Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія колоризації чорно-білих зображень»

Виконав: студент 2 курсу,
групи 1КН-18 м
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
Павлович Р.І.
Керівник: канд.техн.наук, доцент
Арсенюк І. Р.
Рецензент: канд.техн.наук, доцент
Катєльніков Д.І.

Вінниця, 2019 рік

ЗАТВЕРДЖУЮ
Завідувач кафедри____КН____
д.т.н., проф.. Яровий А.А.

_____
(підпис)
“____” _____2019 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.014.18.000.ПЗ

Магістранта групи 1КН-18м Павловича Романа Ігоровича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія колоризації чорно-білих зображень»

Вхідні дані: зображення людини у форматі jpeg, роздільна здатність зображення не більше 1920*1080, кількість класів не менше 4, мова програмування об’єктно-орієнтована.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: схема роботи алгоритму програмного засобу колоризації чорно-білих зображень; структура нейронної мережі колоризації чорно-білих зображень; структура інформаційної технології колоризації чорно-білих зображень; UML-діаграма класів програмного засобу колоризації чорно-білих зображень; головне вікно програмного засобу колоризації чорно-білих зображень; приклад роботи програмного засобу колоризації чорно-білих зображень.

2. Текстова (пояснювальна записка): вступ, аналіз предметної області інформаційної технології колоризації чорно-білих зображень, розробка інформаційної технології колоризації чорно-білих зображень, програмна реалізація інформаційної технології колоризації чорно-білих зображень, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного рівня розвитку інформаційних технологій колоризації чорно-білих зображень. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Розробка математичної моделі та інформаційної технології колоризації чорно-білих зображень			Математична модель, інформаційна технологія, розділ 2 ПЗ
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка ефективності			Програмне забезпечення, розділ 3 ПЗ
4	Підготовка економічної частини			Розділ 4 ПЗ
5	Апробація та/або впровадження результатів дослідження			Тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

"____" _____20___ р. І. Р. Арсенюк
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц., доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)

М. В. Бальзан
ініціали та прізвище

"____" _____20___ р.

Дата попереднього захисту роботи "____" _____20___ р.

Рецензент _____ канд. техн. наук, доц., доц. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)

Д.І. Катєльніков
ініціали та прізвище

Завдання видав
науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

І. Р. Арсенюк
ініціали та прізвище

"____" _____20___ р.

Завдання отримав магістрант _____ Р.І. Павлович
(підпис) ініціали та прізвище

"____" _____20___ р.

РЕФЕРАТ

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології колоризації чорно-білих зображень на основі згорткової нейронної мережі. Дана технологія забезпечує автоматизовану колоризацію чорно-білих зображень та дає змогу користувачу керувати колоризацією.

В ході роботи проведено аналіз предметної області колоризації чорно-білих зображень. Розглянуто аналоги для колоризації чорно-білих зображень. Удосконалено структуру нейронної мережі таким чином, щоб покращити якість колоризації чорно-білих зображень. Розроблено інформаційну технологію колоризації чорно-білих зображень на основі згорткової нейронної мережі та реалізовано на мові програмування Python програмний засіб для колоризації чорно-білих зображень на основі згорткової нейронної мережі.

ABSTRACT

The master's qualification work is dedicated to the development of the information technology of the grayscale images colorization based on a convolutional neural network. This information technology provides automatic grayscale images colorization and provides a user to manage colorization.

During the work was analyzed of the subject area of the grayscale images colorization. Known existing analogs for grayscale images colorization were considered. The neural network structure was improved to increase the quality of grayscale images colorization. An information technology of grayscale images colorization based on convolutional neural network was developed and a software tool for grayscale images colorization based on convolutional neural network was implemented using Python programming language.

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.

Колоризація – 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 колоризації. Робота полягала в перемальовуванні старих кадрів і їх ручному розфарбовуванні.

До появи цифрових технологій колоризацію 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.

У цій галузі з часом були отримані технології, що полегшують це. Розробка автоматичної технології сегментації, велика потужність комп'ютерів та додаткових компонентів, що використовуються для колоризації.

Згодом стали з’являтися все нові і нові методи автоматизованої колоризації зображень. Мета таких методів – автоматично визначити не тільки межі монохромного сегменту, але і його колір. Перевага такого підходу очевидна – наприклад, на колоризацію фотографії не потрібно витрачати багато часу, як

раніше, але, на жаль, якість існуючих алгоритмів тоді ще була далеко не ідеальною.

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

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

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

Для досягнення поставленої мети слід розв’язати такі завдання:

- розглянути та проаналізувати існуючі програмні реалізації розв’язання задачі колоризації чорно-білих зображень;
- запропонувати математичну модель для інформаційної технології колоризації чорно-білих зображень;
- навести стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології колоризації чорно-білих зображень;

- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес колоризації чорно-білих зображень.

Предмет дослідження – інформаційна технологія колоризації чорно-білих зображень.

Метою дослідження магістерської кваліфікаційної роботи є підвищення якості колоризації чорно-білих зображень.

Область застосування – колоризація чорно-білих фотографій, рекольоризація кольорових зображень.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрової інформації, теорія штучних нейронних мереж для реалізації інформаційної технології колоризації 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 архітектуру згорткової нейронної мережі, що відрізняється від відомих, додаванням підмережі, яка використовується для локальних кольорових підказок, що забезпечило підвищення якості колоризації чорно-білих зображень.

Практичне значення одержаних результатів:

1. Розроблено алгоритм колоризації чорно-білих зображень з використанням нейронних мереж.
2. Розроблено структуру нейронної мережі для підвищення точності колоризації чорно-білих зображень на основі згорткових нейронних мереж.
3. Розроблено програмний продукт колоризації чорно-білих зображень з використанням нейронних мереж.

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, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати досліджень апробовані на XLVII науково-технічній конференції підрозділів ВНТУ, XI міжнародній науково-практичній конференції «ІОН-2018», XII міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2019».

Публікації. За результатами досліджень подано свідоцтво про реєстрацію авторського права на твір – комп'ютерна програма «Нейромережевий модуль колоризації чорно-білих досліджень»[4], опубліковано троє тез доповідей міжнародих науково-практичних конференцій[1,2,3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ КОЛОРИЗАЦІЇ ЧОРНО-БІЛИХ ЗОБРАЖЕНЬ

1.1 Аналіз задачі колоризації чорно-білих зображень

У даній кваліфікаційній роботі вирішується задача колоризації 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 колоризації. Додавання кольору до старого фото може дати більше розуміння того, що зображено на фото. Також колоризацію використовують різні дослідники історії, які працюють з різними архівними зображеннями, тому задача колоризації – актуальна в наш час.

Як зазначено раніше, колоризацію проводили вручну до розповсюдження персональних комп'ютерів. З розвитком комп’ютерних технологій колоризація проводилася за допомогою простих графічних редакторів. Однак складність цієї процедури без використання автоматизованого програмного забезпечення дуже висока. Друга причина дослідження – підвищення якості колоризації 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.

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 Аналітичний огляд аналогів

Колоризація 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 колоризації 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 рисунку 1.2.

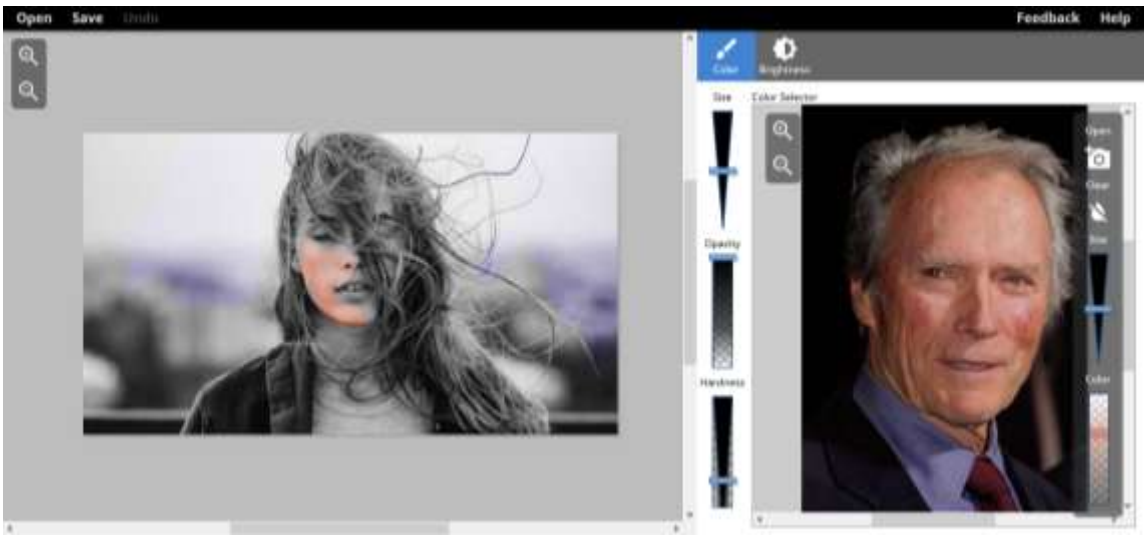

Рисунок 1.2 – вікно веб-додатку ColorizePhoto

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. 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.

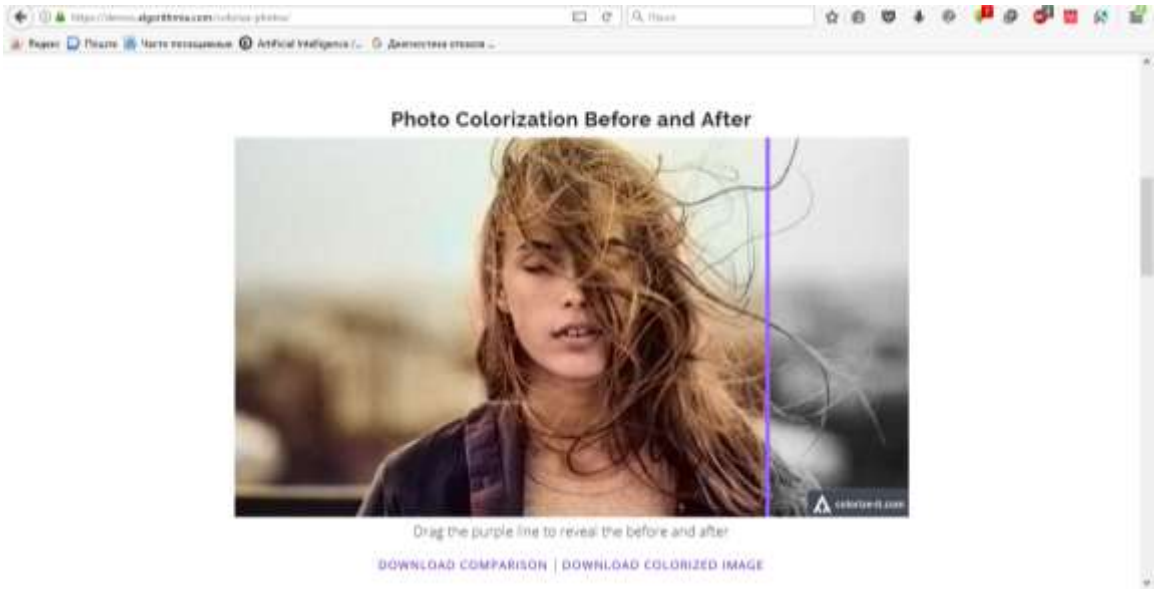

Рисунок 1.3 – вікно колоризації Algorithmia Colorize Photos

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 відскановані 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 може працювати як окрема програма (standalone) або як плагін (plugin) до графічного редактора. Ця програма, на мій погляд, найкраща в даному переліку. Вона має досить багато переваг над іншими. Але все ж якість колоризації не задовільняє сучасні потреби.

Рисунок 1.4 – Приклад колоризації в AKVIS Coloriage

1.3 Постановка задачі

У ході виконання дипломного проектування 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 представити у вигляді матриці G з H рядків і W стовпців. Числами цієї матриці є інтенсивності відповідних пікселів зображення. Як правило інтенсивністю є число від 0 до 1, де 0 означає повну відсутність, чорний колір, а 1 – повну присутність, білий колір. На практиці для цього застосовують цілі або дійсні числа будь-якої розмірності. Чим вона вища, тим більше число відтінків сірого можна уявити. 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

$$F: R^{W*H} \rightarrow R^{3*W*H} \tag{1.1}$$

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

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 колоризації виконано постановку задачі, виконання якої призведе підвищення якості колоризації зображень.

2 РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОЛОРИЗАЦІЇ ЧОРНО-БІЛИХ ЗОБРАЖЕНЬ

2.1 Обґрунтування використання методу обробки зображень в задачі колоризації чорно-білих зображень

Завдання колоризації має підзадачу виділення сегментів чорно-білих зображень з рівним кольоровим тоном. Ця підзадача може бути інтерпретована як завдання пошуку розпізнавання зображень. Існує три основні групи методів розпізнавання:

1. Статистичні.
2. Структурні.
3. Нейромережеві.

Статистичний підхід ґрунтується на математичних правилах класифікації, які формулюються та виводяться з точки зору математичної статистики. Цей метод забезпечує отримання класифікатора, коли відомі щільності розподілу для всіх наборів образів та ймовірність появи образів для кожного класу. У розпізнаванні образів невідомий об'єкт для класифікації представлений як вектор елементарних ознак.

У структурному розпізнаванні символів сутність представляється у вигляді сукупності елементарних частин, їх атрибутів і відношень поряд з глобальними ознаками сутнтості. Ключові моменти цього підходу – це відбір непохідних елементів образу, інтеграція цих елементів та пов'язані з ними зв’язки у граматиці образів і, відповідно, реалізація відповідною мовою процесів аналізу та розпізнавання.

Перспективною альтернативою традиційним методам вирішення проблем розпізнавання зображень є нейронні мережі. Ця сфера активно розвивається сьогодні. Області застосування нейронних мереж зростають, з’являються нові моделі нейронних мереж, існуючі моделі адаптуються для вирішення нових завдань тощо. Нейронна мережа – математична модель, а також її програмне або

апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалока і У. Піттса [12].

Основні переваги нейронних мереж.

1. Розв’язання задач при невідомих закономірностях. Використовуючи вміння вчитися на багатьох прикладах, нейронна мережа здатна вирішити проблеми, в яких закономірності ситуації та взаємозв'язок між входом і виходом невідомі. Традиційні математичні методи та експертні системи в таких випадках не є прийнятними.

2. Стійкість до шумів у вхідних даних. Можливість працювати за наявності великої кількості неінформативних, вхідних сигналів шуму. Не потрібно робити їх попередній відсів, сама нейронна мережа визначить їх непридатними для виконання завдання та відхилить їх.

3. Адаптація до змін навколишнього середовища. Нейронні мережі мають можливість адаптуватися до змін навколишнього середовища. Зокрема, нейронні мережі, які навчаються працювати в певному середовищі, можуть бути легко перекваліфіковані для роботи в умовах незначних коливань параметрів навколишнього середовища. Більше того, у нестаціонарному середовищі (де статистика змінюється з часом) можуть створюватися нейронні мережі, які навчаються в режимі реального часу. Чим вище адаптаційні можливості системи, тим стабільнішою вона буде в нестаціонарному середовищі. Слід зазначити, що адаптивність не завжди призводить до стійкості, іноді це призводить до прямо протилежного результату. Наприклад, адаптивна система, де швидко змінюються параметри, також може швидко реагувати на сторонні збудження, спричиняючи втрату продуктивності. Для того, щоб повною мірою скористатися пристосованістю, основні параметри системи повинні бути достатньо

стабільними, щоб уникнути врахування зовнішніх перешкод та достатньо гнучкими, щоб реагувати на значні зміни зовнішнього середовища [13].

4. Потенційно надвисока швидкодія. Нейронні мережі мають потенціал для високошвидкісної продуктивності завдяки використанню масової паралельної обробки інформації [14].

5. Відмовостійкість при апаратній реалізації нейронної мережі. Нейронні мережі є потенційно відмовостійкими. Це означає, що при несприятливих умовах їх продуктивність незначно знижується. Наприклад, якщо нейрон або його зв’язок пошкоджено, отримання збереженої інформації є складним. Однак, враховуючи розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що лише серйозні пошкодження структури нейронної мережі суттєво вплинуть на її продуктивність. Тому зниження якості нейронної мережі відбувається повільно [15].

Для реалізації інформаційної технології для колоризації чорно-білих зображень було обрано метод, що базується на нейронній мережі.

Нейронні мережі являють собою систему з'єднаних і взаємодіючих між собою нейронів. Такі нейрони зазвичай досить прості. Кожен нейрон такої мережі взаємодіє тільки з сигналами, які він періодично отримує або посилає іншим нейронам. І, тим не менше, будучи з'єднаними 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 – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в пошуку вагових коефіцієнтів для зв'язків між нейронами. При навчанні нейронна мережа здатна знаходити складні зв’язки між входом і виходом, а також виробляти узагальнення. Іншими словами, в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних або частково перекручених даних.

Структурно нейронна мережа представляє з себе безліч нейронів, об'єднаних в шари. Кожен нейрон представляє з себе одиницю, яка одержує дані з безлічі різних входів з попереднього шару по так званим синапсах. У нейроні рахується зважена сума S всіх входів, а на вихід подається значення функції активації

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

Ця величина і вважається значенням аксона [16]. Будову нейрона показано на рисунку 2.1.

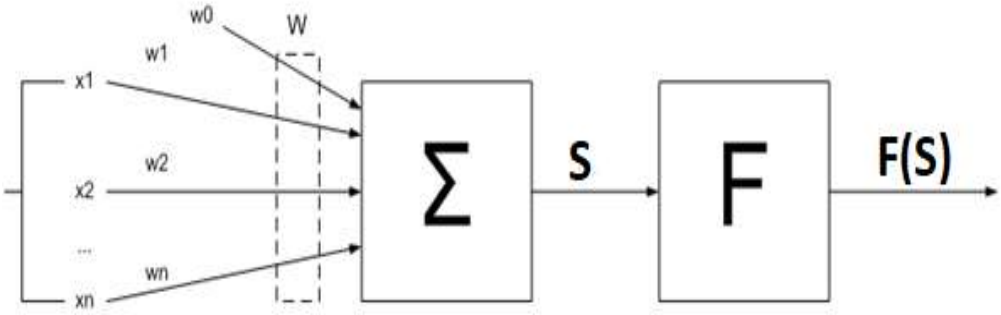

Рисунок 2.1 – Будова нейрона нейронної мережі

Структуру шарів нейронної мережі зображено на рисунку 2.2.

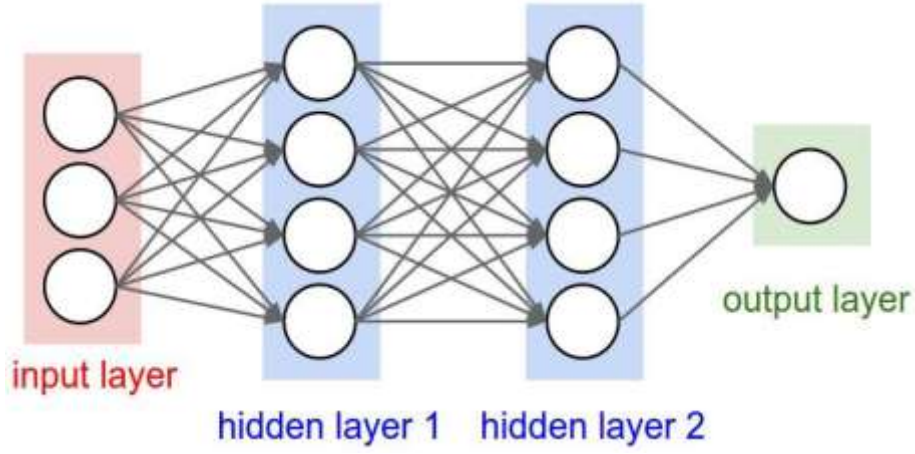

Рисунок 2.2 – Структура шарів нейронної мережі

Функцію активації можна сприймати як будь-яку функцію, але часто використовується лише декілька, оскільки існує ряд вимог, які свідчать про те, що їх продуктивність позитивно впливає на навчання нейронної мережі:

- Нелінійність – якщо функція активації нелінійна, то можна показати, що двошарова нейронна мережа є так званим універсальним аппроксиматором функції. В іншому випадку це не має сенсу для багатьох шарів, оскільки лінійна комбінація лінійних функцій є лінійною функцією. Вище було сказано, що завдання колоризації може бути сформульована як задача пошуку складного відображення F. Здатність нейронних мереж будувати універсальні апроксимації є однією з причин використання їх в даній роботі для вирішення поставленого завдання [17].
- Безперервна диференційованість – ця властивість просто необхідно для можливості застосування для навчання нейронної мережі методів оптимізації, заснованих на обчисленні градієнтів [18].
- Обмеженість – методи оптимізації схильні до нестабільності при навчанні нейронних мереж, коли функції активації мережі є необмеженими.
- Монотонність – доведено, що якщо функція активації є монотонною, то функція помилок на кожному конкретному шарі опукла, що значно збільшує можливості вирішення задач оптимізації, оскільки проблеми опуклої оптимізації досліджуються набагато краще [19].
- Монотонність похідної – показано, що функції активації з цією властивістю узагальнюються значно краще [17].
- Апроксимація тотожного відображення в околі нуля – якщо функція активації володіє цією властивістю, нейронна мережа буде ефективно навчатися, якщо її ваги спочатку ініціалізуються малими випадковими вагами. В іншому випадку потрібно провести більше досліджень, щоб зрозуміти, які вихідні варіанти потрібно для початку навчання [19].

2.2 Розробка моделі нейронної мережі інформаційної технології колоризації чорно-білих зображень

2.2.1 Обгрунтування вибору типу нейронної мережі для роботи з зображеннями

Класичні нейронні мережі використовували тільки повнозв'язні шари. Для невеликих завдань такий підхід підходить, але при роботі з зображеннями кількість параметрів у такій мережі величезна навіть для двох шарів. Тому для вирішення задачі колоризації чорно-білих зображень була обрана згорткова нейронна мережа.

Згорткова нейронна мережа – це клас глибинних штучних нейронних мереж прямого поширення, створена Яном Лекуном для ефективного розв'язання задач розпізнавання зображень[20].

Назва мережевої архітектури була сформована тому, що вона базується на операції згортання, суть якої полягає в тому, що кожен піксель зображення множиться на матрицю згортання (ядро), а результат підсумовується і відправляється у відповідне положення оригінального зображення.

Робота згорткової нейронної мережі зазвичай трактується як перехід від більш простих особливостей зображення до більш абстрактних, виділяючи поняття вищого рівня. У цьому випадку мережа самоналаштовується та виробляє необхідну ієрархію абстрактних функцій (послідовності карт характеристик), фільтруючи незначні деталі та виділяючи важливі.

Таке тлумачення є доволі метафоричним чи ілюстративним. Насправді функції, що створюються складною мережею, незрозумілі та важко інтерпретувати таким чином, що в практичних системах не рекомендується намагатися зрозуміти зміст цих функцій або намагатися їх налаштувати вручну, а натомість потрібно удосконалити структуру та архітектуру мережі для кращих результатів.

У повністю пов'язаній нейронній мережі кожен нейрон кожного шару з'єднаний з кожним нейроном попереднього шару, і кожному з'єднанню присвоюється особистий ваговий коефіцієнт. У згорткової нейронної мережі в операції згортання використовується обмежена невелика матриця ваг, яка відображається у кожному положенні обробленого шару і генерує сигнал активації для нейрона наступного шару у відповідному положенні після кожного зсуву. Тобто для різних нейронів у вихідному шарі використовується однакова вагова матриця, яка також називається ядром згортки. Це трактується як графічне кодування будь-якого знаку, наприклад, наявність похилої лінії під певним кутом. Потім наступний шар, отриманий в результаті операції згортання такої матриці ваг, показує наявність цієї ознаки в обробленому шарі та його координатах, утворюючи так звану карту ознак. В згортковій нейронній мережі набір ваг – це не один елемент, а цілий набір елементів, що кодують зображення (наприклад лінії і дуги під різними кутами). При цьому такі ядра згортки не закладаються дослідником заздалегідь, а формуються самостійно, навчаючи мережу класичним методом зворотного поширення помилки. Прохід кожним з наборів ваг формує власний примірник карти ознак, роблячи нейронну мережу багатоканальною. При перегляді шару матрицею ваг її пересувають зазвичай не на розмір цієї матриці, а на меншу відстань. Так, наприклад, при розмірності матриці ваг 4×4 її зрушують на один або два пікселя замість п'яти. Приклад операції згортки зображено на рисунку 2.3. В даному прикладі згортковий фільтр має розмірність 3×3 [21].

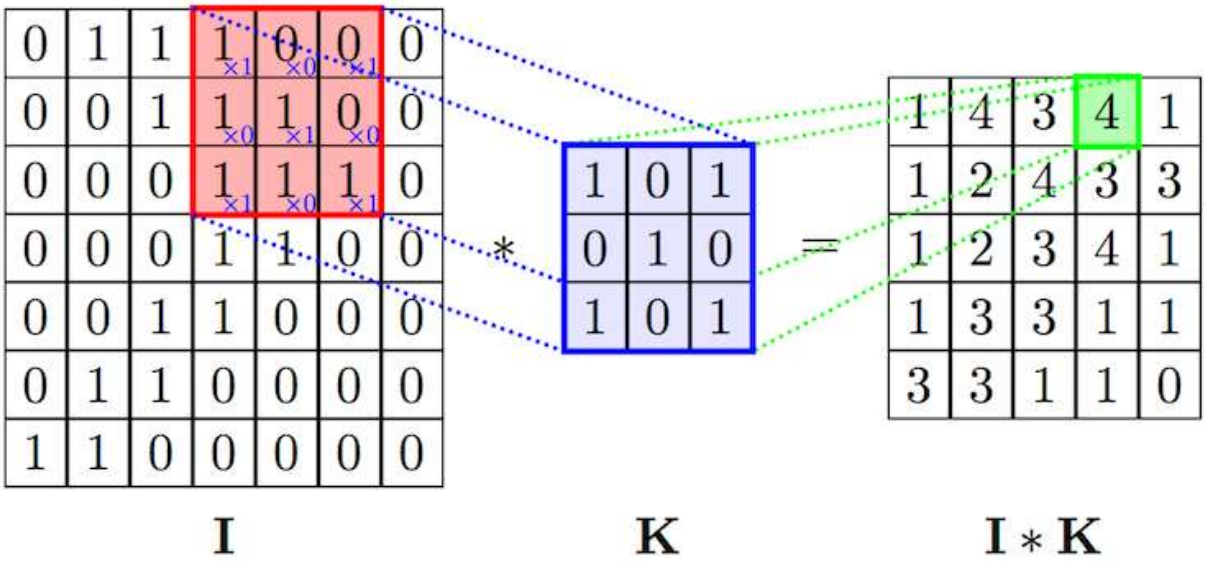

Рисунок 2.3 – Приклад операції згортки

Операція субдискретизації, або пулінга, виконує зменшення розмірності отриманих від згорткового шару карт ознак. У даній архітектурі мережі вважається, що інформація про наявність шуканої ознаки є більш вагомою, ніж точне знання її координат, тому з кількох сусідніх значень карти ознак вибирається одне – максимальне, і приймається за одне зі значень більш щільної карти ознак меншої розмірності. Завдяки цій операції, крім прискорення подальших обчислень, мережа знаходить інваріантність до масштабу вхідного зображення [22].

Розглянемо типову структуру згорткової нейронної мережі більш докладно. Мережа складається з багатьох шарів. Після початкового шару (вхідного зображення) сигнал проходить низку згорткових шарів, в яких чергується власне згортка і субдискретизація (пулінг). Чергування шарів дозволяє будувати більш складні функції, оскільки глибина мережі зростає, з кожним наступним шаром карта зменшується в розмірах, але кількість каналів збільшується. Найчастіше, пройшовши декілька шарів, карта карт стає векторною або навіть скалярною, але таких карт ознак є сотні. Як правило, на виході згорткових шарів мережі додатково встановлюють кілька шарів повнозв'язної нейронної мережі, на вхід якого подаються кінцеві карти ознак.

Шар згортки є основним блоком згорткової нейронної мережі. Шар згортки включає в себе для кожного каналу свій фільтр, ядро згортки яке обробляє попередній шар по фрагментах, підсумовуючи результати матричного добутку для кожного фрагмента. Для ядра згортки вагові коефіцієнти спочатку невідомі і встановлюються в процесі навчання [21].

Особливістю згорткового шару є порівняно невелика кількість параметрів, яка встановлюється при навчанні. Так наприклад, якщо вихідне зображення має розмірність 100×100 пікселів по трьох каналах (це значить 30000 вхідних нейронів), а згортковий шар використовує фільтри з ядром 3×3 пікселя з виходом на 6 каналів, тоді в процесі навчання визначається тільки 9 ваг ядра, однак за всіма сполученням каналів, тобто $3 \times 3 \times 3 \times 6 = 162$, в такому випадку даний шар вимагає знаходження тільки 162 параметрів, що істотно менше кількості шуканих параметрів повнозв'язної нейронної мережі, яка мала б порядок 100 мільйонів.

Шар субдискретизації – це нелінійне ущільнення карти ознак, при цьому група пікселів (зазвичай розміру 2×2) ущільнюється до одного пікселя, проходячи нелінійне перетворення. Найчастіше використовується при цьому функція максимуму (рисунку 2.4) [23]. Перетворення зачіпають непересічні прямокутники або квадрати, кожен з яких скорочується в один піксель, при цьому вибирається піксель, що має максимальне значення. Операція пулінга дозволяє істотно зменшити просторовий обсяг зображення. Пулінг тлумачиться так: якщо деякі ознаки вже були виявлені в попередній операції згортання, то таке детальне зображення більше не потрібно для подальшої обробки і ущільнюється до менш деталізованого. Крім того, фільтрування зайвих деталей допомагає не перенавчатися. Шар пулінга зазвичай вставляється після шару згортки перед наступним шаром згортки.

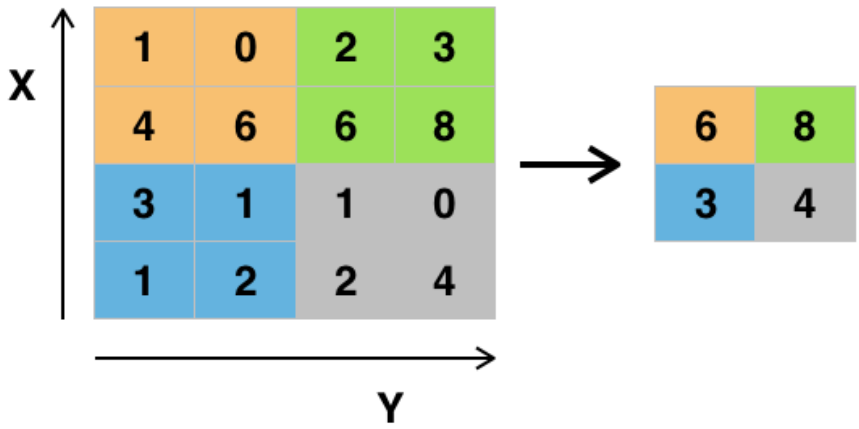

Рисунок 2.4 – Операція субдискретизації з кроком 2 і розміром вікна 2 × 2

Особливо примітною є операція так званої транспонованої згортки чи деконволюції. У чомусь ця операція є протилежною операції згортання. При застосуванні операції згортання на крок, більший за одиницю, отримане зображення має менший розмір, тоді як при застосуванні транспонованої операції згортки розмір результату буде більшим [24]. На рисунку 2.5 зліва показаний приклад роботи звичайної операції згортки, а праворуч – транспонованою.

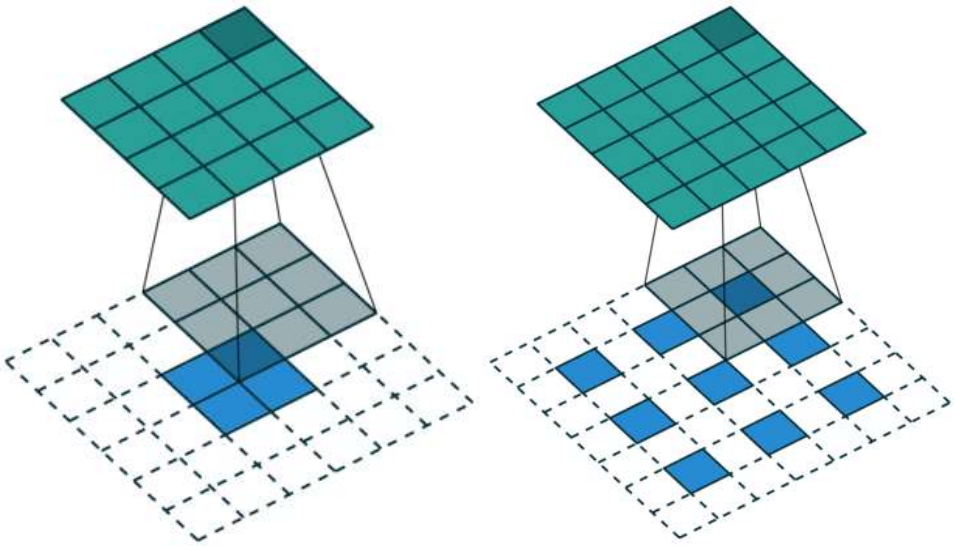

Рисунок 2.5 – Приклад операції згортки і транспонованої згортки

На практиці операція транспонованої згортки часто застосовується для відновлення вихідної розмірності зображення після декількох застосувань шарів звичайної згортки або субдискретизації. Особливо часто цей прийом використовується в генеративних моделях, де з вихідного зображення нейронна мережа повинна генерувати інше.

2.2.2 Вибір парадигми навчання нейронної мережі

Правило навчання або алгоритм навчання – це метод або математична модель, що підвищує продуктивність штучної нейронної мережі і, як правило, це правило застосовується неодноразово по всій мережі.

На сьогодні відомо три парадигми навчання нейронних мереж, в основу яких покладено особливості машинного навчання:

- навчання з вчителем (supervised learning);
- навчання без вчителя (unsupervised);
- навчання з підкріпленням (reinforcement leaming).

Для згорткових нейронних мереж найкраще використовувати навчання з вчителем, що було підтверджено в більшості наукових досліджень і альтернативи по суті відсутні. Навчання з вчителем – передбачає, що для кожного вхідного вектора існує вектор вихідних значень. Разом ці два вектора називають навчальною парою, а множину навчальних пар – навчальною вибіркою. Процес навчання зводиться до почергового подавання на вхід нейронної мережі навчальних пар, вираховування похибки між дійсним і бажаним значенням нейронної та корегування параметрів мережі в бік зменшення цієї похибки.

Продуктинвість роботи нейронної мережі може бути виміряна функцією втрат, яка і оптимізується при навчанні мережі. Якщо функція втрат обрана добре, то загалом, чим менше її значення, тим краще мережа навчилася вирішувати своє завдання. Таким чином, задача навчання нейронної мережі – по

суті, задача мінімізації деякої пов'язаної з цією мережею функції втрат, тобто вона є задачею оптимізації. Тому будь-який метод оптимізації можна використовувати з тим чи іншим успіхом [25].

Нейронні мережі були винайдені ще на початку XX століття, але застосовуватися почали лише нещодавно. Причина цього є винахід методу зворотного поширення помилки [26]. Ця подія зробила революцію в машинному навчанні, оскільки ефективність цього методу значно вища, ніж ефективність усіх методів, які були винайдені раніше. Тому цей метод був обраний для нашої нейронної мережі.

Основна ідея цього методу – розповсюдження сигналів помилок з мережевих виходів на його входи – у напрямку, протилежному прямому розповсюдженню сигналів у звичайному режимі. При кожній ітерації алгоритму поширення помилок обчислюється функція втрат і змінюються ваги нейронної мережі для покращення рішення на отриманих прикладах.

Для того, щоб з'ясувати, яким чином змінювати ваги, також обчислюється градієнт функції втрат за всіма параметрами мережі, тому для можливості застосування методу функція активації нейронів повинна бути диференційована, а її похідні повинні бути відомі. Метод є модифікацією класичного методу градієнтного спуску [20].

Мінімізація функції втрат – складна проблема: параметрів дуже багато (для стандартних прикладів, що реалізуються на ПК їх число доходить до мільйонів), адаптивний рельєф (графік оцінки як функції від корегуємих параметрів) складний, може містити багато локальних мінімумів.

Щоб зрозуміти алгоритм зворотного поширення помилки розглянемо ще раз будову нейронної мережі. Без втрати спільності вважаємо, 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 отримаєм після деякого 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, і на виході мережі буде отримано вектор

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

Таким чином, нейронну мережа можна представити у вигляді композиції функцій і кожен новий шар додає в неї ще одну функцію [18].

Нехай θ – вектор параметрів нейронної мережі. При навчанні для обчислення градієнта функції втрат, потрібно обчислення градієнта функції F. Для обчислення градієнта $\nabla\theta F(x)$ потрібно скористатися правилом диференціювання складної функції [27]:

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

В тому випадку, коли похідні функцій активації є відомими, кожна з похідних легко може бути обчислена. При проході через мережу одного об'єкта першими зможуть бути 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 [28].

До появи методів колоризації на основі нейроних мереж було винайдено багато інших підходів. Один з фундаментальних підходів, на який посилаються багато робіт, і який надихнув багато інших методів, – 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 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 близька. Формально можна сказати, що для кожного пікселя необхідно мінімізувати різницю між каналами $a(r)$ і $b(r)$ і зваженими середніми кольорів його сусідніх пікселів. Випишемо 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;

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

Такий самий метод використаємо в даній магістерській кваліфікаційній роботі.

2.2.3 Вибір способу оптимізації навчання нейронної мережі для колоризації чорно-білих зображень

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 [27]. 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 малюнку 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.

Однією з модифікацій методу стохастичного градієнтного спуску є метод ADAM [29]. 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

оптимуму, піддаючись коливанням менше, ніж їм піддається метод стохастичного градієнтного спуску.

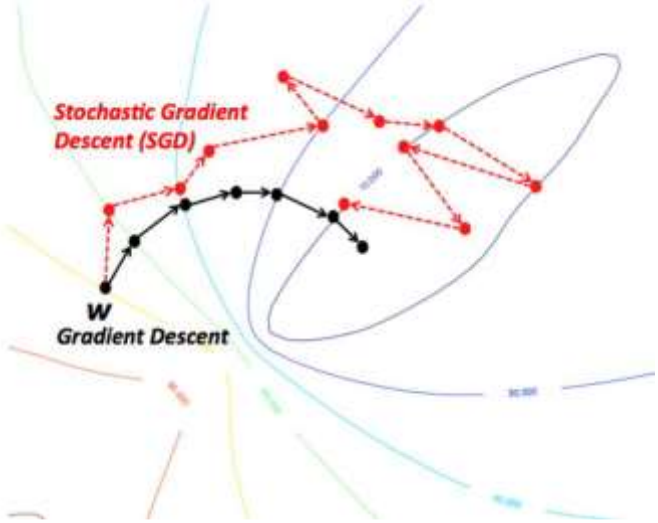

Рисунок 2.6 – Приклад стохастичного і звичайного гардієнтного спуску

ADAM добре зарекомендував себе на практиці і використовується в більшості сучасних нейронних мереж. З цієї причини він буде використаний у цій роботі.

2.2.4 Обгрунтування вибору методу для вирішення проблеми перенавчання

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 [30].

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.

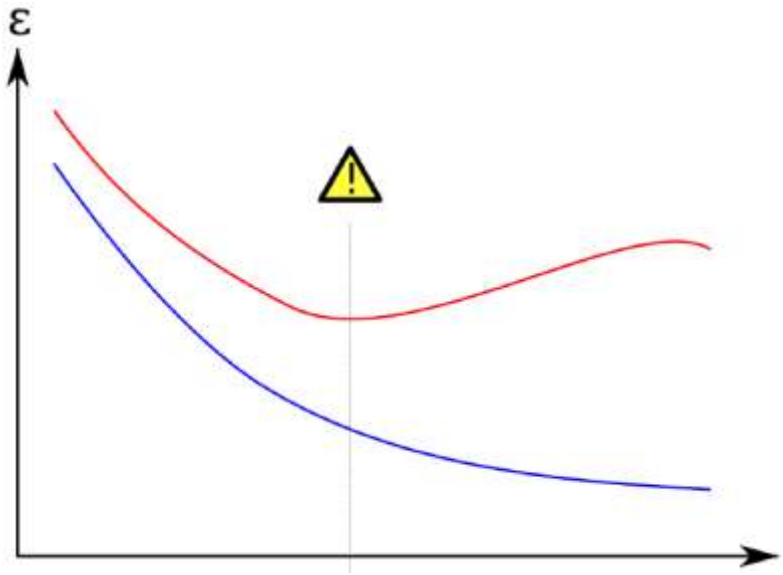

Рисунок 2.7 – Приклад перенавчання

Боротьба з перенавчанням – одна з відкритих 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 [30]. На практиці часто реалізується в такий спосіб. Замість вирішення завдання

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

Вирішується задача

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

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.

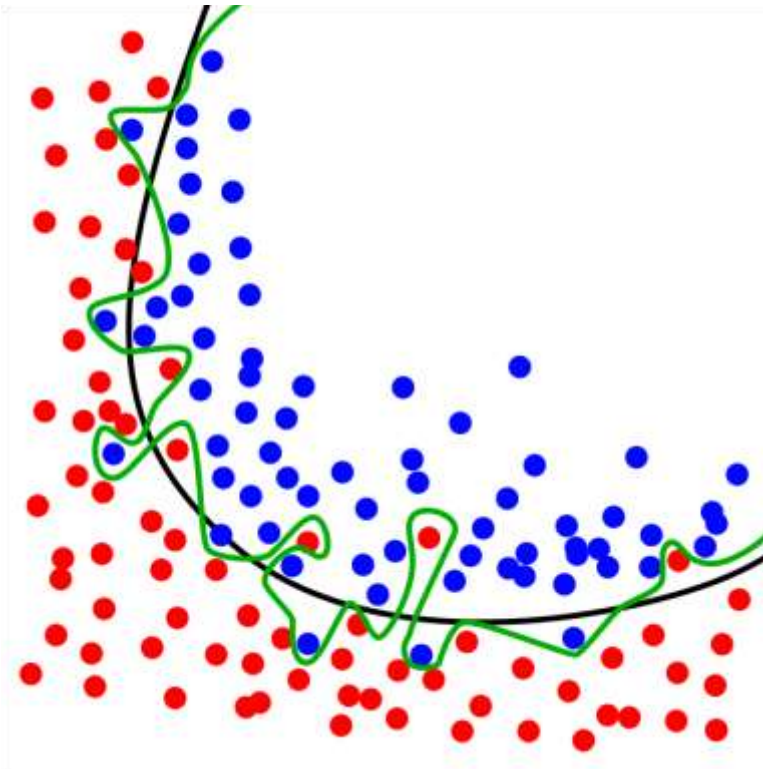

Рисунок 2.8 – приклад надто складної моделі

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 [31].

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.

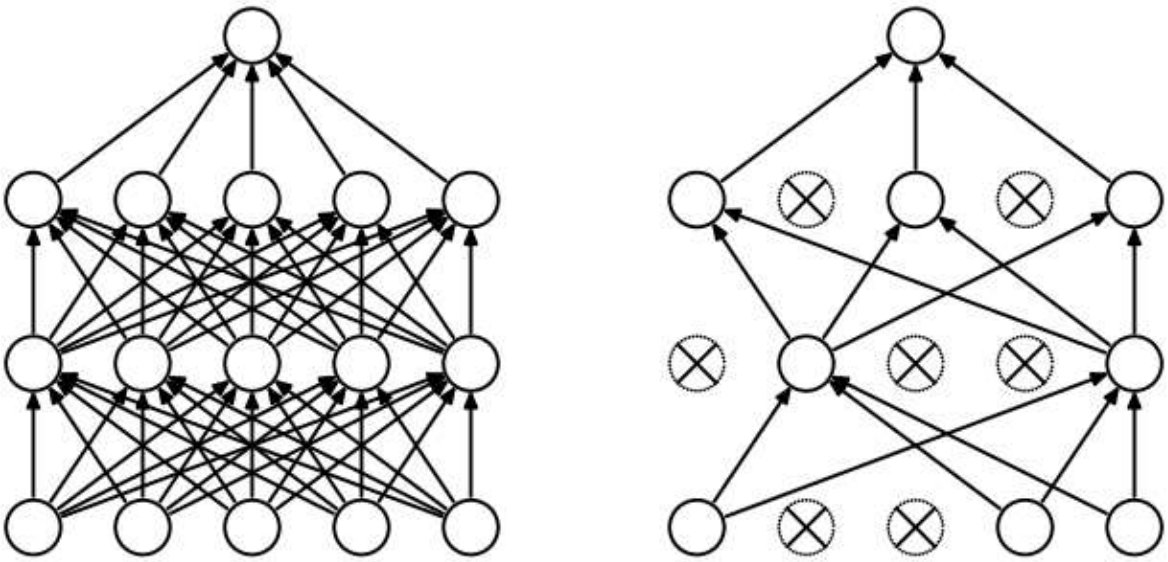

Рисунок 2.9 – Приклад методу виключення

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.

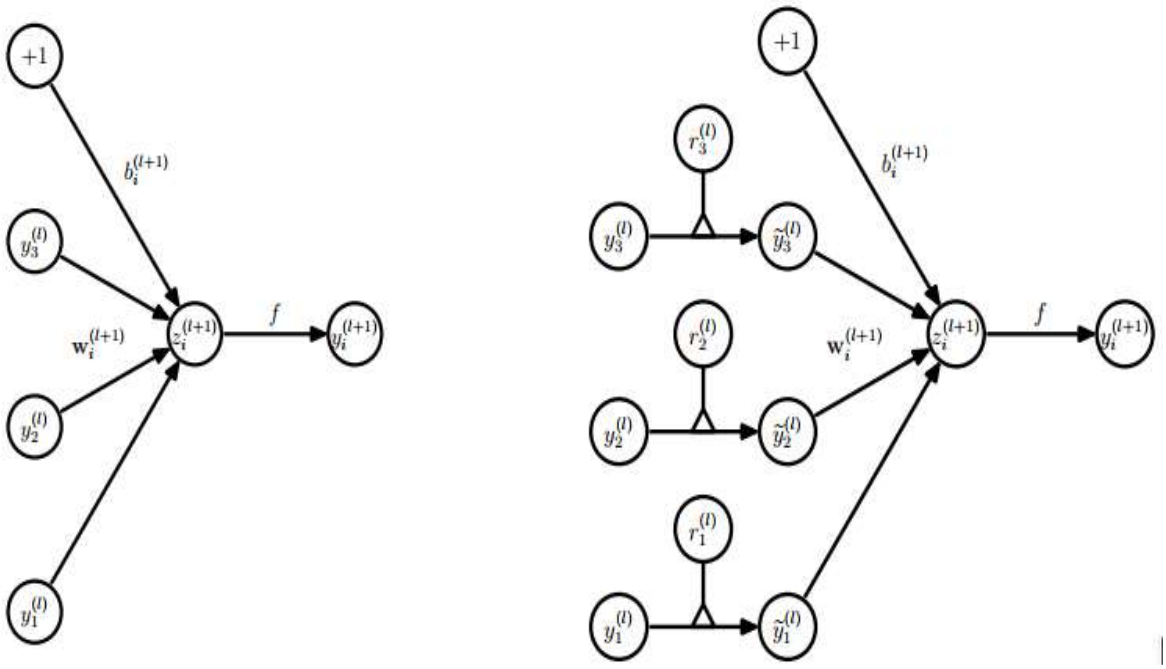

Рисунок 2.10 – Приклад ваг нейронної мережі при застосуванні методу виключень

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.

2.2.5 Обґрунтування використання батчевої нормалізації

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 [33].

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, $X = \big(x\,(1), \ldots, x\,(d)\big)$. Усередині батчу порахуємо для кожного параметра середнє і дисперсію по всіх об'єктах [34]. Після чого просто припустимо, що

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

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 [34].

В результаті якість навчання підвищується істотно. 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.

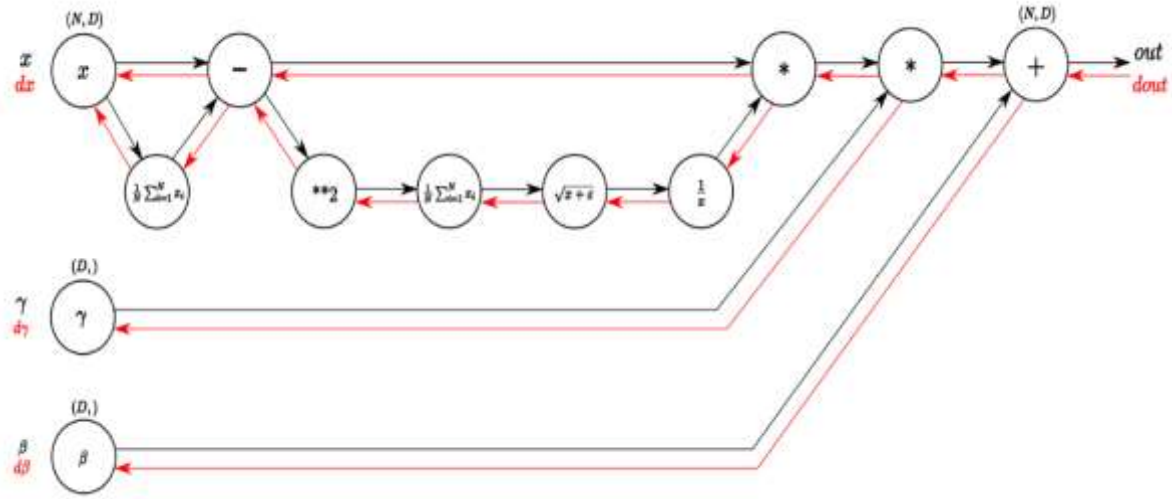

Рисунок 2.11 – Приклад графу обчислення операцій батчевої нормалізації

2.3 Вибір колірної моделі для вирішення задачі колоризації чорно-білих зображень

Колірна модель – математична модель для опису подання кольорів у вигляді наборів чисел (зазвичай з трьох, рідше – чотирьох значень), що називаються кольоровими компонентами або кольоровими координатами. Усі можливі значення кольорів, задані моделлю, визначають кольоровий простір [31].

Як буде показано нижче, спосіб завдання кольору істотно впливає на рішення поставленої задачі. Наприклад, стандартна для більшості додатків модель RGB підходить набагато менше, ніж деякі менш поширені моделі.

2.3.1 Аналіз колірної моделі RGB

RGB (red, green, blue) – адитивна кольорова модель. Вибір кольорів зумовлено особливостями будови і сприйняття людського ока. Адитивною вона

є тому, що задає кольори шляхом їх додавання до чорного кольору [32]. Це можна бачити на малюнку 2.12.

Рисунок 2.12 – Адитивність моделі RGB

Можна уявити колір у вигляді трійки чисел, кожне з яких знаходиться в інтервалі від 0 до 1, тому весь колірний простір можна представити у вигляді куба $1 * 1 * 1$, як показано на рисунку 2.13.

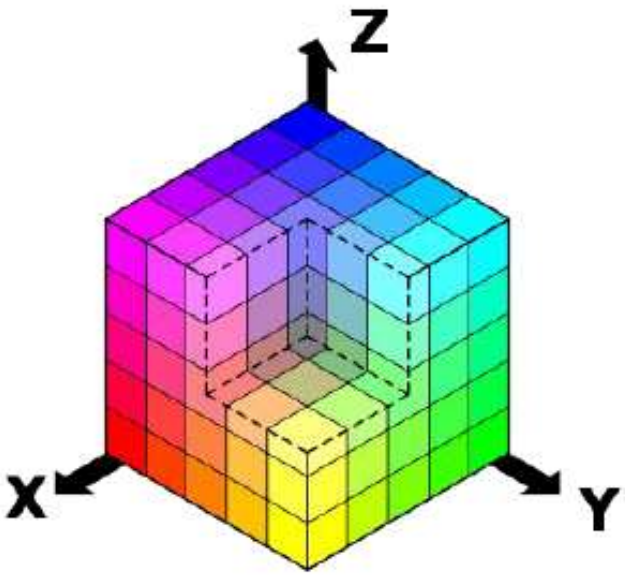

Рисунок 2.13 – Представлення моделі RGB в вигляді одиничного куба

Недоліки RGB-моделі:

1. Залежність від обладнання: Колір, що виникає в результаті змішування кольорових компонентів елемента RGB, залежить від типу люмінофора. У технології виробництва сучасних кінескопів використовуються різні типи люмінофорів (з різними спектральними характеристиками), а встановлення однакових інтенсивності електронних пучків у випадку різних люмінофорів призведе до синтезу різних кольорів.

2. Обмеження гами. Колірний обхват (color gamut) – це діапазон кольорів, який може розрізняти людина або відтворювати пристрій незалежно від механізму отримання кольору. Обмеження гамми полягає в тому, що неможливо отримати всі кольори видимого спектру за допомогою адитивного синтезу. Зокрема, деякі кольори, такі як чисто синій або чисто жовтий, можуть не відображатися точно на екрані.

2.3.2 Аналіз колірної моделі HSV

HSV (Hue, Saturation, Value) − колірна модель, в якій координатами кольору є:

Hue − колірний тон. Варіюється в межах від 0 до 360 градусів, іноді може бути представлений у вигляді діапазону від 0 до 1 або до 100;

Saturation − насиченість. Варіюється в межах від 0 до 100 або від 0 до 1. Чим більше цей параметр, тим «чистіше» колір, тому його іноді називають чистотою кольору. А чим ближче цей параметр до нуля, тим ближче колір до нейтрального сірого;

Value (значення кольору) або Brightness − яскравість. Також задається в межах від 0 до 100 або від 0 до 1 [33].

Модель HSV представляють в циліндричній системі координат або у вигляді конуса, що показано на рис 2.14.

Недолік колірної моделі HSV: так само як і в попередній колірній моделі – обмежений колірний простір.

Переваги:

1. Апаратна незалежність.
2. Більш простий і інтуїтивно зрозумілий механізм управління кольором.

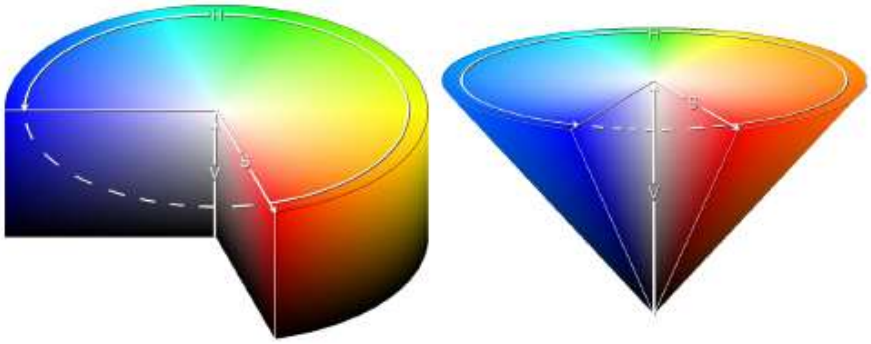

Рисунок 2.14 − Візуалізація колірного простору HSV

2.3.3 Аналіз колірної моделі CIELAB

Колірна модель CIELAB (Lab) розроблялася з цілю створення світлового простору, зміна кольору в якому буде лінійною відносно до людського сприйняття. Тобто з метою, щоб однакова зміна значень координати кольору в різних частинах колірного простору викликало відчуття однакової зміни кольору [34]. Використання такої моделі коректувало би нелінійність сприйняття кольору людиною. L-канал (lightness, яскравість) несе таке ж значення, що і канал Y в моделі YUV, координата a відповідає положенню кольору від зеленого до червоного, а координата b − від синього до жовтого. Це можна бачити на рисунку 2.15. Перетворення відбувається з моделі XYZ – еталонної колірної моделі, заданої в строгому математичному сенсі організацією CIE (International Commission on Illumination, Міжнародна комісія з освітлення) в 1931 році, за такими формулами:

$$f(t) = \begin{cases} t^{1/3}, & t > (6/29)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \end{cases}, \tag{2.14}$$

$$L^* = 116 f(Y/Y_n) - 16, \tag{2.15}$$

$$a^* = 500[f(X/X_n) - f(Y/Y_n)], \tag{2.16}$$

$$b^* = 200[f(Y/Y_n) - f(Z/Z_n)]. \tag{2.17}$$

На відміну від багатьох інших моделей, відображення кольору в яких залежить від апаратних даних, CIELAB визначає колір однозначно. Тому CIELAB знайшов широке застосування в програмному забезпеченні для обробки зображень.

Колірна модель CIE Lab, якщо порівнювати з RGB, дозволяє більш точно визначити колір. Тому, CIE Lab є абсолютною моделлю, а RGB відносною. При будь-якому перетворенні зображення з однієї колірної моделі в іншу спочатку виконується внутрішнє перетворення в модель CIE Lab. Великою перевагою моделі CIE Lab є її незалежність від пристрою, і той факт, що її колірна гамма є найбільшою (отже, містить широкий діапазон кольорів), а також, повна незалежність яскравості L від колірних компонентів a, b.

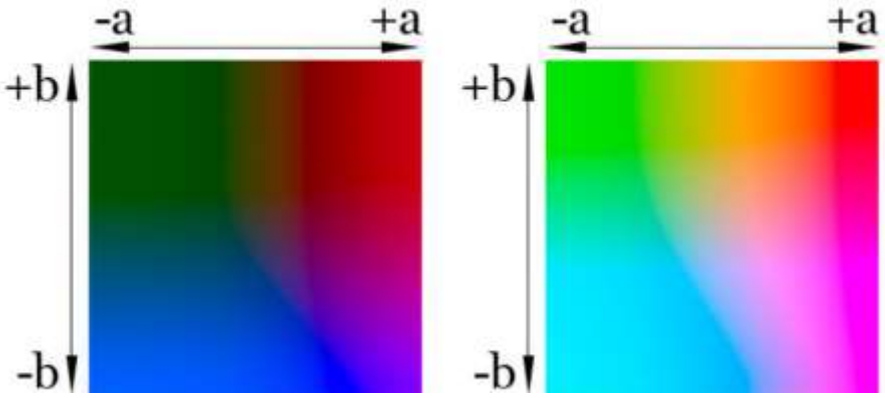

Рисунок 2.15 − Залежність кольору в просторі CIELAB в залежності від значень a і b, при L рівному 0.25 і 0.75

Переваги CIELAB – моделі [35]:

1. Апаратна незалежність;

2. Більший колірний обхват в порівнянні з RGB- і HSV- моделями;

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

Через наведенні вище особливості ми будемо використовувати цю модель для додання кольорових підказок до чорно-білого зображення.

2.4 Розробка структури нейронної мережі для інформаційної технології колоризації чорно-білого зображення

Класичною архітектурою для колоризуючих мереж є структура за типом encoder-decoder – кілька згорткових шарів поспіль, а потім – відповідні їм шари транспонованою згортки [36]. Така архітектура дозволяє мережі розпізнавати складні шаблони аж до самої вузької її частини. Ця половина мережі називається "encoder".

Друга половина мережі робить протилежне – за отриманим поданням генерує необхідний вихід. У нашому випадку – кольорове зображення. Ця частина мережі називається "decoder".

Але стосовно до задачі колоризації цього недостатньо. У такій архітектурі втрачається безліч низькорівневої інформації, наприклад, про границі або об'єкти більш складної форми. Замість цього запропоновано новий підхід до побудови такої архітектури, названий U-net [36]. Його особливістю є те, що шари декодера отримують дані не тільки з попереднього шару, але і з відповідного йому шару енкодера. На рисунку 2.16 показано, яким чином влаштовані зв'язки в мережах типу U-net.

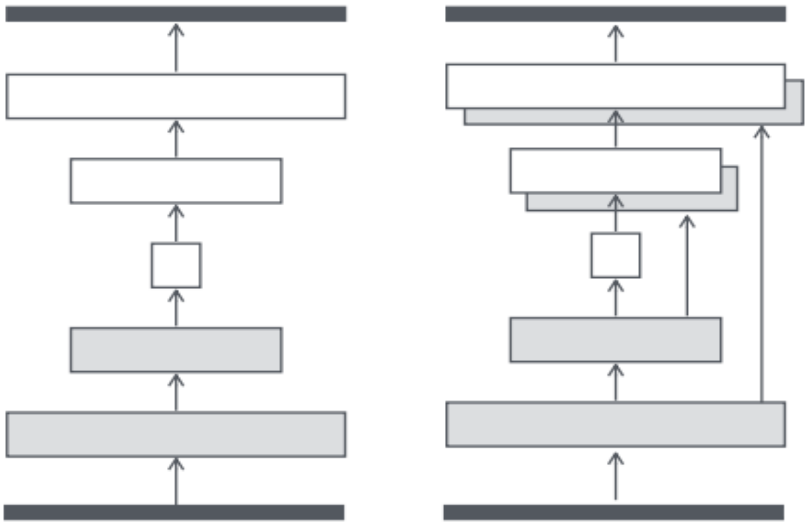

Рисунок 2.16 – Приклад структури мережі encoder-decoder і U-net

Також, як описано вище, на якість навчання значно впливає використання шарів нормалізації. Тут вони використовуються після кожного згорткового шару.

Окремо варто згадати L_1 і L_2 регуляризації. Вище було сказано, що їх використання не доцільне, а все тому, що їх застосування призводить до генерації розмитих зображень з нечіткими краями і плавно мінливими кольорами. З цієї причини регуляризація проводиться тільки в формі батчевої нормалізації і винятків.

Також перевагу в продуктивності дає генерація зображень не в просторі RGB, а в просторі Lab, оскільки канал L за по суті є чорно-білим зображенням. Моделі в цьому випадку потрібно генерувати тільки два канали замість трьох. Розглядалось також використання простору HSV, використовуючи значення чорно-білого зображення в якості каналу V, але було досліджено, що це згубно вплине на якість навчання моделей. При S = 0 значення H-каналу ніяк не впливає на результуюче зображення.

Недоліком більшості моделей, які виконують колоризацію автоматично, є неможливість користувача впливати на результат певним чином, навіть при наявності апріорних знань про кольори в деяких частинах зображення. Це призводить до того, що колір, наприклад одяг, часто відрізняється від оригіналу.

Так само за наявності таких знань багато різних кольорів можуть виглядати однаково правдоподібно. Тому в даній магістерській кваліфікаційній роботі буде використовуватись спосіб забарвлення з допомогою розповсюдження користувацьких підказок.

Архітектура розробленої, в рамках дипломного проектування, згорткової нейронної мережі типу U-net, модифікованої за рахунок підмережі користувацьких підказок, зображено на рисунку 2.17.

Вхідними даними є чорно-біле зображення(L-канал в колірній моделі CIE Lab), маска з нулів і одиниць, де одиниці відповідають пікселям, в яких присутня призначена користувачем підказка, і два канали a і b – самі підказки.

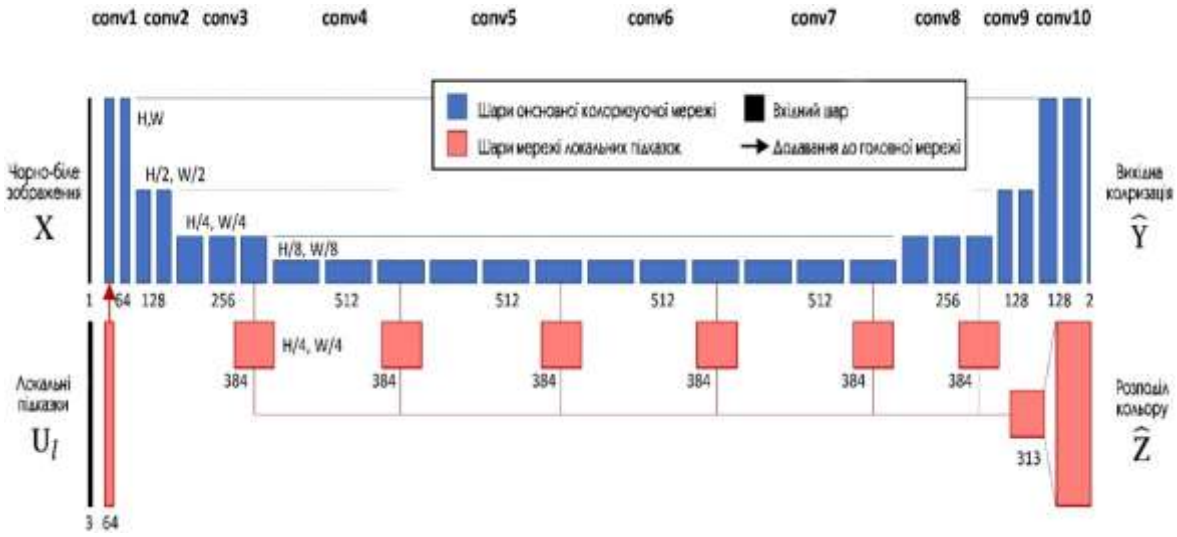

Рисунок 2.17 – Структура нейронної мережі інформаційної технології колоризації чорно-білих зображень

На відміну від класичних методів, разом з каналами a і b мережа також генерує розподіл кольорів для кожного пікселя, який потім використовується в інтерфейсі.

Нейронна мережа складається з двох частин: основна колоризуюча мережа архітектури U-net (10 згорткових блоків) і мережа локальних підказок.

2.5 Розробка схеми алгоритму функціонування інформаційної технології колоризації чорно-білих зображень

Розробимо схему загального алгоритму функціонування інформаційної технології колоризації чорно-білого зображення. Запропонована схема зображена на рисунку 2.18.

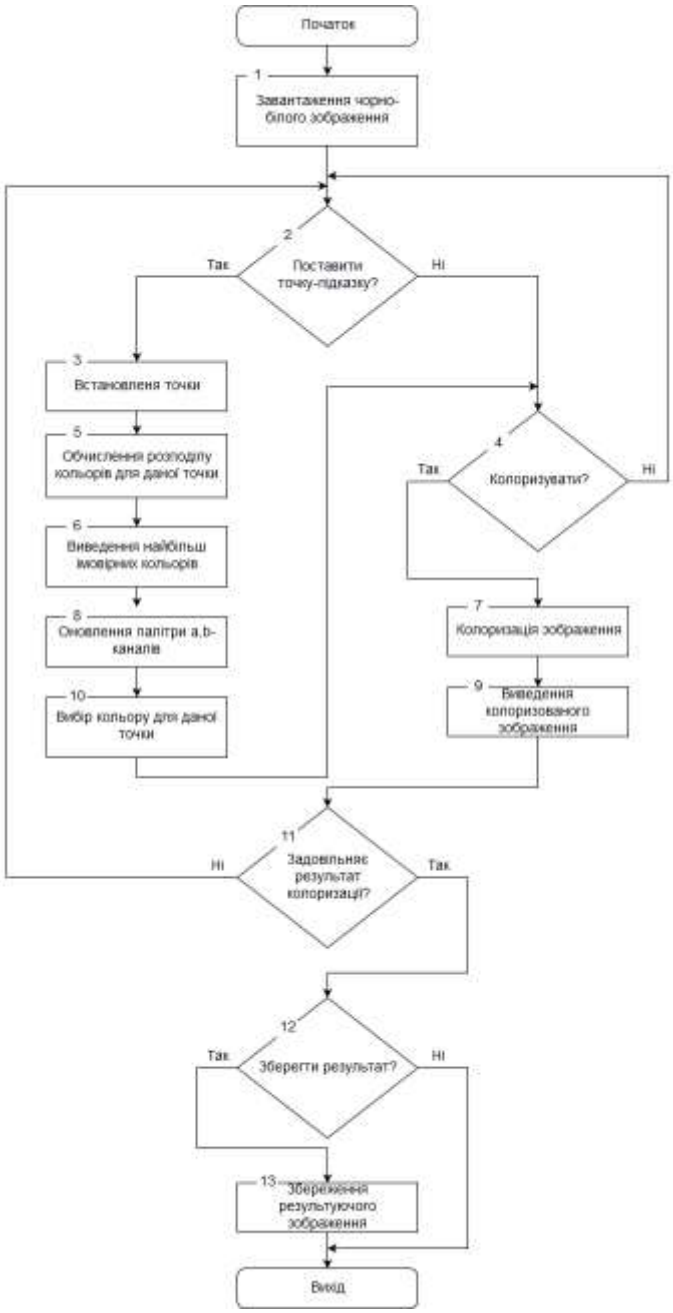

Рисунок 2.18 – Схема загального алгоритму функціонування інформаційної технології колоризації чорно-білих зображень

2.6 Розробка структури інформаційної технології колоризації чорно-білих зображень за допомогою згорткових нейронних мереж

На вхід подається зображення у форматі JPEG, PNG а також матриця користувацьких підказок. Після цього відбувається зміна масштабу зображення до заданого у налаштуваннях нейронної мережі. Значення кожного пікселя перетворюється так, щоб їх діапазон був від 0 до 100. Після цього зображення потрапляє на вхід нейронної мережі і відбувається процес прямого поширення. Наступним є процес обробки вихідних даних та виведення отриманого результату.

Структура інформаційної технології колоризації чорно-білих зображень з використанням нейронних мереж зображено на рисунку 2.19.

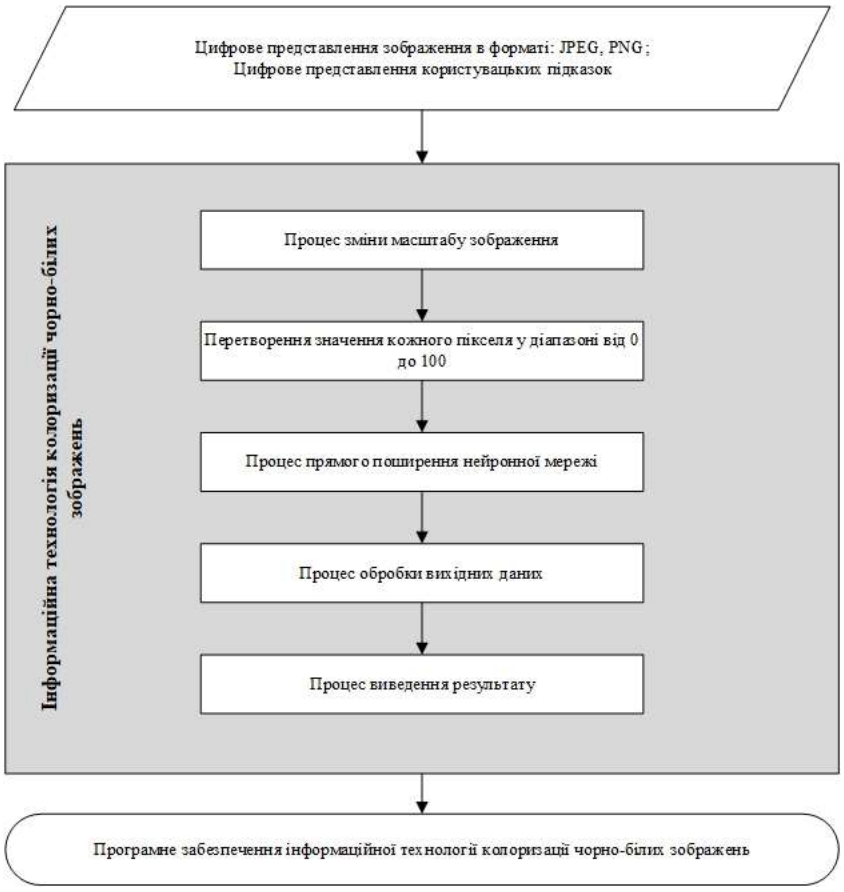

Рисунок 2.19 – Структура інформаційної технології розпізнавання та каталогізації рослин за зображенням листя з використанням нейронних мереж

2.7 Висновок

У другому розділі розроблено модель інформаційної технології колоризації чорно-білих зображень з використанням згорткової нейронної мережі. Цей метод є одним з найефективніших для вирішення задачі колоризації. Обрано метод навчання нейронної мережі, метод регуляризації, метод нормалізації, метод обчислення градієнту функції втрат.

Також розроблено архітектуру нейронної мережі для інформаційної технології колоризації чорно-білих зображень, що базується на архітектурі типу U-net, і обрано колірну модель CIELAB.

На основі цього розроблено загальну структурну схему та схему алгоритму функціонування інформаційної технології колоризації чорно-білих зображень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОЛОРИЗАЦІЇ ЧОРНО-БІЛИХ ЗОБРАЖЕНЬ

3.1 Обгрунтування вибору архітектурної організації обчислень на GPU (Caffe)

Зменшити час навчання нейронної мережі в декілька разів можна за рахунок використання GPU (Graphics Processing Unit). Сучасні GPU підходять не тільки для обробки графіки і комп'ютерних ігор, але і для обчислень. GPU містить сотні (а деякі моделі тисячі) обчислювальних ядер, тому розрахунки на ньому виконуються набагато швидше, ніж на CPU [37].

GPU (графічний процесор) – це окремий процесор розташований на відеокарті, який виконує обробку 2D або 3D графіки. Наявність процесора на відеокарті, звільняє комп'ютерний процесор від зайвої роботи і може виконувати всі інші важливі завдання швидше. Характерною ознакою графічного процесора (GPU), є те, що він максимально націлений на пришвидшення розрахунку саме графічної інформації (текстур і об'єктів). Завдяки своїй архітектурі такі процесори набагато ефективніше обробляють графічну інформацію, ніж звичайний центральний процесор комп'ютера.

У 2003 році GPU вперше отримала підтримку 32-бітних точних обчисленнях. Direct3D виділився основним інтерфейсом програмування, першим, хто надав підтримку шейдерів. З'явилися перші програми, які використовують GPU для високоефективних обчислень, напрямок GPGPU почав розвиватись. GPGPU (General-Purpose computing on Graphic Processing Units) – використання графічних процесорів для вирішення довільних обчислювальних задач. Для програмування GPU пропонується потоковий підхід до програмування. Цей підхід передбачає розбиття програми на відносно невеликі етапи (ядра), які обробляють елементи потоків даних. Ядра відображаються як шейдери, а потоки даних – як текстури в GPU [37].

Порівняння швидкості навчання нейронної мережі на CPU та GPU наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння швидкості навчання нейронної мережі на CPU та GPU

	GPU (NVIDIA GeForce 920M)	CPU (Intel Core i3-5005u)
Швидкість навчання однієї епохи, ~хв.	8	31
Швидкість навчання п’яти епох з використанням fine tuning, ~хв.	34	169

Багато сучасних систем навчання нейронних мереж підтримують GPU. Серед таких бібліотеки Theano, Tensorflow, Keras, Caffe. Приємна особливість в тому, що не доведеться переробляти програму на Keras, потрібно лише змінити налаштування обчислювального «бекенду».

Бібліотека – збірка підпрограм або об'єктів, що використовуються для розробки програмного забезпечення. У деяких мовах програмування (наприклад, в Python) те ж, що модуль, в деяких – кілька модулів [38].

У сучасних умовах розробити великий проект без використання готових бібліотек практично неможливо. Бібліотеки дозволяють програмісту зосередитися на поставлених ними завданнях, а не будувати інфраструктуру низького рівня для їх вирішення. Такий підхід може значно скоротити час на розробку та тестування програмного продукту, спростити його підтримку, а отже, зменшити його вартість.

Caffe використовує технологію CUDA, яка працює тільки на GPU виробництва компанії NVIDIA.

Compute Unified Device Architecture (CUDA) – це програмна модель, яка описує обчислювальну паралельність та структуру ієрархічної пам'яті безпосередньо для мови програмування. З точки зору програмного забезпечення,

реалізація CUDA – це кросплатформна система для збирання та виконання програм, частини яких працюють на процесорі та графічному процесорі. CUDA використовується для розробки GPGPU-додатків без прив’язки до графічних API і підтримується всіма GPU NVIDIA, починаючи з серії GeForce 8 [38].

Ця паралельна архітектура обчислень може значно підвищити продуктивність обчислень за рахунок використання графічних процесорів, які можуть дуже ефективно керувати великими блоками даних. Платформа паралельних обчислень CUDA використовується на сьогоднішній день в тисячах GPU-прискорених додатків і тисячах опублікованих наукових статтях [37].

CUDA використовується для наукових та дослідницьких цілей у таких галузях, як медична візуалізація, фінансове моделювання та енергетичні дослідження. Це також допомагає створити нове покоління програм для кінцевих користувачів у таких сферах, як перетворення мобільного відео та поліпшення якості відео на комп’ютерах.

CUDA має ряд переваг перед традиційним обчисленням загального призначення на GPU (GPGPU), використовуючи графічні API:

– Розсіяне читання – код можна читати з довільних адрес у пам'яті;

– Єдина віртуальна пам’ять (CUDA 4.0 і вище);

– Єдина пам’ять (CUDA 6.0 і вище);

– Спільна пам’ять – CUDA відкриває зону швидкої спільної пам’яті, яку можна розділити між потоками. Це призводить до більш високої пропускної здатності, ніж це можливо за допомогою текстурних пошуків.

– Швидше завантаження та відтворення на GPU

– Повна підтримка цілих і побітових операцій, включаючи цілі текстурні переходи.

На відміну від NVIDIA CUDA, що дозволяє GPU виконувати будь-які обчислення, бібліотека cuDNN спеціально розроблена для тренування глибоких нейронних мереж. Вона містить оптимізовані для GPU реалізації згорткових і рекурентних мереж, різних функцій активації (напівлінійна, сигмоїдальна,

гіперболічний тангенс, softmax), алгоритм зворотнього поширення помилки тощо. cuDNN дозволяє навчати нейронні мережі на GPU в кілька разів швидше, ніж просто CUDA.

cuDNN поширюється безкоштовно. Щоб її отримати, потрібно зареєструватися на сайті розробників NVIDIA.

Бібліотека cuDNN дозволяє використати наступну функціональність:

– Процедури forward-backward оптимізації.

– Операції над тензорами і їх використання в алгоритмах побудови нейронних мереж.

– Функції активації нейронної мережі (кусочно-лінійна, сигмоїдальна, функція гіперболічного тангенса).

– Просте інтегрування в будь-яку конфігурацію нейронної мережі.

3.2 Програмна реалізація

3.2.1 Обґрунтування вибору мови програмування

При виборі засобів для розробки програмного проекту необхідно врахувати надзвичайно велику кількість різноманітних аспектів, найбільш важливим із яких є мова програмування, тому що вона в значній мірі визначає інші доступні засоби. На приклад, для розробки користувацького графічного інтерфейсу розробникам необхідна сумісна з мовою програмування GUI-бібліотека, яка надає готові елементи інтерфейсу, такі, як кнопки и меню. При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність роботи додатку та ефективність використання пам’яті. Важливим чинником для вибору мови, в даному випадку є її універсальність.

Для розробки програмних систем такого рівня існує три мови, які відповідають вимозі універсальності: Java, C# та Python.

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU General Public License [37].

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові

програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [37].

Основним недоліком мови C# перед Java та Python є її пропріетарна ліцензія та можливість використання усіх її можливостей в повному обсязі лише на ОС Windows. Оскільки для розроблюваної система бажаною є повна підтримка UNIX-подібних операційних систем, зокрема GNU/Linux. Тому реальним є вибір між Java та Python.

Швидкість розробки програмного забезпечення мовою Python є на порядок вищою у порівнянні із Java завдяки його динамічній типізації. Крім того Python є інтерпретованою мовою, що полегшує внесення змін та налаштування програмної системи на етапі впровадження.

Проте основною перевагою при виборі мови Python було те, що для цієї мови програмування є найбільше нейромережевих бібліотек, такі як: Theano, Pylearn2, Deepnet, Caffe, Keras.

Тому основною мовою програмування обрано мову Python.

3.2.2 Обгрунтування вибору нейромережевої бібліотеки

Як зазначалося раніше, Python є найбагатшим у програмуванні бібліотек нейронної мережі. Бібліотека Caffe була обрана для впровадження інформаційної технології колоризації чорно-білих зображень.

Розробка Caffe ведеться з вересня 2013 р. Початок розробки поклав Yangqing Jia під час його навчання в каліфорнійському університеті в Берклі. З зазначеного моменту Caffe активно підтримується Центром Зору і Навчання Берклі (The Berkeley Vision and Learning Center, BVLC) і співтовариством розробників на GitHub. Бібліотека поширюється під ліцензією BSD 2-Clause.

Caffe реалізована з використанням мови програмування C ++, є обгортки на Python і MATLAB. Офіційно підтримувані операційні системи – Linux і OS X, також є неофіційний порт на Windows. Caffe використовує бібліотеку BLAS

(ATLAS, Intel MKL, OpenBLAS) для векторних і матричних обчислень. Поряд з цим, в число зовнішніх залежностей входять glog, gflags, OpenCV, protoBuf, boost, leveldb, nappy, hdf5, lmdb. Для прискорення обчислень Caffe може бути запущена на GPU з використанням базових можливостей технології CUDA або бібліотеки примітивів глибокого навчання cuDNN [38].

Розробники Caffe підтримують можливості створення, навчання і тестування повнозв'язаних і згорткових нейромереж.

Підводячи підсумки, хотілося б ще раз підкреслити, що в порівнянні з аналогами Caffe має низку переваг, а саме:

1. «чиста» архітектура дозволяє здійснити миттєве розгортання. Мережі визначаються простими файлами конфігурації, без жорсткого «вшивання» параметрів даних в код. Це робить перемикання між процесором та графічним процесором легким та швидким – дає можливість тренувати моделі на ПК із потужним графічним процесором, а потім використовувати його на будь-якій кластерній машині.

2. відкритий код дозволяє не тільки контролювати впровадження, але і модифікувати його під свої потреби. За перші 6 місяців активного використання Caffe понад 300 незалежних розробників внесли свій внесок у розвиток бібліотеки, просто налаштувавши її на себе.

3. швидкість роботи робить Caffe незамінним інструментом для комерційного використання. Всього лише на одній машині з графічним процесором рівня NVIDIA K40 за допомогою бібліотеки можна обробити більше 400 млн зображень на добу. Це відповідає швидкості навчання в 5 мілісекунд на зображення і швидкості тестування в 2 мілісекунди на зображення. Розробники справедливо підкреслюють, що Caffe – лідер за швидкістю роботи серед згорткових нейронних мереж [39].

3.2.3 Обґрунтування вибору середовища розробки

Інтегроване середовище розробки, ICP (англ. IDE, Integrated development environment) – система програмних засобів, використовувана програмістами для розробки програмного забезпечення [40].

Зазвичай, середовище розробки включає в себе:

- текстовий редактор,
- компілятор або інтерпретатор,
- засоби автоматизації збирання,
- відладчик.

Іноді містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованої розробки ПЗ. Хоча й існують ICP, призначені для декількох мов програмування – такі як Eclipse, Embarcadero RAD Studio, Qt Creator, останні версії NetBeans, Xcode або Microsoft Visual Studio, але зазвичай ICP призначається для одної певної мови програмування – як, наприклад, Visual Basic, Delphi, Dev-C++.

Інтегровані середовища розробки були створені для того, щоб максимізувати продуктивність програміста завдяки тісно пов'язаним компонентам з простими користувацькими інтерфейсами. Це дозволяє розробнику зробити менше дій для перемикання різних режимів, на відміну від дискретних програм розробки.

ICP, зазвичай, являє собою єдину програму, в якій проводилася вся розробка. Вона, зазвичай, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програмного забезпечення. Мета середовища розробки полягає в тому, щоб абстрагувати конфігурацію, необхідну, щоб об'єднати утиліти командного рядка в одному модулі, який

дозволить зменшити час, щоб вивчити мову, і підвищити продуктивність розробника. Також вважається, що важка інтеграція завдань розробки може далі підвищити продуктивність. Наприклад, ICP дозволяє проаналізувати код і тим самим забезпечити миттєвий зворотний зв'язок і повідомити про синтаксичні помилки.

Сьогодні існує досить велика кількість ICP для Python: Boa Constructor, Eclipse, PyDev , Eric, IDLE, Komodo, PyCharm, PyScripter, SPE, Wing IDE. Проте їхні можливості поки що не можуть задовольнити усіх потреб розробників програмного забезпечення. Винятками є Eclipse та PyCharm.

Основним середовищем для розробки сервісу колоризації чорно-білих зображень було обрано PyCharm, інтерфейс головного вікна якого зображено на рисунку 3.1.

PyCharm – інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django та Flask. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA [41].

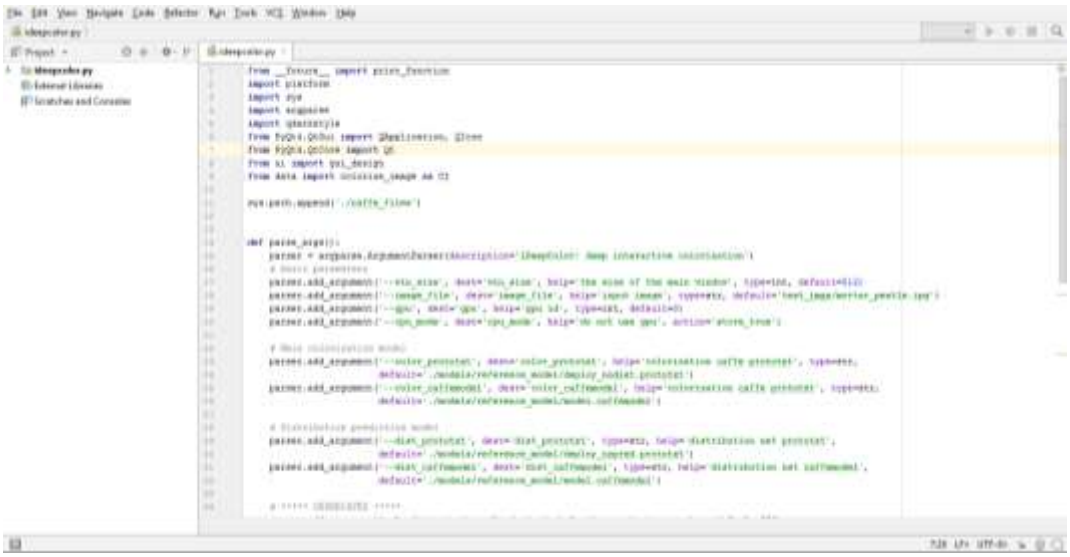

Рисунок 3.1 – Головне меню ICP PyCharm

PyCharm працює під операційними системами Windows, Mac OS X і Linux.

Властивості ICP PyCharm:

- Статичний аналіз коду, підсвічування синтаксису і помилок.
- Навігація за проектом і вихідного коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів.
- Рефакторинг: перейменування, витяг методу, введення змінної, введення константи, підйом і спуск методу і т. д.
- Інструменти для веб-розробки з використанням фреймворку Django та Flask.
- Вбудований відладчик для Python.
- Вбудовані інструменти для юніт-тестування.
- Підтримка систем контролю версій: загальний користувальницький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття.
- Велика кількість плагінів, що дозволяють значно розширити можливості ICP, зокрема плагіни інтеграції із MongoDB

3.2.4 Розробка схеми алгоритму тренування і валідації та тестування нейронної мережі

Основні етапи при застосуванні технік машинного навчання – це тренування, валідація та тестування. Етап тренування відбувається циклічно, кожен цикл триває одну епоху (повний перебір зразків тренувального набору даних), етап валідації відбувається після кожної епохи тренуваня і служить для перевірки покращення роботи моделі на невідомих даних (з набору даних для валідації).

Послідовність наступних кроків описує алгоритм тренування та валідації:

1. Завантаження архітектури та підмодулів нейронної мережі.
2. Ініціалізація ваг (випадковим чином)

3. Задання функції втрат.
4. Ініціалізація алгоритму оптимізації.
5. Ініціалізація набору даних для тренування і валідації.
6. Ініціалізація генератора даних для набору даних тестування і валідації.
7. Епоха тренування:
    7.1. Для кожного міні-пакета (mini-batch) із генератора даних для тренування:
        7.1.1. Очищення градієнта.
        7.1.2. Крок проходу вперед.
        7.1.3. Обчислення втрат.
        7.1.4. Крок проходу назад.
        7.1.5. Крок оптимізації.
8. Епоха валідації:
    8.1. Для кожного mini-batch із генератора даних для валідації:
        8.1.1. Крок проходу вперед.
        8.1.2. Обчислення втрат.
9. Збереження моделі.
10. Тонке налаштування моделі (fine tuning).
11. Збереження результуючої моделі.

Відповідну схему алгоритму тренування і валідації нейронної мережі наведено на рисунку 3.2.

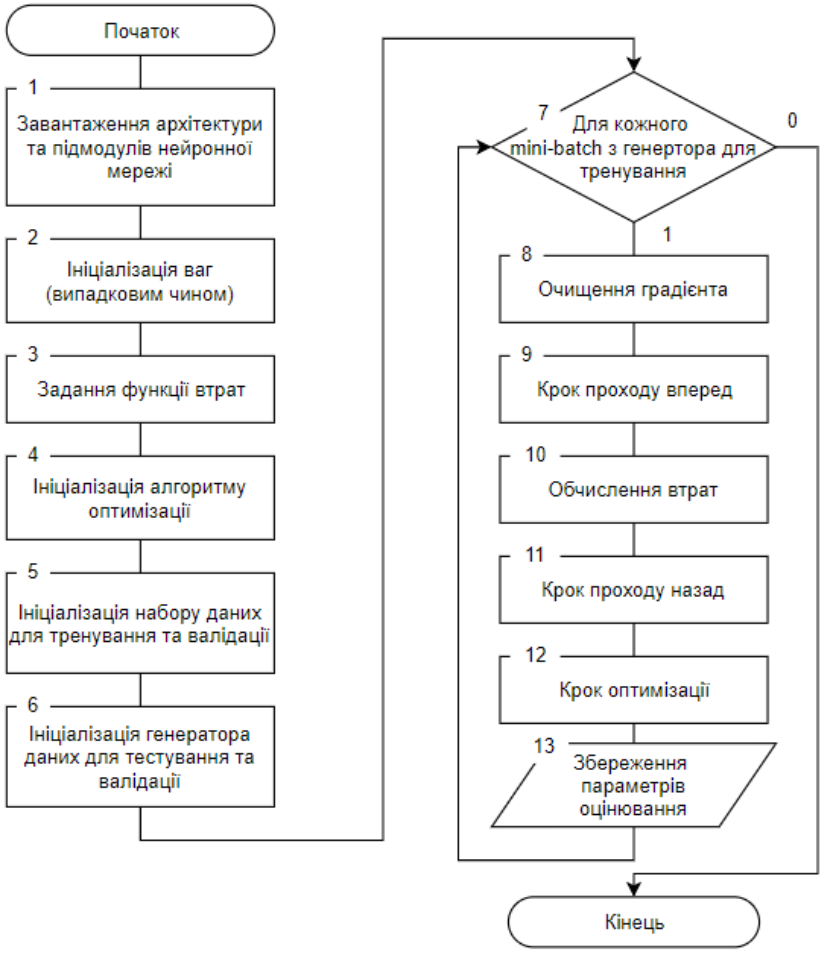

Рисунок 3.2 – Загальна схема алгоритму тренування і валідації нейронної мережі

Алгоритм тестування можна описати, як послідовність наступних кроків:

1. Завантаження параметрів моделі з файла.
2. Завантаження архітектури та підмодулів нейронної мережі.
3. Застосування параметрів моделі
4. Ініціалізація функції втрат.
5. Ініціалізація набору даних для тестування.
6. Ініціалізація завантажувача даних для набору даних для тестування.
7. Епоха тестування:

7.1. Для кожного міні-пакета із завантажувача даних для тестування:

7.1.1. Крок проходу вперед.

7.1.2. Обчислення втрат.

7.1.3. Збереження моделі.

Відповідну схему алгоритму тестування нейронної мережі наведено на рисунку 3.3

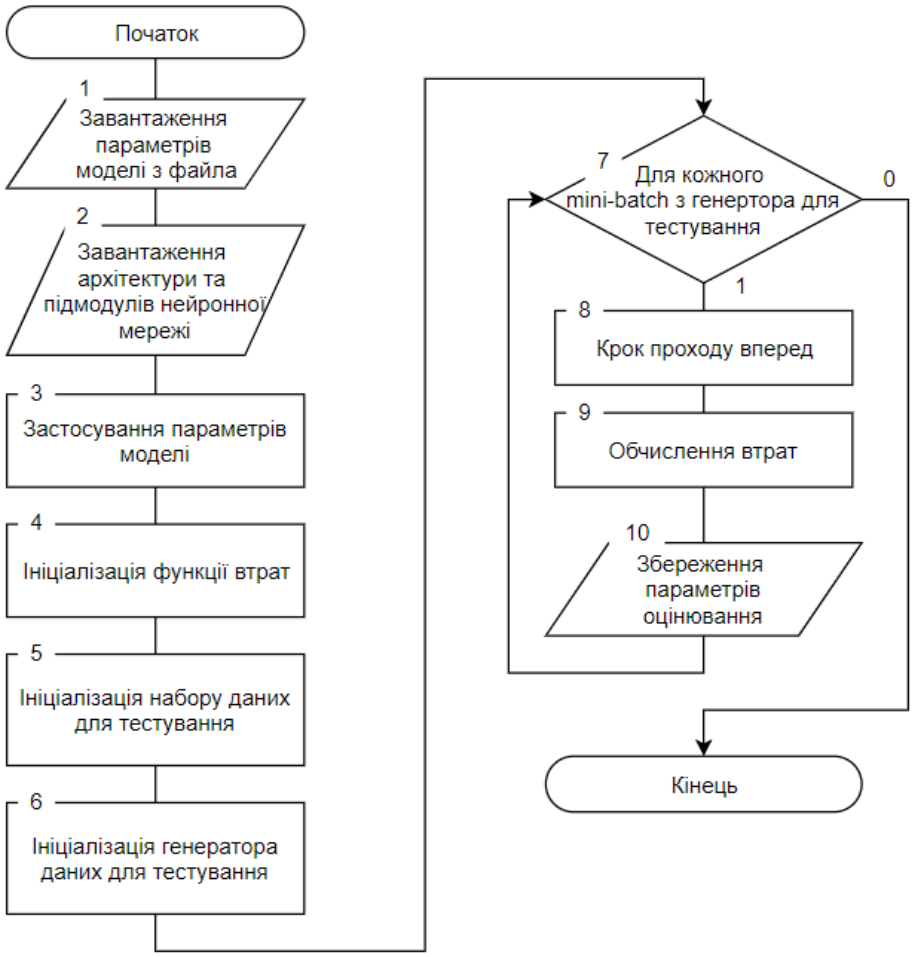

Рисунок 3.3 – Загальна схема алгоритму тестування нейронної мережі

3.2.5 Побудова UML-діаграми класів розробленого програмного засобу

На рисунку 3.4 зображено UML-діаграму класів розробленої системи.

Розроблено класи Main, GUIDesign, ColorizeImage, LabGamut.

Клас Main відповідає за початкову активність програми. Викликає клас Main GUIDesign.

Клас GUIDesign відповідає за відображення головного меню програми і взаємодію з користувачем.

Клас ColorizeImage відповідає за основну логіку колоризації чорно-білих ображень

Клас LabGamut відповідає за логіку програми, в якій відбувається підбір запропонованих для користувача кольорів точок-підказок.

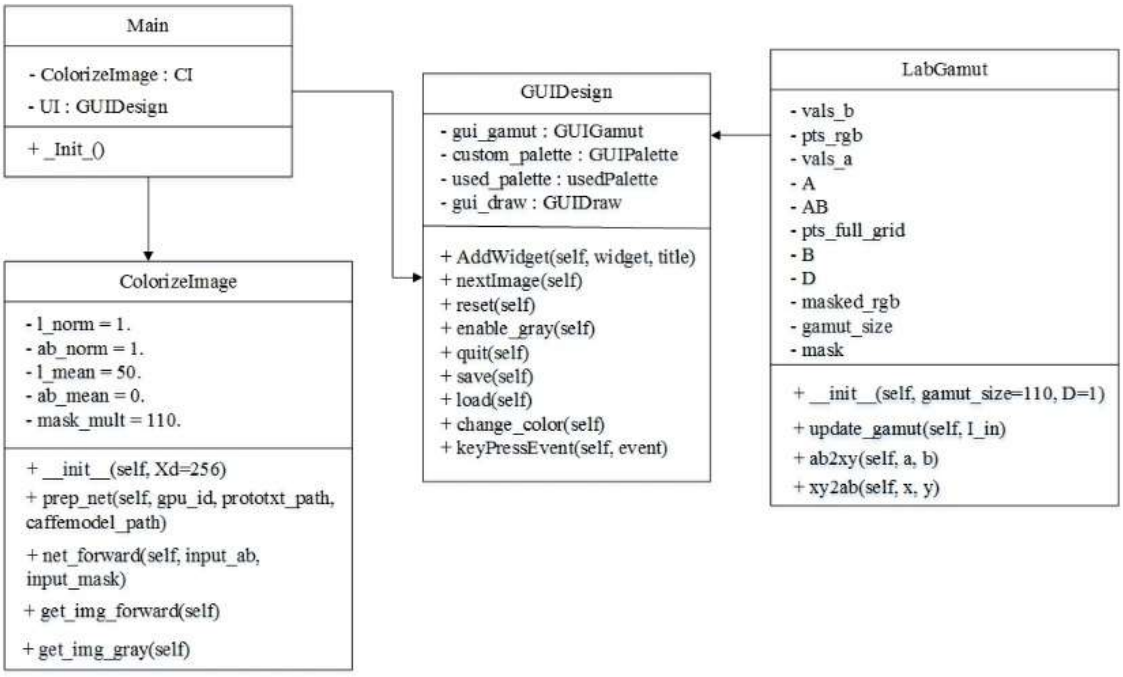

Рисунок 3.4 – UML-діаграма класів програмного засобу інформаційної технології колоризації чорно-білих зображень

3.2.6 Тестування розробленого програмного засобу колоризації чорно-білих зображень та аналіз результатів його роботи

Розроблена система колоризації чорно-білих зображень була протестована, що підтвердило коректність її роботи.

Було проведено 100 запусків додатку, протестовано можливості його роботи, що дало можливість адекватно оцінити його роботу.

Після запуску програми відкривається головне вікно, в якому відбувається всі дії (рисунки 3.5-3.6).

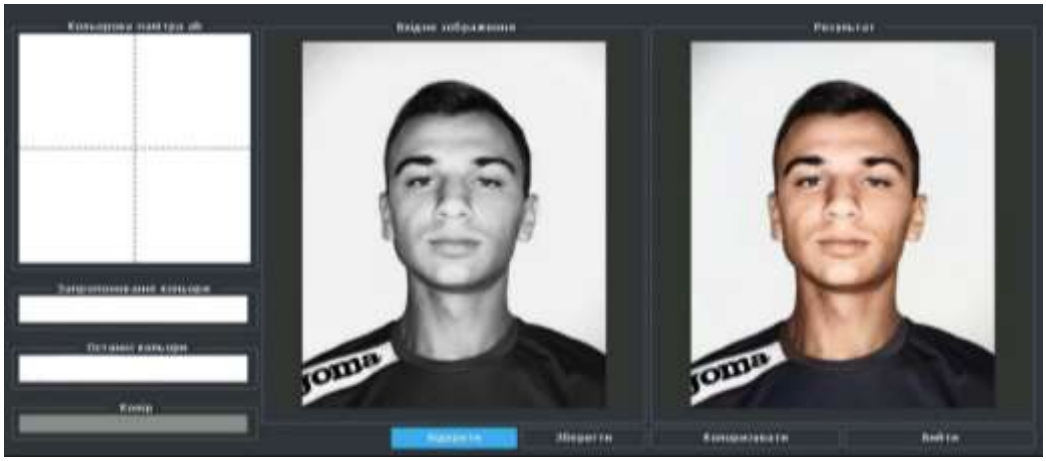

Рисунок 3.5 – Головне вікно розробленого програмного засобу інформаційної технології колоризації чорно-білих зображень без використання користувацьких підказок

Рисунок 3.6 – Головне вікно розробленого програмного засобу інформаційної технології колоризації чорно-білих зображень з використанням користувацьких підказок

Головне вікно розробленої системи колоризації чорно-білих зображень містить такі елементи взаємодії з користувачем:

- Кнопка «Відкрити» – для відкриття чорно-білого зображення, яке потрібно колризувати;

- Панель «Вхідне зображення» містить відкрите чорно-біле зображення. Також ця панель використовується користувачем для розставлення кольорових точок-підказок;
- Панель «Корольова палітра ab» – на ній відображається кольорова гама a,b-каналів кольорового простору CIELAB для кожної точки поставленої користувачем. Ця гама формується на базі значення L-каналу в даній точці зображення. На ній можна обрати колір для точки-підказки;
- Панель «Запропоновані кольори» – на ній відображаються набір найбільш підходящих кольорів для кожної точки, запропоновані нейронною мережею. На ній можна обрати колір для точки-підказки;
- Панель «Останні кольори» – на ній відображаються останні використані кольори. На ній можна обрати колір для точки-підказки;
- Панель «Колір» – на ній відображається поточний колір;
- Кнопка «Колоризувати» – для початку колоризації;
- Панель «Результат» містить колоризоване зображення. Зображення оновлюється після кожного запуску колоризації;
- Кнопка «Зберегти» – для збереження колоризованого зображення на вашому комп’ютері;
- Кнопка «Вийти» – для закриття програми.

Як бачимо, що результат колоризації без користувацької підказки відрізняється від результату з однією підказкою. В таблиці 3.2 наведено результати порівняння роботи розробленого програмного модуля з іншим програмним забезпеченням для колоризації чорно-білих зображень. На рисунку 3.7 зображено порівняння результатів роботи розробленої програми з аналогом.

Рисунок 3.7 – Приклад колоризації розробленим додатком і аналогом

Отже, після запуску тестових запусків програми можна зробити наступні висновки: програма колоризує зображення як з підказкою про колір, так і без підказки. Було встановлено, що отримане нами зображення після використання підказок краще забарвлюється, на відміну від способу без підказок. Результати, отримані під час тестування програми, відповідають очікуваним результатам. Розроблений програмний додаток працює відповідно до завдань проектування. Розроблений додаток має кращу якість колоризації порівняно з аналоговими програмами. Якість зросла майже на 2,8% порівняно з прототипом, що є дуже хорошим результатом.

Оцінювання якості відбувалось в такий спосіб, ми взяли кольорове зображення, створили чорно-білу версію цього зображення. Потім колоризували

дане чорно-біле зображення розробленою програмою і програмою аналогом. І порахували індекс структурної схожості DSSIM, який застосовується для вимірювання схожості між двома зображеннями [42]. Він коливається він -1 до 1. Ми порахували цей індекс для таких пар зображень: оригінал – зображення колоризоване розробленим програмним модулем; оригінал – зображення колоризоване Akvis Coloriage; оригінал – зображення колоризоване Colorize Photos Algorithmia.

Таблиця 3.2 – Порівняльний аналіз достовірності роботи різних алгоритмів вирішення задачі

Програма	DSSIM
Розроблений додаток без підказок	0,094
Розроблений додаток з 5 підказками	0,102
Розроблений додаток з 10 підказками	0,106
AKVIS Coloriage	0,087
Colorize Photos Algorithmia	0,072

3.3 Висновок

У даному розділі розглянуто особливості архітектурної організації обчислень на GPU і обгрунтовано доцільність її використання.

Здійснено вибір мови програмування, інтегрованого середовища розробки, нейромережевої бібліотеки. В роботі використано мову програмування Python та бібліотеку Caffe, які є найпотужнішими інструментами для роботи зі згортковими нейронними мережами, а PyCharm є найзручнішими середовищем для розробки для даної мови програмування.

Розроблено алгоритми тренування і валідації та тестування нейронної мережі. Розроблено UML-діаграму класів додатку колоризації чорно-білих зображень. На основі цього було програмно реалізовано додаток.

Проведено тестування розробленого програмного додатку, яке підтвердило коректність його роботи. Також було проведено порівняльний аналіз створеної системи з існуючими аналогами по критерію «якість» колоризації. В підсумку в порівнянні з аналогом якість колоризації зросла на 2.8%.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Арсенюк І.Р. та Сілагін О.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	Арсенюк І.Р.	Сілагін О.В.
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	3
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 43$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 22 дні;

t – число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	6000	272,72	5	1363,5
Інженер-програміст	3400	154,54	50	7727
Всього:				9090,5

Розрахуємо додаткову заробітну плату:

$$З_{дод}= 0,1\cdot 9090,5=909,05 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп}= (9090,5+909,05)\cdot \frac{36,3}{100} = 3629,83 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і}, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	130	1	130
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				352

Витрати на силову електроенергію розраховуються за формулою:

$$\text{В}_\text{е}=\text{В}\cdot\text{П}\cdot\text{Ф}\cdot\text{К}_\text{п}\ , \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп’ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=200 год.);

$\text{К}_\text{п}$ – коефіцієнт використання потужності ($\text{К}_\text{п} < 1$, $\text{К}_\text{п} = 0{,}7$).

$$В_е= 1,7 \cdot 0,6 \cdot 200 \cdot 0,7 = 142,8 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (9090,5 + 909,05) = 9999,55 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

$$В = 9090,5+909,05+3629,83+625+352+142,8+9999,55= 24748,73 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha}, \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{24748,73}{1} = 24748,73.$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{24748,73}{0,9} = 27498,58 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 150 користувачів, протягом другого року – на 125 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 500 + (400 + 25) \cdot 150 = 76250 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125) = 129375 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125 + 100) = 171875 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

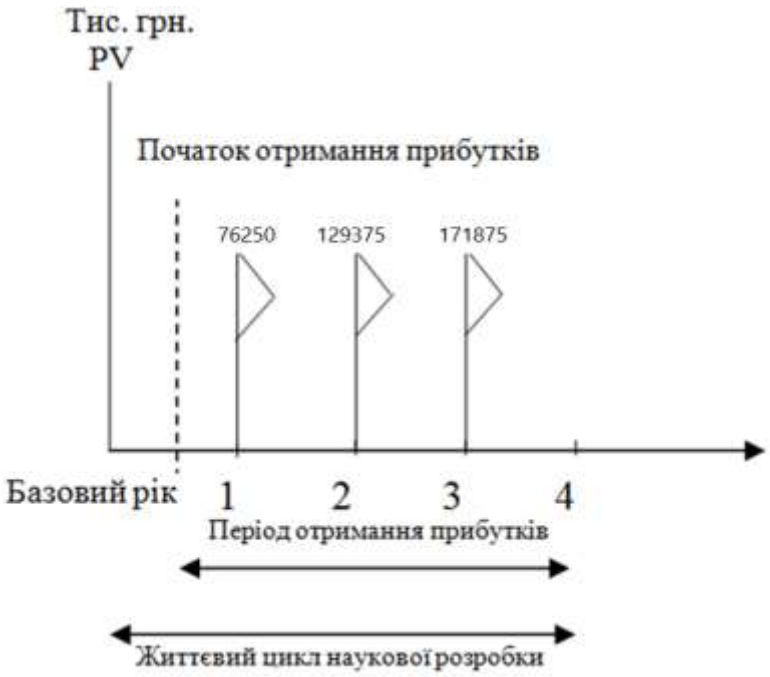

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{27498{,}58}{(1+0{,}1)^0} + \frac{76250}{(1+0{,}1)^2} + \frac{129375}{(1+0{,}1)^3} + \frac{171875}{(1+0{,}1)^4} = 305109{,}38 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 305109,38 - 27498,58 = 277610,8 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{277610,8}{27498,58}} - 1 = 1,23 \text{ або } 123\ \%.$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 123% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в},$$

$$T_{ок} = \frac{1}{1{,}23} = 0{,}81 \text{ року}.$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію колоризації чорно-білих зображень на основі згорткової нейронної мережі. Під час аналізу предметної області було відзначено, що автоматизована колоризація чорно-білих зображень можуть знайти своє застосування в багатьох сферах та значно полегшує життя людям. Проте сьогодні не існує досконалого засобу, який є зручним в використані, та дозволяє з достатньою якістю колоризувати чорно-білі зображення. Визначено основні проблеми при колоризації. Аналіз предметної області колоризації чорно-білих зображень показав, що виконання цього завдання з використанням штучних нейронних мереж та машинного навчання є найбільш якісним методом.

Розроблено архітектуру згорткової нейронної мережі, з власним удосконаленням, яке полягає в додаванні ще одніє мережі кольорових підказок і це дозволило покращити якість колоризації, досліджено основні методи навчання згорткових нейронних мереж та обрано найбільш доцільний. В ході роботи запропоновано структуру інформаційної технології.

Було розроблено алгоритм роботи програмного засобу, на основі якого було реалізовано програмний засіб колоризації чорно-білих зображень на основі згорткової нейронної мережі за допомогою мови програмування Python та середовища розробки PyCharm, та з використанням нейромережевої бібліотеки Caffe.

Проведено тестування програмного засобу, яке показало, що удосконалена архітектура згорткової нейронної мережі дала можливість покращити якість колоризації чорно-білих зображень на в середньому на 2,8 %. В результаті виконання даної кваліфікаційної роботи поставлені задачі були виконані в повній мірі. Отже, мета магістерської роботи досягнута

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Павлович Р.І Порівняльний аналіз методів навчання нейронних мереж. / Павлович Р.І. [Електронний ресурс] – режим доступу : https://card-file.onaft.edu.ua/handle/123456789/10175

2. Павлович Р.І Колризація чорно-білих зображень з використанням згорткових нейронних мереж. НТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії. / Павлович Р.І. [Електронний ресурс] – режим доступу : https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2018/paper/view/4117

3. Матеріали XI міжнародної науково-практичній конференції «ІОН-2018» [Електроний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/

4. Свідоцтво про реєстрацію авторського права на твір № 89015. «Нейромережевий модуль колоризації чорно-білих зображень» / Арсенюк І.Р., Павлович Р.І. Дата реєстрації Державною службою інтелектуальної власності України 29.11.2019.

5. Компьютерная Графика и Мультимедиа Сетевой журнал Выпуск №4(4)/2006 Методы сегментации изображений: автоматическая сегментация [Электронный ресурс] – Режим доступа: http://cgm.computergraphics.ru/content/view/147.

6. Automatic Colorization [Electronic resource] / Tinyclouds – Mode of access: http://tinyclouds.org/colorize/

7. Adobe Photoshop [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Adobe_Photoshop

8. Colorize Photo [Електроний ресурс] – Режим доступу: https://www.colorizephoto.com/

9. Algorithmia Colorize Photos [Electronic resource]] – Mode of access: https://demos.algorithmia.com/colorize-photos/

10. AKVIS Coloriage [Електроний ресурс] – Режим доступу: http://akvis.com/ru/coloriage/index.php

11. Колоризация полутонового изображения [Текст] / А.П. Платонов, И.В. Лезина // Королевские чтения: всероссийская молодежная научная конференция / Самарск. гос. аэрокосмич. ун-т им. академика С.П. Королева. Самара: СГАУ, 2011.

12. Мак-Каллок У. С., Питтс В. Логическое исчисление идей, относящихся к нервной активности // Автоматы / Под ред. К. Э. Шеннона и Дж. Маккарти. — М.: Изд-во иностр. лит., 1956. — С. 363—384. (Перевод английской статьи 1943 г.)

13. Головко, В.А под ред. проф. Галушкина, А.И. Нейронные сети: обучение, организация и применение [Текст]/ Головко, В.А. – ИПРЖР, Москва, 2001 г.

14. Борисов, Е.С. Нейронные сети: обучение с учителем [Электронный ресурс]/ Борисов, Е.С.– http://scorcher.ru/neuro/science/perceptron/ mem32.htm

15. Doyle, W. Operations useful for similarity invariant pattern recognition. // Journal ACM. [Текст]/ Doyle, W.. – 1962 Том. 9, № 2. С. 259-267.

16. Хайкин, С. «Нейронные сети» [Текст]/ Хайкин, С. – Москва, 2006 г. – 1105 с.

17. Леоненков, А. Самоучитель UML [Текст] / А. Леоненков. - СПб. : БХВ- Санкт-Петербург, 1999. - 304 с

18. Дерябкин, В.П. Проектирование автоматизированных систем обработки информации и управления. Курс лекций. СГАУ, 2011 г. [Текст]/ Дерябкин, В.П.. – 120с.

19. Прохоров, С.А. Аппроксимативный анализ случайных процессов. [Текст]/А.С.Прохоров. –2-е изд., перераб. и доп./СНЦ РАН, 2001. – 125с.

20. Paul J. Werbos , The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting. / New York, NY: John Wiley & Sons, Inc. – 1994.

21. Alona, What is Backpropagation / Alona // Personal page [Electronic resource] – Mode of access: https://alonalj.github.io/2016/12/10/What-is-Backpropagation/

22. M. Andrychowicz, Learning to learn by gradient descent by gradient descent / Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman [and others] // arXiv [Electronic resource]. – 30.11.2016. – Mode of access: https://arxiv.org/pdf/1606.04474.pdf

23. L. Bottou, Online Algorithms and Stochastic Approximations / Bottou Leon, Online Learning and Neural Networks. – Cambridge University – 1998

24. Diederik P. Kingma, ADAM: a method for stochastic optimization / Diederik P. Kingma, Jimmy Lei Ba in ICLR 2015, 2015

25. I. Goodfellow, Generative Adversarial Nets / Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu [and others] // arXiv [Electronic resource]. – 10.06.2014 – Mode of access: https://arxiv.org/pdf/1406.2661.pdf

26. M. Mirza, Conditional Generative Adversarial Nets / Mehdi Mirza // arXiv [Electronic resource]. – Mode of access: https://arxiv.org/pdf/1411.1784.pdf

27. Как устроены нейронные сети? [Электронный ресурс] – Режим доступа: https://rb.ru/opinion/neuron-networks/

28. 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

29. 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

30. Convolutional Neural Networks (LeNet) - DeepLearning 0.1 documentation. / deeplearning.net [Electronic resource]. – 2011.– Mode of access: http://deeplearning.net/tutorial/lenet.html

31. 10. D. Scherer, Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition / Scherer, Dominik; Müller, Andreas C.; Behnke, Sven in Artificial Neural Networks (ICANN), 20th International Conference on. Thessaloniki – 2010, Greece: Springer. pp. 92–101.

32. Колірна модель [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Колірна_модель

33. Цвет в компьютерной графике [Электроный ресурс] – Режим доступа: http://eor.dgu.ru/lectures_f/Курс_леций_Компьютерная_геометрия_и_графика_Гаджиев_А_М/лекция_4.html

34. HSV [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/HSV

35. CIELAB [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/CIELAB

36. U-Net: Convolutional Networks [Electronic resource]. – Mode of access: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

37. Сравнение языков программирования [Электроный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Сравнение_языков_программирования

38. Caffe – Deep Learning Framework [Electronic resource] – Mode of access: http://caffe.berkeleyvision.org/

39. Инструментарий специалиста по большим данным: Caffe [Электроный ресурс] – Режим доступа: http://datareview.info/article/instrumentariy-spetsialista-po-bolshim-dannyim-caffe/

40. Інтегроване середовище розробки [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки

41. PyCharm [Электроный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PyCharm

42. SSID [Электроный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/SSID