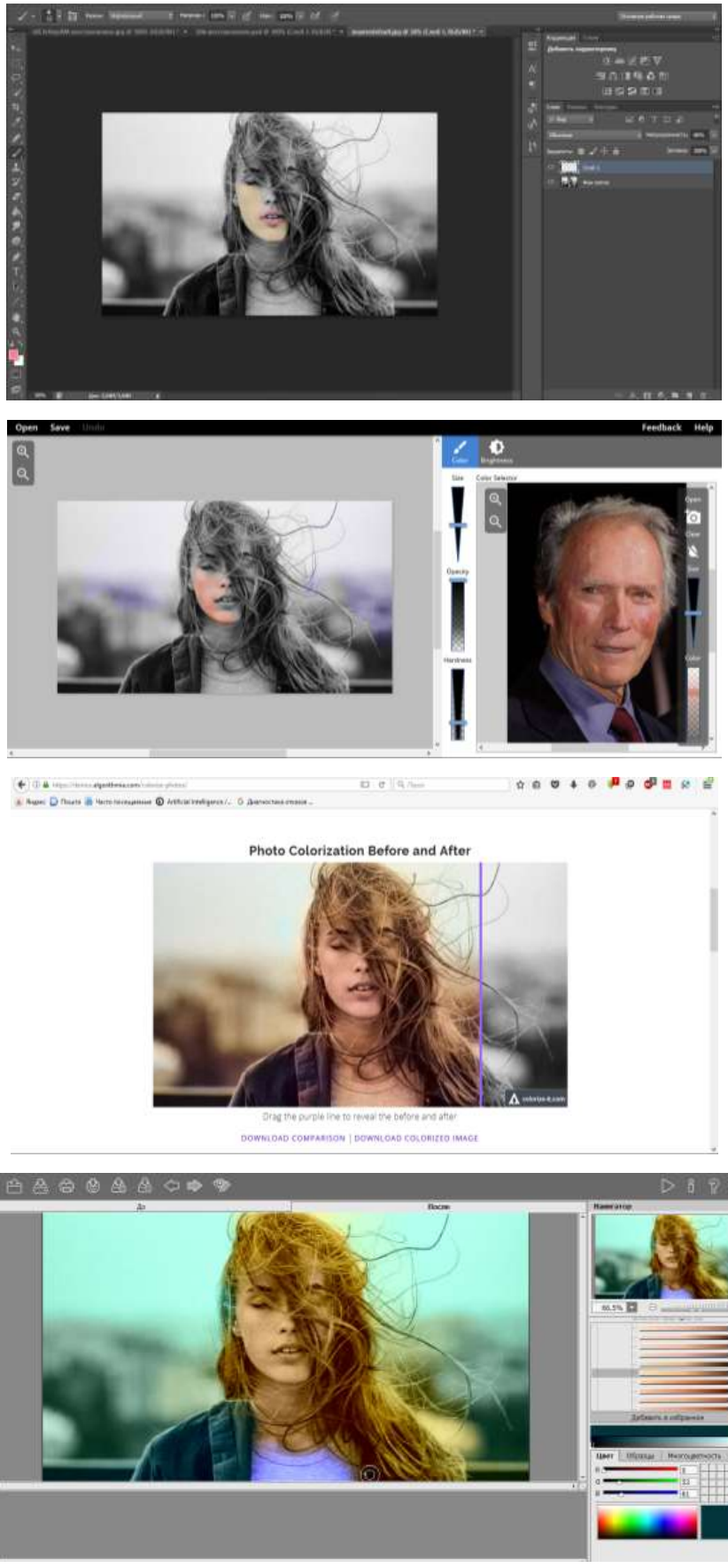

Рисунок В.1 – Порівняльний аналіз аналогів

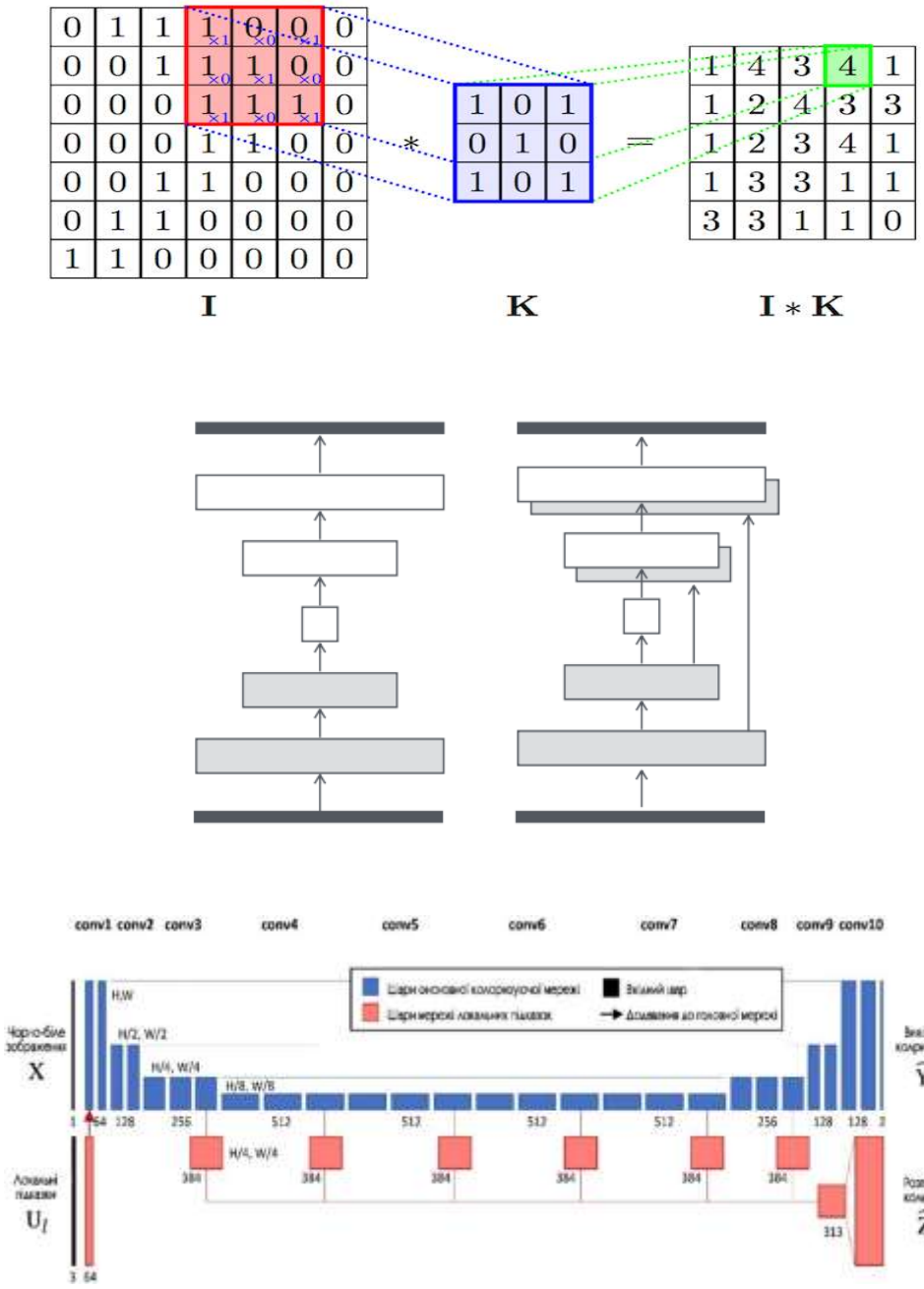

Рисунок В.2 – Обґрунтування вибору нейромережевого методу обробки зображень

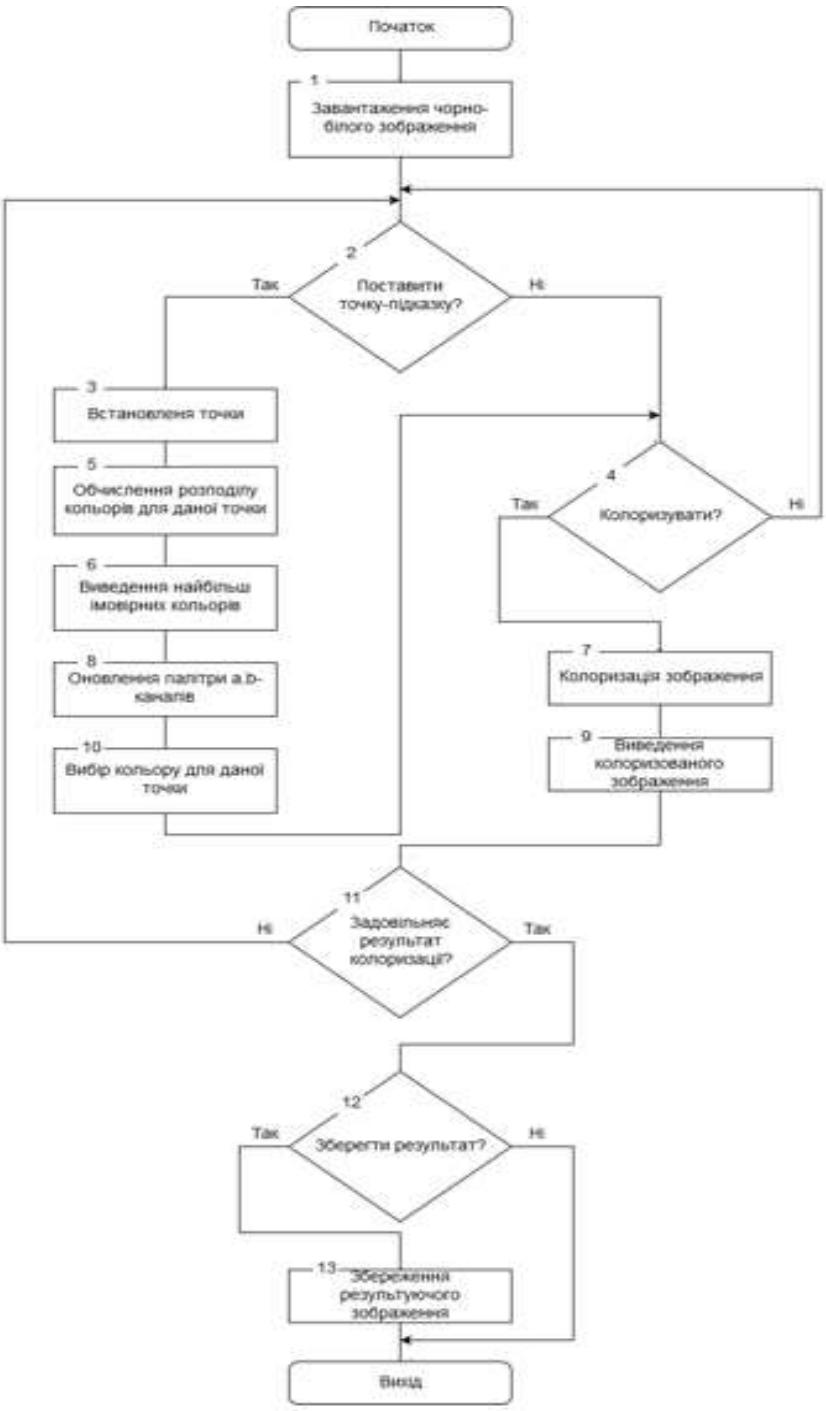

Рисунок В.3 – Алгоритм функціонування інформаційної технології цифрової обробки фотографії

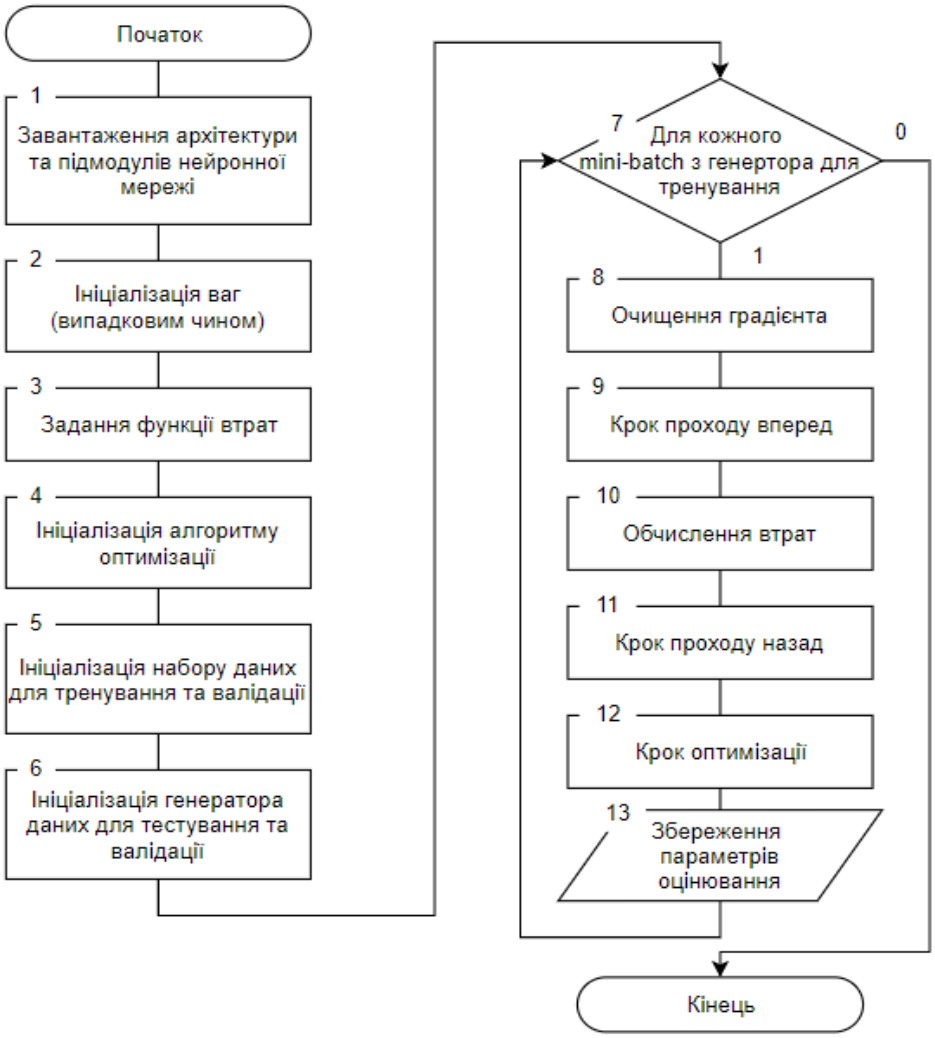

Рисунок В.4 – Алгоритм тренування й валідації нейронної мережі цифрової обробки фотографії

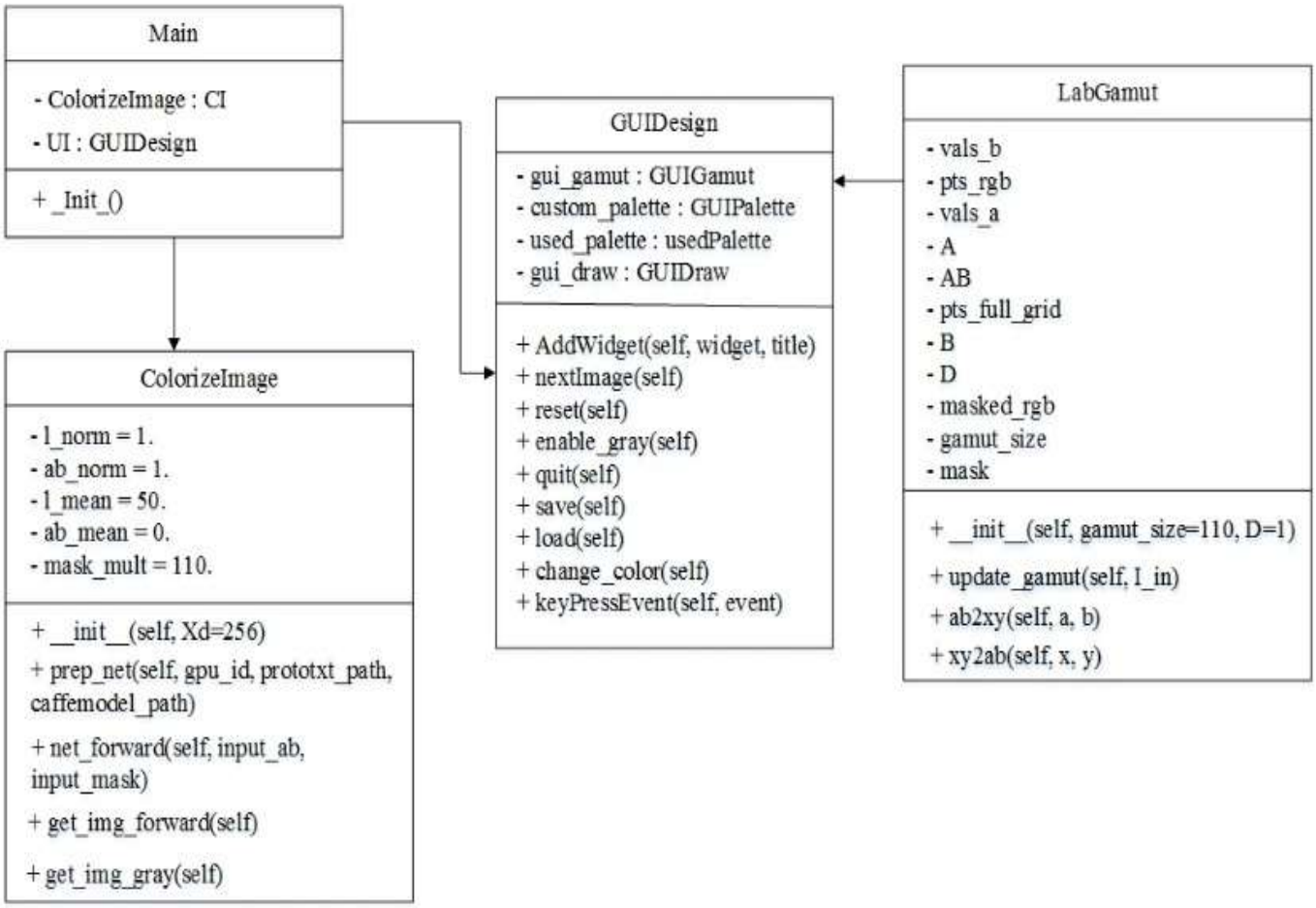

Рисунок В.5 – UML-діаграма класів програмного забезпечення цифрової обробки фотографій

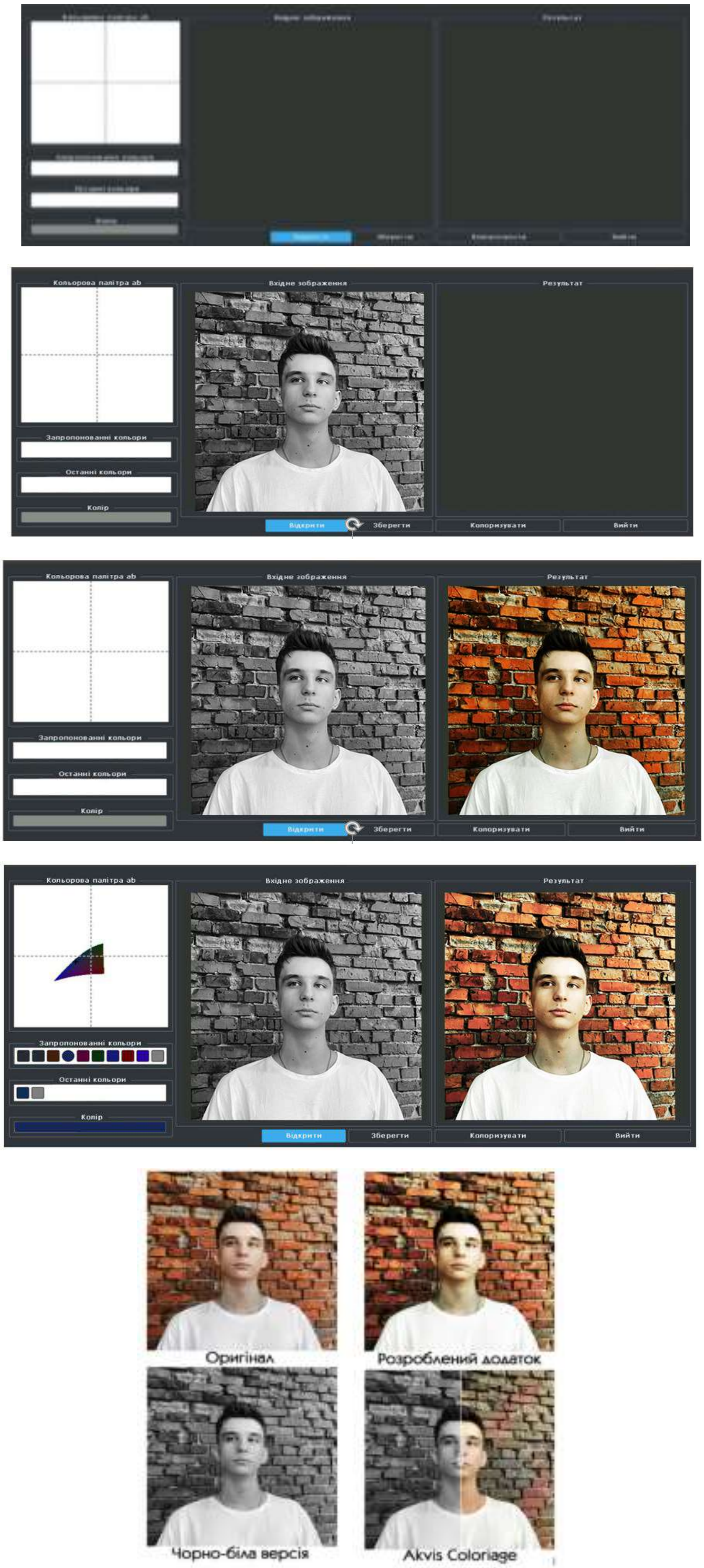

Рисунок В.6 – Інтерфейс програмного забезпечення цифрової обробки фотографії

Додаток Г (довідниковий)

Інструкція користувача

Після запуску програми відкривається головне вікно програми (рис. Г.1).

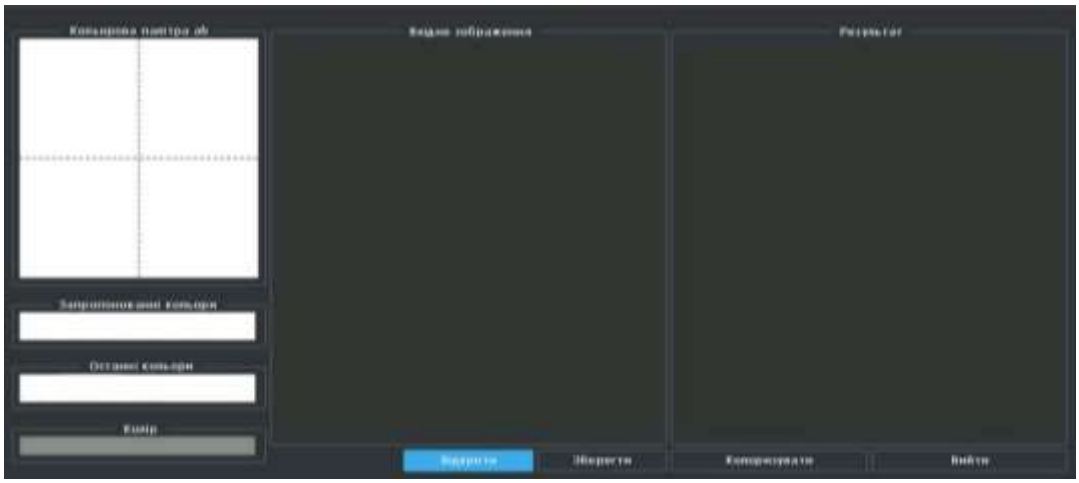

Рисунок Г.1 – Початкова активність

Аби почати роботу програми необхідно натиснути кнопку «Відкрити», щоб завантажити чорно-біле фото (рис. Г.2).

Рисунок Г.2 – Активність з відкритим чорно-білим зображенням

Після цього можна або автоматично колоризувати (рис. Г.3) або поставити точку-підказу і вибрати її колір, а після цього колоризувати (рис. Г.4).

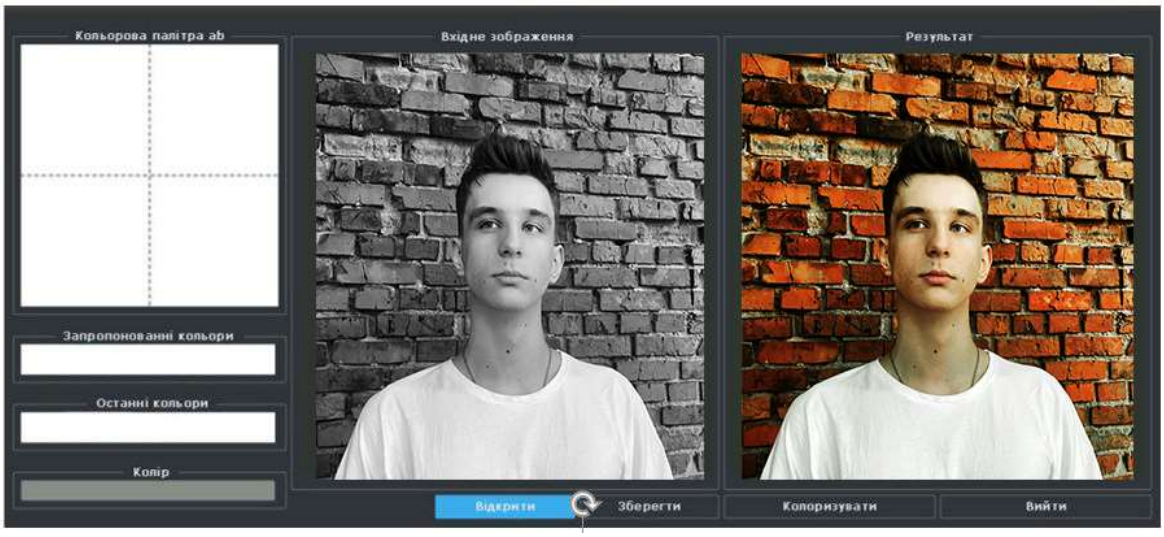

Рисунок Г.3 – Введення колоризованого зображення без підказок

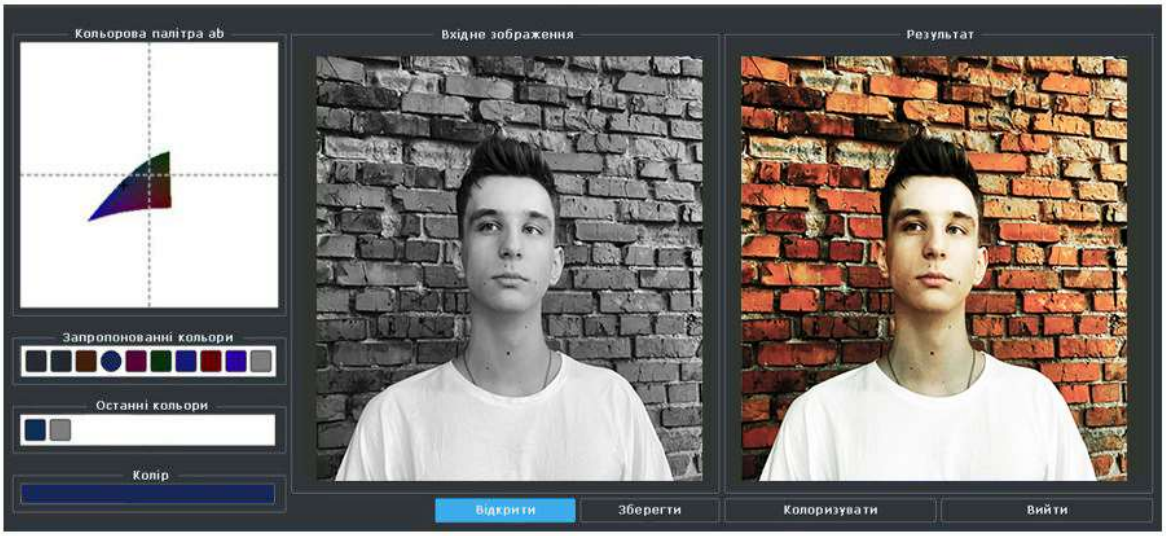

Рисунок Г.4 – Виведення колоризованого зображення з підказкою

Додаток Д (довідниковий)

Довідка про впровадження

МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ ____ від "__" _____ 20__ р.

ДОВІДКА

Дана Данилишину Владиславу Вікторовичу в тому, що результати, одержані ним в процесі виконання магістерської кваліфікаційної роботи, а саме алгоритми та програмні засоби цифрової обробки фотографії, планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» Бодяк В.М.

ТОВ "ІТІ"