Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ КОЛОРИЗАЦІЇ НАПІВТОНОВИХ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2
Кирилова В.А.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кириловій Вікторії Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів колоризації напівтонових зображень

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 30 грудня 2023 р.

3. Вихідні дані до роботи науково-методична література, чорно-білі (напівтонові) зображення, контекстні дані, дані для аналізу якості, використані програмні засоби: мова програмування Python, середовище розроблення Google Colab.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд існуючих методів колоризації і їх переваги та недоліки.
2. Реалізація програмного забезпечення для колоризації зображень.
3. Експериментальна оцінка та порівняння реалізованих методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми колоризації зображень, постановка задачі, аналіз методів, тестові зображення, реалізація методів для проведення аналізу та порівняння.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	03.11.23-05.11.23	
3	Аналіз літератури з досліджуваної проблеми	05.11.23-08.11.23	
4	Дослідження методів	08.11.23-15.11.23	
5	Аналіз технічних засобів	15.11.23-17.11.23	
6	Програмна реалізація	17.11.23-25.11.23	
7	Оформлення пояснювальної записки	25.11.23-26.11.23	
8	Перевірка на плагіат	04.12.2023	
9	Рецензування	15.12.2023	
10	Підготовка презентації та доповіді	25.12.2023	
11	Занесення роботи в електронний архів	05.01.2024	
12	Попередній захист кваліфікаційної роботи	09.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 79 с., 3 табл., 59 рис., 1 дод., 46 джерел.

МЕТОДИ КОЛОРИЗАЦІЇ, НАПІВТОНОВЕ ЗОБРАЖЕННЯ, АЛГОРИТМИ КОЛОРИЗАЦІЇ, АВТОМАТИЗОВАНА КОЛОРИЗАЦІЯ.

Об’єктом дослідження є методи колоризації напівтонових зображень, що включають в себе різноманітні підходи та алгоритми для надання кольору саме чорно-білим фотографіям.

Метою дослідження є вивчення та оцінка методів колоризації напівтонових зображень з використанням різних алгоритмів та технологій.

Проведено аналіз методів колоризації напівтонових зображень на прикладах окремих зображень (фотографій), зроблених в реальному житті. Реалізовано методи колоризації для подальшої експериментальної оцінки та порівняння.

У результаті дослідження здійснена програмна реалізація методів колоризації, проведено аналіз та порівняння. Проведене дослідження розкриває переваги та недоліки існуючих методів.

COLORIZATION METHODS, HALFTONE IMAGE, COLORIZATION ALGORITHMS, AUTOMATED COLORIZATION.

The object of research is the methods of coloring halftone images, which include various approaches and algorithms for giving color to black and white photographs.

The purpose of the study is to study and evaluate the methods of coloring halftone images using various algorithms and technologies.

An analysis of the methods of coloring halftone images was carried out using the examples of individual images (photographs) taken in real life. Colorization methods have been implemented for further experimental evaluation and comparison.

As a result of the research, software implementation of colorization methods was carried out, analysis and comparison were carried out. The conducted research reveals the advantages and disadvantages of existing methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Networks (глибокі нейронні мережі)

GAN – Generative Adversarial Network (генеративна змагальна мережа)

VAE-GAN – Variational Autoencoder Generative Adversarial Network (Генеративна зворотна мережа з варіаційним автокодером)

TPN – Text-to-Palette Generation Networks (мережі генерації тексту в кольори)

PCN – Palette-based Colorization Networks (мережі колоризації на основі палітр)

OpenCV – Open Source Computer Vision Library (бібліотека комп’ютерного зору з відкритим кодом)

ВСТУП

Колоризація зображень є однією з важливих галузей обробки зображень, оскільки вона дозволяє надати чорно-білим або напівтоновим зображенням реалістичного кольору. Завдання колоризації, яке призначене для автоматизації процесу додавання кольору до зображень, має безпосередній вплив на різні сфери життя та діяльності. У наш час колоризація є необхідною складовою візуальних ефектів у кіноіндустрії та відеоіграх, в архівній реставрації цінних історичних документів та фотографій, в медичній діагностиці та аналізі зображень, а також в журналістиці, де кольорові фотографії та відео надають подіям більшу об’єктивність і силу передачі інформації.

Завданням колоризації є призначення кольорів для кожного пікселя на зображенні. Це важливо не лише для творчого виразу, але й для забезпечення правильної інтерпретації наданого зображення. Однак завдання колоризації залишається складним через велику кількість факторів, що впливають на правильний вибір кольору для кожного пікселя. До цих факторів відносяться контекст зображення, наявність об’єктів на зображенні, освітлення, тіні, архітектурні особливості тощо.

Мета даної роботи полягає в дослідженні та порівнянні різних методів колоризації напівтонових зображень, таких як традиційні методи та нові розробки в галузі штучного інтелекту. Буде проведено аналіз наукових підходів та практичних методів, які використовуються для вирішення цього завдання. Метою дослідження буде визначення їхньої ефективності у відтворенні кольору та деталей на різних типах зображень, а також розгляд можливих шляхів покращення результатів та зменшення обмежень.

Процес колоризації має численні виклики та аспекти, які потребують уважного вивчення і дослідження. Буде акцентуватися увага на таких ключових факторах, як підтримка деталей у зображенні, відтворення природного кольору, обробка освітлення, корекція тіней та взаємодія з

об’єктами на зображенні. У роботі буде розкрито результати досліджень, аналізу та порівняння різних методів, визначено їхні переваги та недоліки.

Важливо зазначити, що завдання колоризації стає актуальнішим завдяки розвитку технологій та поширенню масового доступу до обчислювальних ресурсів. В останні роки з’явилися численні програмні рішення та онлайн-сервіси, які пропонують автоматизовані методи колоризації, що робить цю функцію доступною широкому загалу користувачів.

Проте, необхідно пам’ятати, що незважаючи на досягнення, задача колоризації залишається відкритою для подальших досліджень і покращень. Сучасні методи, хоч і дають істотні результати, все ще мають обмеження та виклики, які вимагають подальшого дослідження. Ці обмеження можуть включати проблеми з точністю відтворення кольорів у складних сценах, втрату деталей під час процесу колоризації, а також необхідність великого обсягу даних для навчання нейронних мереж.

Ця робота допоможе розкрити потенціал та обмеження різних методів колоризації, спростити їхню реалізацію та розуміння. Результати дослідження можуть служити корисною інформацією для фахівців у галузі обробки зображень, мистецтва та комп’ютерного бачення, сприяючи подальшому розвитку цієї галузі технологій.

1 ОГЛЯД ОСНОВНИХ ПРИНЦИПІВ КОЛОРИЗАЦІЇ НАПІВТОНОВИХ ЗОБРАЖЕНЬ

Напівтоновим називають зображення, яке використовує різні відтінки або градації одного кольору для передачі інформації. Це поняття широко використовується в контексті фотографії, друку, мистецтва та графіки для створення зображень, які передають деталі та глибину, використовуючи обмежений спектр кольорів.

У кольоровому напівтоновому зображенні використовуються відтінки одного кольору для створення градацій та додавання глибини до зображення.

Найбільш розповсюдженим типом серед напівтонових зображень є чорно-біле напівтонове зображення. У цьому випадку використовуються відтінки сірого, щоб передати різні рівні яскравості від чорного до білого. Це дозволяє створювати зображення, які відтворюють градації світлового та тіньового ефектів, забезпечуючи високий рівень деталізації.

Головна перевага напівтонових зображень полягає в їх здатності точно передавати тіні, світлові відтінки та деталі на зображенні. Вони широко використовуються в друкарстві для відтворення фотографій, в мистецтві для створення тонких переходів між відтінками, а також в графічному дизайні для створення привабливих візуально та ефективних зображень.

Колоризація таких зображень слугує корисним інструментом для отримання більшої інформації. Наприклад, якщо мова йдеться про історичні фотографії або портрети. Також це надає додаткові можливості для експериментів з палітрою художникам для створення картин та відкриття нових поєднань кольору. Колоризація може бути використана для створення реалістичних зображень природи, архітектури та інших об’єктів, де кольоровий контекст важливий для повного сприйняття об’єкта.

Ці всі цілі є цілком досяжними завдяки вже існуючим різним методам колоризації.

Незалежно від способу реалізації, в різних методах колоризації є декілька спільних рис. Кожен метод колоризації починає з чорно-білого зображення. Серед загальних кроків можна вказати попередню обробку зображення, наприклад підвищення контрасту (рис. 1.1) або позбавлення шуму (рис. 1.2) для покращення якості результату.

а) б)

Рисунок 1.1 – Обробка зображення для підвищення контрасту:
а) зображення з низьким контрастом; б) зображення з підвищеним контрастом

а) б)

Рисунок 1.2 – Обробка зображення для позбавлення зайвого шуму:
а) зображення з високим рівнем шуму; б) зображення з покращеним рівнем шуму

Також деякі методи визначають та виділяють об’єкти або регіони на зображенні, такі як обличчя, трава, небо тощо (рис. 1.3). Ця інформація може використовуватися для кращого призначення кольорів окремим об’єктам.

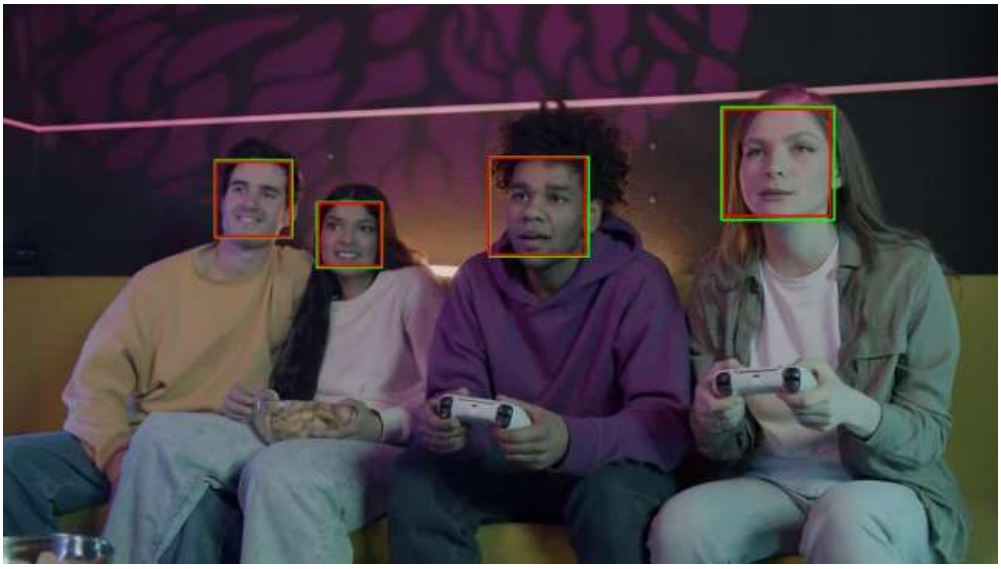

Рисунок 1.3 – Виділення обличь людей на зображенні

Основним пунктом для колоризації є вибір кольорів для різних об’єктів і регіонів на зображенні. Це може бути виконано вручну користувачем або автоматично на основі аналізу контексту та розпізнавання об’єктів. Наприклад, алгоритм «Colorization using optimization» [1] використовує призначення кольорів окремим ділянкам зображення для подальшої колоризації (рис. 1.4 [1]). Натомість, такий алгоритм як «Color2Embed» [2] використовує зображення зі схожим контекстом (рис. 1.5).

Вибір правильних кольорів важливий для досягнення реалістичних результатів. Проте, існують випадки, коли алгоритм може відпрацювати не зовсім точно, якщо йому не вистачає інформації. Для прикладу, на рисунку 1.6 опірним зображення для колоризації чорно-білого яблука було використано зображення неба, як ресурс з мінімальним контекстом. В результаті, можна побачити, що колоризація таки відбулася, проте, за відсутністю необхідного контексту, кінцеву діяльність алгоритму можна назвати помилковою.

а) б)

Рисунок 1.4 – Процес колоризації зображення з використанням алгоритму «Colorization using optimization»:
а) зображення з визначеними кольорами для певних ділянок; б) результат колоризації

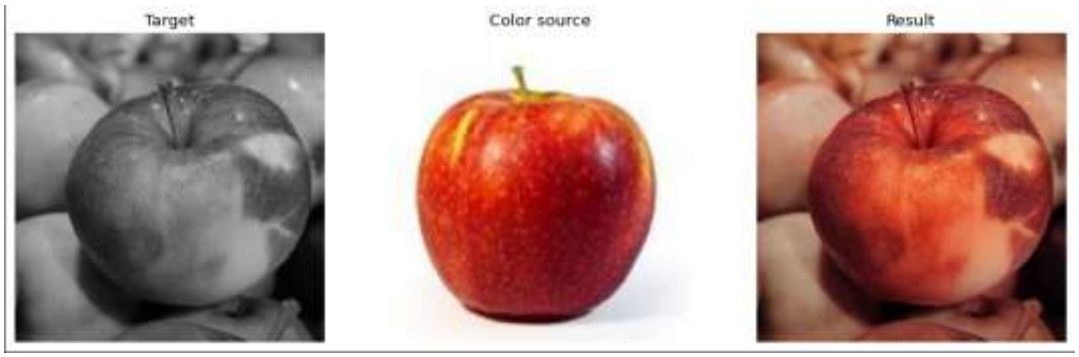

Рисунок 1.5 – Результат колоризації на базі зображення зі схожим контекстом

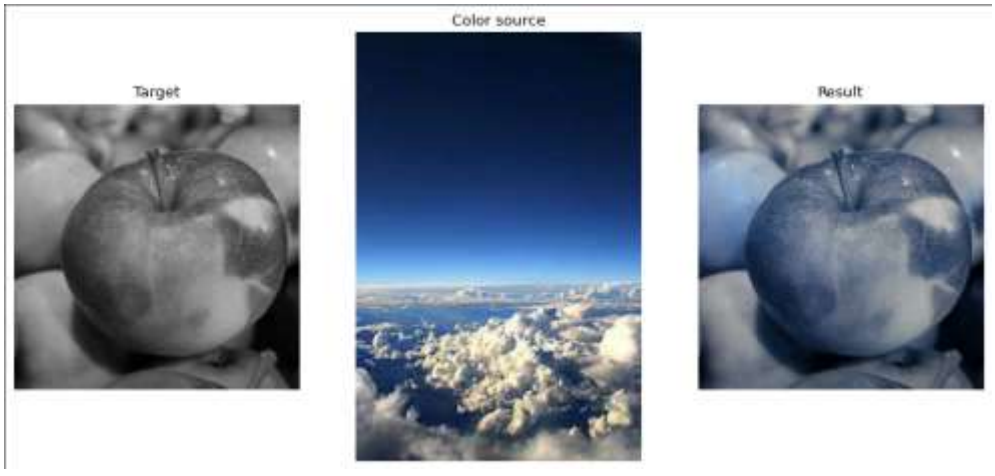

Рисунок 1.6 – Результат колоризації на базі зображення з некоректним контекстом

Багато методів враховують контекст зображення для кращого визначення кольорів. Цей підхід допомагає зробити колоризацію більш реалістичною, оскільки він враховує те, як кольори пов’язані з об’єктами та їхнім оточенням.

Наприклад, якщо на зображенні є об’єкти, які можуть слугувати контекстом для колоризації неба та трави, то підхід може враховувати це призначення кольорів. Тобто, якщо вказати траву внизу та небо вгорі, то колір трави може бути зеленим, а колір неба – синім.

Для старих або вінтажних фотографій, може використовуватися додатковий контекст, щоб вибрати кольори, характерні для певного історичного періоду. Наприклад, колір одягу, автомобілів та інтер’єрів може бути відповідним окремому часовому періоду.

1.1 Розпізнавання окремих об’єктів та фону на зображенні

Важливо правильно виділити об’єкти на передньому плані та фон, оскільки їм можуть призначатися різні кольори. Це необхідно для створення правдоподібних колоризованих зображень.

Розпізнавання окремих об’єктів та фону може бути реалізовано за допомогою різних алгоритмів та бібліотек машинного навчання [3 – 5].

Для сегментації об’єктів та фону на зображенні можна використовувати глибокі нейронні мережі, такі як Convolutional Neural Networks (CNN) [6] або Fully Convolutional Networks (FCN) [7]. Модель може бути навчена на великій кількості зображень, де об’єкти та фон позначені як різні класи. Потім модель здатна визначити, до якого класу належить кожен піксель на новому зображенні.

Іншим підходом може бути використання алгоритмів машинного навчання, таких як Random Forests [8] або SVM [9], з інженерією ознак для сегментації. У цьому випадку ознаки (наприклад, текстура або яскравість)

обчислюються для кожного пікселя, і модель навчається передбачати, чи відноситься піксель до об’єкта чи фону.

Деякі бібліотеки для обробки зображень, такі як OpenCV [10], містять функції для сегментації об’єктів. Вони можуть використовувати методи порогової обрізки, розмиття та знаходження контурів для виділення об’єктів на зображенні.

1.2 Збереження деталей і текстури зображення

Для того, щоб колоризовані зображення виглядали природними, важливо зберегти деталі та текстуру оригінального зображення. Методи надання кольору повинні враховувати контури та структуру об’єктів. Цей аспект означає, що колоризовані зображення повинні мати високу ступінь відтворення деталей та текстури, які були присутні на початковому чорно-білому зображенні.

Один з ключових аспектів збереження деталей – це відтворення контурів об’єктів на зображенні. Досягнення чіткого та точного відображення контурів допомагає зберегти деталі об’єктів, включаючи їх форму та розмір. Це питання було висвітлено вище та досі підлягає додатковому вивченню.

1.2.1 Текстура поверхні

Зображення може містити текстуру на поверхні об’єктів або фону. Під час колоризації важливо враховувати цю текстуру, щоб зробити зображення реалістичним. Наприклад, текстура деревини або каменю має бути правильно відтворена для того, щоб зберегти оригінальних контекст зображення.

Врахування та виявлення текстури поверхні на зображенні можуть бути здійснені різними методами та алгоритмами обробки зображень та комп'ютерного зору:

– фільтри, такі як фільтр Габора [11] (1.1) або фільтр текстурних характеристик, можуть бути використані для виявлення текстури на зображенні. Ці фільтри відділяють структурні та текстурні характеристики зображення, допомагаючи виділити текстуру поверхні (рис. 1.7 [12]);

– методи, які використовують локальні бінарні шаблони (LBP [13]), дозволяють аналізувати текстурні особливості на рівні пікселів. Вони визначають, як піксель відмінюється від своїх сусідів, що допомагає виявити текстурні зміни на поверхні;

– використання текстурних дескрипторів, таких як текстурні матриці Грея [14], може допомогти аналізувати текстуру поверхні. Ці матриці представляють статистичну інформацію про розподіл інтенсивності пікселів на зображенні;

– фрактальний аналіз використовує фрактальні властивості для аналізу текстур. Він може бути корисним для виявлення текстурних особливостей, які мають фрактальну структуру.

Зазвичай, в залежності від конкретного завдання, для визначення текстури на зображенні може бути застосовано один або комбінацію цих методів. Для колоризації зображень, наприклад, врахування текстури може бути використано для збереження інформації про текстурні особливості під час призначення кольорів.

$$g(z) = eg(z) = e^{-\frac{z^2}{2\sigma^2}} \cos(2\pi\theta z), \qquad (1.1)$$

де σ – стандартне відхилення гаусівського ядра, яке визначає амплітуду функції;

ω – частота коливань, яка визначається як $\omega = \frac{1}{T}$, де T – період функції $cos(2\pi\omega z)$.

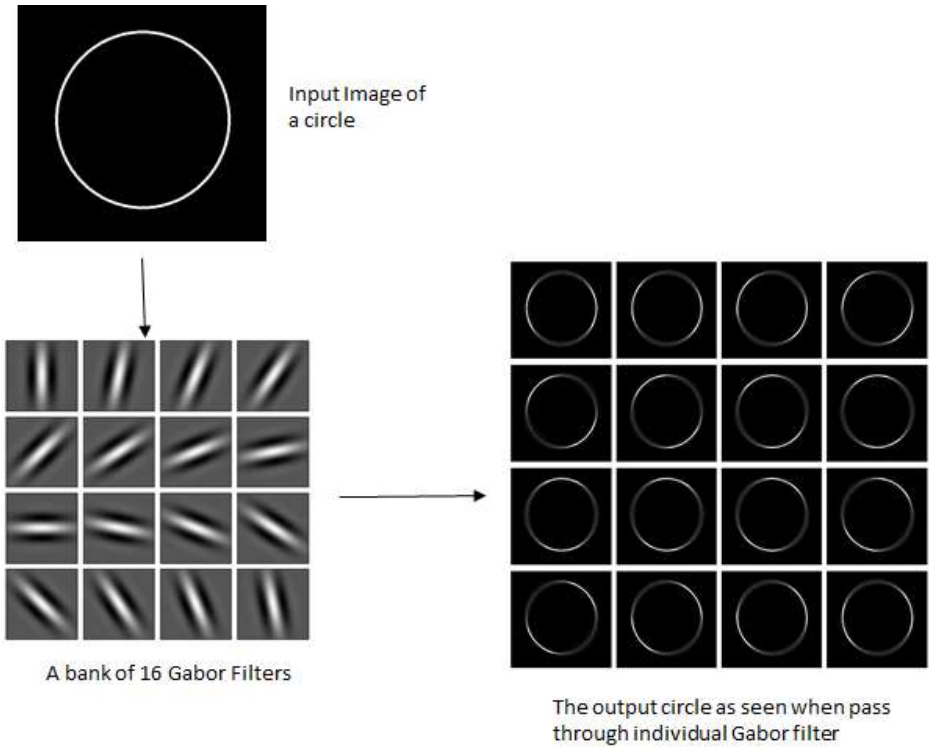

Рисунок 1.7 – Відповідні орієнтовані краєві особливості виявляються при проходженні через окремі орієнтовані фільтри Габора

1.2.2 Відтворення текстур

Для відтворення певної текстури може бути застосовано використання зразків текстур. Цей підхід передбачає використання існуючих текстурних зразків або зразків текстур, що були створені окремо, і їхнє застосування до колоризованого зображення. Це може бути зручним для відтворення конкретних текстур, таких як деревина, камінь, текстиль тощо.

Використання генеративних моделей, таких як генеративні зворотні мережі (GAN [15]), дозволяє створювати нові текстурні зразки на основі існуючих. GAN можуть навчитися генерувати текстури, які подібні до тих, які були б на оригінальних зображеннях.

Під час колоризації зображення текстуру можна поступово додавати, використовуючи різні алгоритми та методи. Наприклад, текстура може бути додана на окремому етапі обробки зображення. Для наочності, оригінальне зображення було відредаговано таким чином, що текстура та деталі не є достатньо чіткими, щоб їх можна було розгледіти, далі зображення було відредаговано повторно з метою покращення текстури, результати показано на рисунку 1.8. Звісно, що під час відтворення текстур, деякі деталі можуть бути втрачені або ж відтворені некоректно, але це не відміняє того факту, що покращення зображення допоможе отримати більш правдоподібний результат колоризації.

а) б)

в)

Рисунок 1.8 – Обробка зображення з різним рівнем деталізації:
а) оригінальне зображення; б) зображення з нечіткими деталями та текстурою; в) зображення з покращеною текстурою

Під час колоризації важливо адаптувати текстуру до контексту зображення. Наприклад, якщо текстура повинна залишатися на однорідній поверхні, то важливо враховувати геометричні особливості цієї поверхні.

Текстура повинна відображати вплив освітлення та тіней на поверхню. Для цього можуть використовуватися методи моделювання освітлення.

Текстура може включати як мікроструктури (наприклад, зерна, деревину), так і макроструктури (наприклад, смуги асфальту на дорозі). Важливо враховувати обидва типи структур для відтворення текстури.

Якщо зображення містить людей або об’єкти з одягом, то важливо зберегти деталі на тканинах та деталях одягу. Також важливо враховувати деталі на шкірі, такі як пігментні плями, зморшки і т. д.

Збереження деталей також включає в себе правильне відтворення рельєфу та освітлення на об’єктах. Це означає відтворення тіней, відблисків та висвітлення залежно від контексту та джерела світла на зображенні (рис. 1.9).

Рисунок 1.9 – На зображенні з поганим відтворенням текстур збереглися певні відблиски

Важливо відзначити, що точність відтворення текстури залежить від якості алгоритмів та моделей, що використовуються під час колоризації, а також від якості початкових зображень. У деяких випадках може бути

необхідним вручну або автоматично поправляти результати для досягнення бажаного відтворення текстури на колоризованому зображенні.

Важливо уникати артефактів, таких як розмазування чи шум під час колоризації, оскільки вони можуть зіпсувати деталі та текстуру.

Збереження деталей і текстури вимагає від колоризаційних методів використання точних алгоритмів та обробки, які здатні ретельно аналізувати і відтворювати важливі характеристики зображень. Зазвичай це досягається шляхом використання глибокого навчання або інших методів машинного зору, які дозволяють автоматично виявляти та відтворювати деталі на зображеннях.

1.2.3 Візуальна оцінка, корекція і постпроцесінг

Після завершення процесу колоризації важливо віддати належну увагу візуальній оцінці результатів. Це етап, на якому автор чи користувач, визначає, чи вдалося досягти бажаних результатів у плані відтворення кольору та текстури на зображенні.

Після колоризації зображення варто уважно розглянути отриману колоризовану версію. Аналіз зображення дозволяє оцінити, чи вдалося відтворити барви та текстуру так, як було задумано (рис. 1.10). Особливу увагу необхідно звернути на якість колоризації та відповідність кольорів на зображенні. Якщо кольори відрізняються від очікуваних, це може вимагати корекції.

Рисунок 1.10 – Результат колоризації зображення з недостатньою деталізацією

Текстурні характеристики також на відтвореному зображені мають бути збережені на задовільному рівні. Це означає, що деталі, такі як зерна, структура тканин чи текстура поверхні, мають бути відтворені точно та відповідати оригінальним деталям.

Якщо під час візуальної оцінки виявлені недоліки у відтворенні кольору або текстур, може бути необхідно додати виправлення. Це може включати в себе зміну кольорів, налаштування насиченості, яскравості та інших параметрів. Корекція допоможе покращити якість та вигляд зображення.

Важливо зберегти оригінальну версію чорно-білого зображення. Це дозволить порівнювати колоризовану версію з оригіналом і переконатися, що під час процесу колоризації був отриманий бажаний результат, або такий, що наближений до бажаного.

Візуальна оцінка та корекція грають важливу роль у створенні якісних колоризованих зображень, допомагаючи забезпечити їхню відповідність вимогам та задоволення очікувань щодо кольору та текстури.

1.3 Постановка задачі дослідження

Таким чином, об’єктом дослідження є методи колоризації напівтонових зображень, що включають в себе різноманітні підходи та алгоритми для надання кольору саме чорно-білим фотографіям.

Метою дослідження є вивчення та оцінка методів колоризації напівтонових зображень з використанням різних алгоритмів та технологій.

Головні цілі включають:

– аналіз існуючих методів колоризації і їх переваги та недоліки;
– реалізація програмного забезпечення для колоризації зображень;
– експериментальна оцінка та порівняння реалізованих методів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– аналіз існуючих методів колоризації і їх переваги та недоліки;

- реалізація програмного забезпечення для колоризації зображень;
- експериментальна оцінка та порівняння реалізованих методів;
- зробити висновки та порівняльний аналіз розроблених методів.

Очікуваним результатом дослідження є аналіз та порівняння методів колоризації напівтонових зображень, що дозволить зробити висновки щодо присутніх недоліків та можливих покращень. В перспективі з’явиться можливість поліпшити якість колоризованих зображень та розширити можливості їхнього використання в різних галузях.

2 ДОСЛІДЖЕННЯ МЕТОДІВ КОЛОРИЗАЦІЇ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

Методи колоризації на основі глибокого навчання можна поділити на три категорії: повністю автоматична колоризація, колоризація з використанням користувацьких малюнків та колоризація на основі прикладів. Для повністю автоматичної колоризації були розроблені алгоритми [16], які вдосконалювалися за допомогою навчання з вчителем та дозволяють безпосередньо перетворювати чорно-біле зображення на кольорове. Однак результати колоризації не завжди виходять задовільними.

Під час колоризації з використанням користувацьких малюнків необхідна додаткова робота з боку користувачів такого підходу. Це може бути корисним для задоволення індивідуальних запитів до колоризації. Недоліком цього підходу є те, що користувачам потрібно вкласти додаткові зусилля, вибираючи численні точки на конкретних ділянках чорно-білого зображення, що потребує додаткового часу та ресурсів. Некоректні малюнки або деталі можуть призвести до дефектів у кінцевих результатах. Крім того, для користувачів з недостатнім досвідом може бути важко забарвлювати велику кількість зображень.

2.1 Алгоритм Colorization using optimization

Алгоритм Colorization using optimization – це метод колоризації напівтонових зображень, який базується на задачі оптимізації. Основна мета цього алгоритму – призначити кольори пікселям на чорно-білому зображенні так, щоб результат виглядав природно та реалістично [1].

Основні етапи алгоритму Colorization using optimization:

– початкове чорно-біле зображення спершу сегментується на окремі області або регіони. Ця сегментація може бути виконана за допомогою алгоритмів комп’ютерного зору або ручного визначення областей [17];

– для кожного сегментованого регіону визначаються його характеристики, такі як форма, текстурні особливості і геометричні властивості. Це важливо для врахування контексту при призначенні кольорі;

– початкові кольори призначаються пікселям кожного регіону. Зазвичай ці кольори обираються на підставі вихідного кольорового зображення, якщо воно доступне, або на підставі стандартних кольорових палітр та правил;

– створюється функція вартості (Cost Function), яка оцінює, наскільки добре початкові кольори відповідають реальним кольорам на зображенні. Ця функція може включати в себе різні компоненти, такі як відстань в кольоровому просторі, текстурні властивості та геометричні обмеження. Задача оптимізації полягає у знаходженні набору кольорів, які мінімізують цю функцію вартості;

– оптимізація функції вартості виконується з метою зміни призначених кольорів так, щоб досягти мінімального значення функції. Це може бути здійснено різними методами оптимізації, такими як градієнтний спуск або методи, специфічні для задач оптимізації на графах;

– після оптимізації кольорів може бути виконане додаткове уточнення та відновлення кольорів для поліпшення якості колоризації. Цей крок може включати в себе фільтрацію, реставрацію та інші обробки.

Алгоритм Colorization using optimization вимагає встановлення правил та параметрів для функції вартості та методів оптимізації. Він дозволяє здійснити точну колоризацію та надає певний контроль над процесом, але може бути більш обчислювально вимогливим порівняно з деякими іншими методами. Приклад роботи алгоритму показано на рисунку 2.1 [1].

а) б)

Рисунок 2.1 – Робота алгоритму Colorization using optimization:
а) зображення, підготоване до опрацювання алгоритмом; б) результат дії алгоритму

Як було зазначено вище, необхідно накласти певне обмеження на те, що двом сусіднім пікселям буде призначено один колір за умови, якщо їхня інтенсивність є рівною. Таким чином буде мінімізовано різницю між кольором *U(r)* у пікселі *r* і середньозваженим кольором у сусідніх пікселях (2.1).

$$J(U) = \sum_r \left(\bigcup_{(r)} - \sum_{S \in N(r)} \omega_{rs} u(s) \right)^2, \quad (2.1)$$

де ω_{rs} – вагова функція, яка дорівнює одиниці, велика, коли *U(r)* подібна до *U(s)* і мала, коли дві інтенсивності різні.

Як вже було сказано, цей алгоритм має певні переваги, оскільки надає більшої свободи в питанні надання кольору, якщо це необхідно зробити за індивідуальними запитами. Проте він все ж таки потребує багато часу та додаткових зусиль для процесу колоризації.

2.2 Згорткові нейронні мережі

Згорткові нейронні мережі (Convolutional Neural Networks) є потужним інструментом у сфері комп'ютерного бачення і зображення, вони знайшли широке застосування в задачах колоризації напівтонових зображень [6], зокрема, в обробці великого обсягу візуальної інформації завдяки своїй здатності автоматично визначати та виділяти значущі ознаки зображень.

Основні компоненти згорткової нейронної мережі для колоризації зображень включають:

– згорткові шари (Convolutional Layers): ці шари використовуються для виявлення різних рис об'єктів на зображенні. Згорткові фільтри просуваються по вхідному зображенні і виконують операції згортки для виділення особливостей;

– пулінг-шари (Pooling Layers): пулінг допомагає зменшити розмір вихідного зображення, зберігаючи важливі ознаки. Зазвичай використовується максимальний пулінг;

– повнозв'язані шари (Fully Connected Layers): повнозв'язані шари використовуються для розпізнавання засобами нейронної мережі, які кольори відповідають кожному пікселю на вихідному зображенні;

– функції активації: для нейронів в мережі використовуються активаційні функції, такі як ReLU (Rectified Linear Activation) [18], що допомагають уникнути лінійності і додати нелінійність до моделі;

– функцію втрат (Loss Function): функція втрат обчислює різницю між передбаченими кольорами і справжніми кольорами на зображенні. Зазвичай використовуються функції втрат, такі як Mean Squared Error (MSE) [19];

– оптимізатор: для навчання мережі використовуються оптимізатори, такі як Adam [20] або SGD [21], для мінімізації функції втрат.

Процес колоризації відбувається таким чином:

Крок 1. Вхідне чорно-біле зображення подається на вхід мережі.

Крок 2. Згорткові шари та пулінг-шари використовуються для виділення особливостей на зображенні.

Крок 3. Повнозв’язані шари генерують прогнози кольорів для кожного пікселя на вихідному зображенні.

Крок 4. Функція втрат порівнює прогнози з справжніми кольорами.

Крок 5. Оптимізатор використовується для корекції ваг мережі з метою мінімізації втрат.

Крок 6. Процес навчання повторюється протягом кількох епох до досягнення задовільного результату.

Візуальне представлення процесу роботу згорткової нейронної мережі наведено на рисунку 2.2 [22].

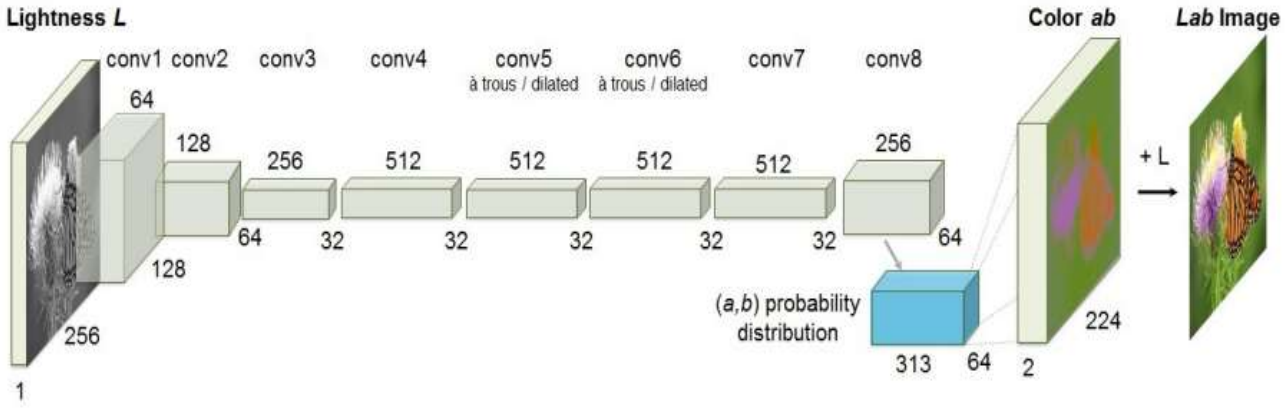

Рисунок 2.2 – Схематичне зображення принципу дії CNN

Згорткова нейронна мережа дозволяє комп’ютерам ефективно працювати з великими обсягами візуальної інформації. Що дозволяє використовувати її в обробці зображень без великих затрат часу. На рисунку 2.3 [22] наведено приклад колоризації чорно-білого зображення з допомогою згорткових нейронних мереж. Згорткова нейронна мережа дозволяє автоматично визначати відтінки та текстурні характеристики об’єктів на зображенні, що робить її потужним інструментом у візуальному аналізі.

Рисунок 2.3 – Зображення до та після колоризації використовуючи CNN

2.3 Генеративні зворотні мережі

Генеративні зворотні мережі (GAN [15]) є типом глибоких нейронних мереж, які були вперше представлені в 2014 році Яном Лекуном та його колегами. GAN вирізняється своєю нейронною архітектурою, яка включає два основних компоненти: генератор та дискримінатор. Ці компоненти навчаються одночасно шляхом конкурентного змагання, і це змагання допомагає генератору створювати основу для попередніх даних. Їх можна описати наступним чином:

– генератор – це нейронна мережа, яка призначена для створення нових даних. У випадку GAN для колоризації зображень, генератор намагається генерувати кольорові версії чорно-білих зображень;

– дискримінатор – це інша нейронна мережа, яка діє як суддя. Він призначений для визначення, наскільки дані є правдоподібними чи справжніми. Дискримінатор намагається розрізняти справжні дані від тих, що згенеровані генератором.

Генератор і дискримінатор конкурують у мінімізації спільної функції втрати. Для генератора ця функція втрати оцінює, наскільки добре він зміг

обманути дискримінатор, тобто наскільки генеровані зображення схожі на справжні. Для дискримінатора функція втрати визначає, наскільки точно він розрізняє справжні дані від згенерованих.

Під час тренування GAN, генератор і дискримінатор взаємодіють альтернативно для досягнення оптимальності. Генератор старається мінімізувати функцію втрати, зокрема, зменшити розрив між його вихідними зображеннями та реальними зображеннями. З іншого боку, дискримінатор прагне максимізувати функцію втрати, розрізняючи між згенерованими та реальними зображеннями. Ця взаємодія створює конкурентну динаміку, що сприяє поліпшенню якості генерованих зображень під час тренування.

Цей процес змушує генератор і дискримінатор досягати балансу між створенням реалістичних зображень та їх правильною ідентифікацією. Спільна оптимізація цих двох компонентів допомагає досягти оптимальних результатів у генерації зображень за допомогою GAN.

На рисунку 2.4 наведено схематичне зображення роботи GAN [23].

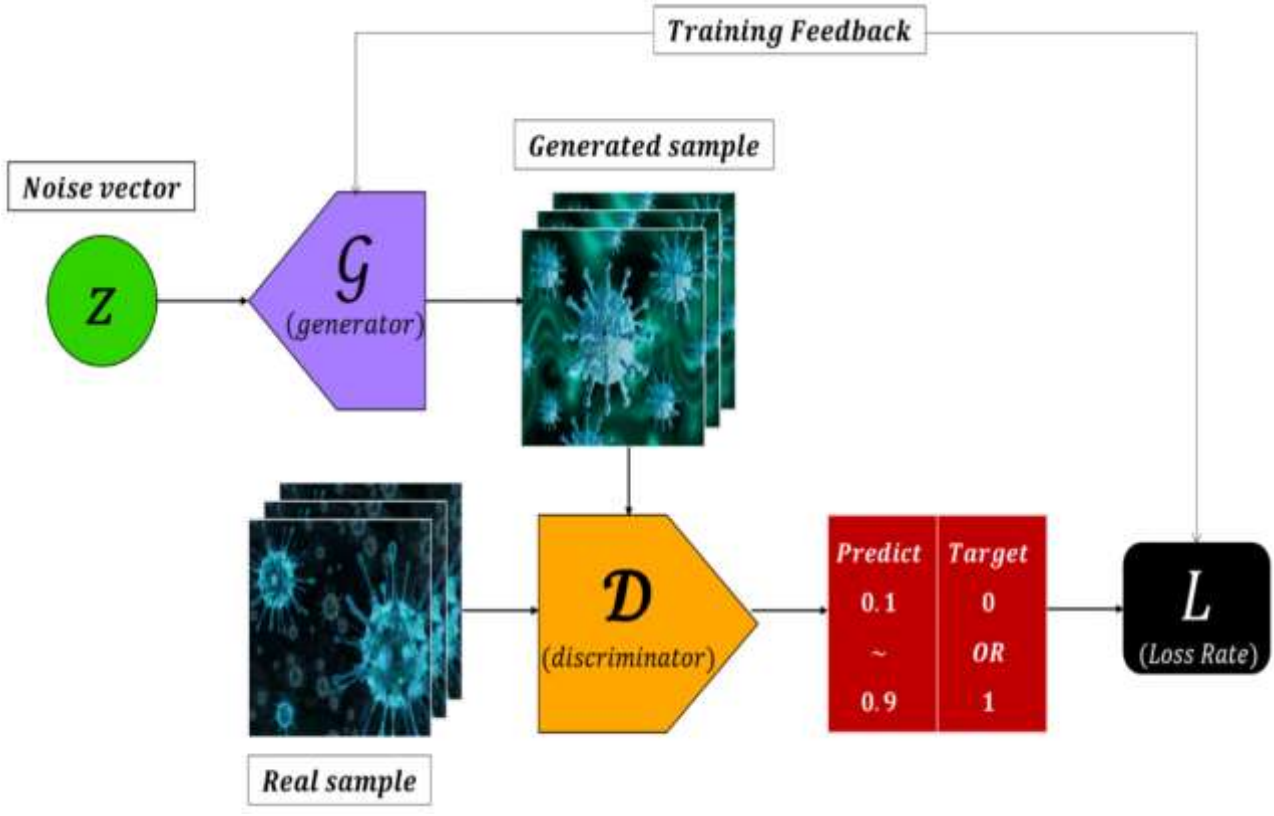

Рисунок 2.4 – Принцип дії GAN під час колоризації зображення

2.4 Генеративна зворотна мережа з варіаційним автокодером

Генеративна зворотна мережа з варіаційним автокодером (Variational Autoencoder Generative Adversarial Network, VAE-GAN [24]) це глибока нейронна мережа, яка використовується для колоризації чорно-білих зображень. Вона поєднує в собі дві потужні моделі: варіаційний автокодер (VAE) і генеративну зворотну мережу (GAN), для досягнення високої якості колоризації зображень.

Як видно з назви, на відміну від попередньо описаної генеративної зворотної мережі, дана мережа містить у собі варіаційний автокодер (VAE):

– encoder: цей компонент перетворює вхідні чорно-білі зображення на латентний простір, де кожен латентний вектор представляє собою певний аспект зображення. Encoder генерує середнє значення і дисперсію для кожного латентного вектора;

– репараметризація: для вибору латентного вектора з відповідного розподілу, використовується репараметризація. Ця операція дозволяє здійснювати градієнтне навчання VAE;

– decoder: decoder перетворює латентний вектор назад у чорно-біле зображення. Він намагається відтворити вхідне зображення з латентного простору.

VAE навчається максимізувати ліквідність відтвореного зображення та мінімізувати відстань між розподілом латентних векторів від encoder і певним апріорним розподілом.

По завершенню навчання VAE-GAN можна використовувати encoder для перетворення чорно-білих зображень в латентний простір і generator для отримання колоризованих версій цих зображень. Такий підхід дозволяє отримувати реалістичні колоризовані зображення з використанням інформації, отриманої з чорно-білих зображень. На рисунку 2.5 зображено схематичне представлення принципу дії VAE-GAN [24].

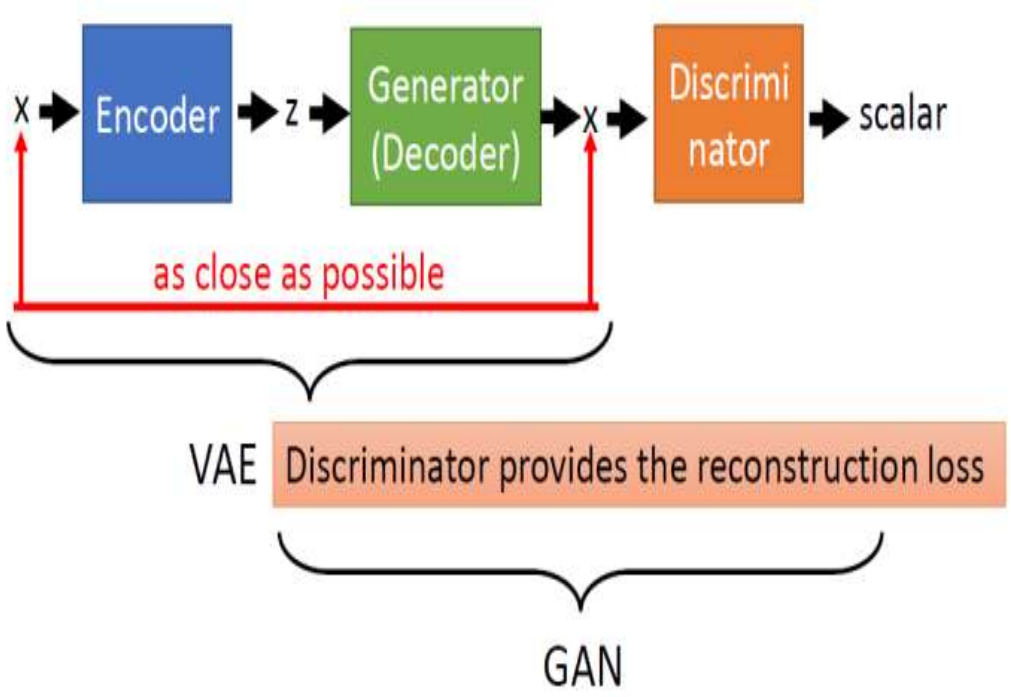

Рисунок 2.5 – Принцип дії нейронної мережі типу VAE-GAN

2.5 Автоматизована колоризація на основі класифікації

Автоматизована колоризація на основі класифікації – це метод колоризації напівтонових зображень, який використовує класифікацію об’єктів на зображенні для автоматичного призначення кольорів цим об’єктам. Основна ідея полягає в тому, щоб спочатку визначити, які об’єкти або регіони присутні на зображенні, а потім використовувати класифікацію, щоб визначити кольори для кожного об’єкта [25].

Автоматизована колоризація на основі класифікації поєднує в собі концепції машинного навчання, глибокого навчання та комп’ютерного зору для колоризації чорно-білих або напівтонових зображень.

Спершу, зображення сегментується на окремі області або регіони [26, 27]. Це важливий крок, оскільки він дозволяє визначити, які частини зображення відповідають різним об’єктам або фону.

Після сегментації об’єкти або регіони класифікуються. Кожному об’єкту присвоюється клас або категорія, яка вказує на його тип (наприклад, «хмари»,

«дерево», «автомобіль», «одяг», «рослина» тощо). Це може бути досягнуто за допомогою навченої моделі, яка розпізнає об’єкти на зображенні та визначає їхні характеристики.

Для кожного класу об’єктів визначається кольорова палітра або набір можливих кольорів, які відповідають цьому класу. Ця інформація визначає, які кольори можуть бути призначені об’єктам під час колоризації. Кольорову палітру можна визначати вручну або автоматично на основі навчання на великій кількості зображень.

За допомогою класифікації та кольорової палітри можна автоматично призначити кольори кожному об’єкту на зображенні. Зазвичай це робиться за допомогою алгоритмів глибокого навчання, які враховують контекст об’єкта та його оточення для надання більш реалістичних результатів.

Деякі методи також враховують текстурні особливості та контекст зображення, щоб зробити колоризацію більш природньою. Наприклад, колір пікселя може залежати від кольору сусідніх пікселів, що допомагає забезпечити гладкі переходи між кольорами.

Деякі системи також надають можливість втручання користувача, де він може вручну призначити або змінити кольори об’єктів на зображенні.

Автоматизована колоризація на основі класифікації дозволяє ефективно перетворити чорно-білі зображення у кольорові, зберігаючи контекст та реалістичність кольорів об’єктів на зображенні.

Існує декілька прикладів реалізації автоматизованої колоризації на основі класифікації. Цей метод використовується в різних сервісах та проєктах.

2.5.1 Автоматична колоризація в Adobe Photoshop

Деякі версії Adobe Photoshop мають функцію автоматичної колоризації, яка використовує технології машинного навчання для визначення об’єктів та

призначення їм кольорів. Ця функція робить процес колоризації зображень більш швидким та доступним для користувачів без глибоких знань у галузі обробки зображень [27 – 29].

Adobe Photoshop використовує інтелектуальні алгоритми, засновані на машинному навчанні, щоб визначити оптимальні кольори для кожного об’єкта на зображенні. Ці алгоритми навчалися на великих наборах даних з кольоровими та чорно-білими зображеннями, щоб розуміти, які кольори найкраще підходять для різних об’єктів.

Функція автоматично розпізнає об’єкти на зображенні [30 – 32]. Вона може визначати такі об’єкти, як обличчя, пейзажі, тварини та інші деталі на зображенні.

Після розпізнавання об’єктів функція автоматично призначає кольори для кожного об’єкта на основі навчання на основі даних. Вона може враховувати текстурні особливості об’єктів і контекст зображення, щоб забезпечити більш реалістичні результати.

Adobe Photoshop також надає можливість користувачам додатково налаштовувати та коригувати відповідно до власних побажань результати автоматичної колоризації. Користувач може змінювати кольори об’єктів, додавати фільтри, змінювати насиченість і контрастність та вносити інші покращення до кінцевого результату.

Крім того, функція автоматичної колоризації може бути застосована не тільки до окремих фотографій, але й до відеороликів. Це дозволяє колоризувати великий обсяг рухомих зображень, таких як старі фільми.

Після проведення автоматичної колоризації, оригінальне чорно-біле зображення залишається без змін. Користувач може зберегти як колоризовану, так і оригінальну версію зображення (рис. 2.6 [33]).

Рисунок 2.6 – Колоризація в Adobe Photoshop

2.5.2 DeOldify

DeOldify – це відкритий проєкт з метою відновлення та колоризації старих чорно-білих фотографій і фільмів за допомогою глибокого навчання та штучного інтелекту. Він створений з використанням різноманітних моделей глибокого навчання та алгоритмів, і він надає засоби для відновлення колірних деталей на чорно-білих зображеннях та фільмових фрагментах [34].

DeOldify використовує архітектуру GAN [15] для колоризації зображення. В ньому є генератор, який додає кольори критику (дискримінатору), метою якого є критика колоризації, створеної генератором. В проекті використовується спеціальний метод навчання GAN під назвою No-GAN [35].

Автори використовують архітектуру U-Net [36] для генератора, модифікуючи її для використання спектральної нормалізації в моделі.

No-GAN: Це метод навчання GAN, розроблений авторами моделі. Основна ідея за цією моделлю – отримати переваги навчання GAN, витрачаючи мінімальний час на пряме навчання.

DeOldify намагається автоматично визначити, які кольори були на оригінальному зображенні, і надає можливість забезпечити призначені кольори та параметри процесу колоризації [27].

Проєкт також включає в себе інтерактивні інструменти, призначені для взаємодії з користувачами та надання їм можливості активно впливати на процес колоризації. Зокрема, у користувачів є можливість вручну визначати кольори для окремих об’єктів, що надає додатковий рівень керованості та творчої свободи в ході редагування зображень.

Однією з ключових особливостей DeOldify є його можливість ефективно застосовувати колоризацію до старих чорно-білих фільмів і відео. Це відкриває нові перспективи відновлення колірної інформації в рухливих зображеннях, дозволяючи зробити старі фільми більш привабливими і актуальними для сучасних глядачів. DeOldify стає потужним інструментом для відновлення та оновлення історичного візуального контенту.

Проєкт DeOldify є відкритим і має активну спільноту розробників і користувачів. Це дозволяє спільноті вдосконалювати і розширювати функціональність проекту [37].

Він широко використовується ентузіастами, істориками та архіваріями для відновлення та відображення минулих подій у барвистому світлі (рис. 2.7 [34]).

Рисунок 2.7 – Результат роботи DeOldify

2.6 Color2Embed

Що стосується колоризації на основі прикладів, то її продуктивність дещо більша, ніж у випадку повністю автоматичної колоризації, завдяки використанню додаткової інформації [38].

Проте існують дві проблеми, пов’язані з реалізацією колоризації на основі прикладів. Перша проблема полягає в тому, як ефективно підготувати базу даних з посиланнями на приклади. Оскільки база даних, що використовується в колоризації зображень, завжди велика (наприклад, ImageNet [39]), знаходити відповідне та правильне посилання на приклад для кожного вхідного зображення є завданням із великими затратами часу та інших ресурсів.

Друга проблема полягає в тому, що для вхідного чорно-білого зображення відсутній кольоровий зразок для побудови пари для навчання з наглядом. Деякі методи [40] використовують попередньо навчений алгоритм витягування сірих зображень, щоб побудувати велику базу даних із зображеннями для посилань. Очевидно, це дуже часомістке та важкозрозуміле завдання. Більше того, якщо ми бажаємо провести експерименти з іншими наборами даних, то попередній алгоритм посилань повинен бути повторно навчений.

Ці виклики роблять реалізацію колоризації на основі прикладів складною та витратною справою.

Запропонований метод [38] складається з трьох основних модулів: мережі генерації вбудовувань кольору, мережі генерації вбудовувань контенту та мережі поступової формалізації ознак (Progressive Feature Formalisation Network – PFFN [38]):

– мережа генерації вбудовувань кольору: перший модуль витягує високорозмірні ознакові карти з наданого кольорового зображення в просторі RGB, після чого глибокі ознакові карти проходять через створений

багатошаровий персептрон (MLP) [41], щоб згенерувати абстрактні вбудовування кольору;

– мережа генерації контенту: другий модуль призначений для кодування цільового чорно-білого зображення в проміжні глибокі ознаки;

– мережа поступової формалізації ознак (PFFN): останній модуль приймає вбудовування контенту та кольору як вхідні дані для генерації кольорового зображення з використанням декількох блоків поступової формалізації ознак (PFFB). Ці блоки призначені для впровадження посилань вбудовувань кольору в процес відтворення.

Загалом, запропонований метод використовує комбінацію мереж для генерації вбудовувань кольору та контенту, а також для відтворення кольорового зображення з урахуванням вбудовувань кольору. Цей підхід дозволяє отримати колоризовані зображення з урахуванням інформації про кольори з посилань та контент з чорно-білих зображень (рис. 2.8).

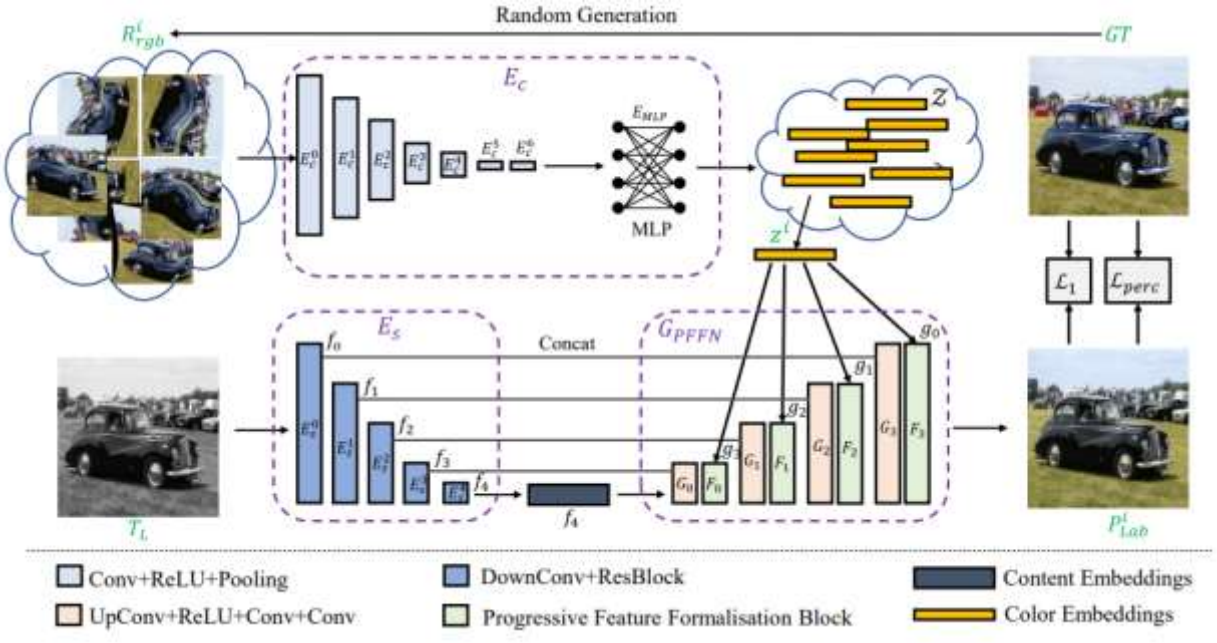

Рисунок 2.8 – Ілюстрація запропонованого методу, що висвітлює процес колоризації зображень [38]

2.7 Text2Color

Цей метод [42] може створювати різноманітні палітри, при наданні текстової інформації. Користувачі можуть за бажанням обирати, яку палітру використовувати для кінцевого результату колоризації.

Text2Colors складається з двох мереж: мережі генерації тексту в кольори (Text-to-Palette Generation Networks – TPN) і мережі колоризації на основі палітр (Palette-based Colorization Networks – PCN). Процес полягає в навчанні першої мережі генерувати колірні палітри на основі багатослівного тексту, а потім у навчанні другої мережі передбачати процеси колоризації на основі чорно-білого зображення та згенерованих палітр. Для обох мереж використовуються умовні генеративні зворотні мережі (cGAN) [42, 43].

Мережа TPN генерує розумні кольорові палітри, пов’язані з текстовим вводом. Для цього використовуються вектори слів, ініціалізовані за допомогою 300-вимірних попередньо навчених векторів з GloVe [44]. Слова, які не входять до набору попереднього навчання, ініціалізуються випадково.

Після кодування GRU x в приховані стани $h = \{h1, \cdots, hT\}$, додається випадковий шум до закодованого представлення тексту, зразок відповідних змінних $\hat{c}$ відбувається за допомогою гаусівського розподілу $N(\mu(h), \Sigma(h))$. Послідовність умовних векторів $\hat{c} = \{\widehat{c_1}, \cdots, \widehat{c_T}\}$ використовується як умова для генератора для виходу палітри $\hat{y}$, тоді як її середній вектор $\bar{C} = \frac{1}{T}\Sigma_{i=1}^{T}\hat{C}$ використовується як умова для дискримінатора. Об’єктивна функція першої cGAN може бути виражена як

$$L_{D_0} = E_{y\sim p}[\log D_0(\bar{c}, y)] + E_{x\sim p}\left[\log\left(1 - D_0(\bar{c}, y)\right)\right], \qquad (2.2)$$

де дискримінатор D_0 намагається максимізувати L_{D_0} проти генератора G_0, який намагається мінімізувати L_{G_0}. Попередньо навчені вектори слів x та реальну кольорову палітру y вибирають із реального розподілу даних $Pdata$) [42, 43].

Вивчення відображення від тексту до кольору властиво багатомодальному характеру. Наприклад, текст «осінь» може бути відображений в різні правдоподібні кольорові палітри. Якщо текст стає довшим, наприклад, «від середини літа до осені» або «осінній вітер і опадання листя», можливість відповідних палітр стає більш широкою та різноманітною [42]. Для належного моделювання багатомодальності задачі використовується техніка умовного збагачення (Conditioning Augmentation – CA) [45]. Замість використання фіксованої послідовності закодованого тексту як вхід до нашого генератора, ми випадковим чином вибираємо латентний вектор $\hat{c}$ із гаусівського розподілу $N(\mu(h), \Sigma(h))$. Ця випадковість дозволяє моделі генерувати кілька правдоподібних палітр за однакового текстового вводу (рис. 2.9 [42]).

Рисунок 2.9 – Представлення згенерованої палітри кольорів моделі TPN

Метод являє собою генеративну модель, яка може створювати кілька палітр із багатою текстовою інформацією та колоризувати чорно-білі зображення за допомогою згенерованих палітр. Результати оцінки підтверджують, що TPN може генерувати правдоподібні кольорові палітри на основі текстового вводу та може враховувати багато різних кольорів. Результати PCN також показують, що різноманітні кольори в палітрі ефективно відображаються в результатах колоризації.

3 ОГЛЯД ТА ТЕСТУВАННЯ МЕТОДІВ КОЛОРИЗАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

Методи колоризації чорно-білих зображень стали важливою галуззю комп'ютерного зору та обробки зображень. У цьому контексті Python, зокрема бібліотека PyTorch, виявляється потужним інструментом для реалізації таких методів. Python не лише забезпечує зручний та ефективний синтаксис програмування, але і володіє багатофункціональністю, необхідною для складних завдань обробки зображень.

Ще однією перевагою використання PyTorch є його глибока інтеграція з нейронними мережами. Модуль torch.nn дозволяє легко визначати та тренувати складні моделі глибокого навчання для колоризації зображень. Крім того, PyTorch надає різноманітні оптимізаційні інструменти, такі як автоматичне обчислення градієнтів, що полегшує процес навчання та підвищує продуктивність в розробці алгоритмів колоризації. Отже, використання PyTorch для реалізації методів колоризації навпаки зображень обґрунтоване й дозволяє здійснити якісні та швидкі рішення завдань з обробки та аналізу зображень.

Вибір між PyCharm та іншими інтегрованими середовищами розробки (IDE) для реалізації методів колоризації залежить від індивідуальних вподобань розробника та вимог конкретного проєкту. PyCharm, як інші інструменти, має свої переваги, але кінцевий вибір може бути обумовлений рядом чинників.

PyCharm, створений для роботи з Python, надає ряд переваг для розробників, які працюють у цій мові програмування:

– підтримка Python: PyCharm розроблений спеціально для мови програмування Python і надає повноцінну підтримку цієї мови, включаючи автодоповнення коду, рефакторинг, аналіз коду та інші корисні функції;

– інтеграція з іншими інструментами: PyCharm легко інтегрується з різноманітними іншими інструментами та бібліотеками, що часто використовуються в проєктах Python, такими як NumPy, SciPy, Matplotlib та інші;

– система керування версіями: PyCharm підтримує роботу з різними системами керування версіями, такими як Git, що робить спільну роботу над проєктами більш зручною;

– спеціальні інструменти для аналізу та відлагодження коду: IDE має вбудовані інструменти для аналізу та відлагодження коду, що полегшує виявлення та усунення помилок;

– просте встановлення та використання: всі необхідні бібліотеки дуже просто встановити для використання.

3.2 Огляд вимог та реалізація методів

Різні методи колоризації потребують різні бібліотеки для набори даних. Від цього може залежати ефективність та швидкість роботи на різних типах даних. Деякі алгоритми можуть добре працювати на зображеннях природних сцен, тоді як інші можуть бути ефективні для аналізу медичних зображень чи відео. Також різні завдання можуть вимагати специфічного підходу та налаштувань.

Для обробки та порівняння методів було обрано декілька зображень з різним контекстом: їжа (рис. 3.1), тварини (рис. 3.2), люди (рис. 3.3), рослини (рис. 3.4) та зображення міста (рис. 3.5). На рисунках 3.1 – 3.5 зображення представленні в оригінальному (а) та монохромному (б) вигляді. Кожен метод буде порівнюватися за швидкістю обробки, складністю попередньої підготовки та якістю кінцевого результату.

а) б)

Рисунок 3.1 – Зображення їжі:

а) оригінальне кольорове зображення; б) чорно-біле зображення

а) б)

Рисунок 3.2 – Зображення тварини:

а) оригінальне кольорове зображення; б) чорно-біле зображення

а) б)

Рисунок 3.3 – Зображення людини:

а) оригінальне кольорове зображення; б) чорно-біле зображення

а) б)

Рисунок 3.4 – Зображення рослини:

а) оригінальне кольорове зображення; б) чорно-біле зображення

а) б)

Рисунок 3.5 – Зображення міста:
а) оригінальне кольорове зображення; б) чорно-біле зображення

3.2.1 Метод Colorization using optimization

У цьому методі використовуються бібліотеки OpenCV, Pillow, scikit-image, tqdm, scipy та matplotlib:

– openCV (Open Source Computer Vision Library) – це бібліотека для комп’ютерного зору та обробки зображень. Вона реалізує набір інструментів для роботи з зображеннями і відео, включаючи обробку, аналіз, виявлення об’єктів, розпізнавання обличчя, а також роботу з камерами та потоковим відео [8];

– pillow – це форк бібліотеки Python Imaging Library (PIL), яка надає простий інтерфейс для завантаження, збереження та обробки зображень в різних форматах. Вона підтримує основні операції зі зображеннями, такі як зміна розміру, обрізка, зміна кольорів тощо;

– scikit-image – це колекція алгоритмів для обробки та аналізу зображень, яка базується на бібліотеці SciPy. Вона містить реалізації багатьох методів обробки зображень, таких як фільтрація, сегментація, вимірювання об’єктів, а також інші інструменти для роботи з зображеннями;

– tqdm – це бібліотека для створення стилізованих прогрес-барів у консольних програмах Python. Вона надає зручний спосіб візуалізації прогресу виконання довгих задач, таких як цикли обробки даних або обчислення;

– sciPy – це бібліотека для наукових обчислень в Python. Вона включає багато модулів для розв’язання різних завдань, таких як оптимізація, інтеграція, обробка сигналів, обробка зображень, статистика тощо. SciPy часто використовується в комбінації з NumPy для роботи з числовими даними.

На першому етапі зчитуються градації сірого (gI) та кольорове зображення (cI), які пізніше конвертуються у простір кольорів NTSC за допомогою функції rgb2ntsc. Це перетворення дозволяє працювати з зображеннями у форматі YIQ, де Y – яскравість, а I та Q – компоненти кольору.

Лістинг 3.1 Реалізація зчитування градацій сірого та кольорового зображення:

```
gI = imread(g_name) / 255.0
cI = imread(c_name) / 255.0
sgI = rgb2ntsc(gI)
scI = rgb2ntsc(cI)
```

Далі створюється бінарне зображення colorIm, де позначені області, де колоризація потрібна. Це визначення ґрунтується на аналізі абсолютної різниці між градаціями сірого та кольорового зображень.

Лістинг 3.2 Реалізація створення бінарного зображення:

```
colorIm = (np.sum(abs(gI - cI), axis=2) > 0.01)
nI = getColorExact(colorIm, ntscIm)
```

Функція getColorExact викликається для кожного пікселя, використовуючи оптимізаційний підхід. Оптимізація включає вирішення

системи лінійних рівнянь для визначення значень пікселів колоризованого зображення. Кожен піксель розглядається з урахуванням його локального контексту.

Зображення конвертуються назад до простору кольорів RGB за допомогою функції ntsc2rgb. Результат відображається та зберігається у файлі.

Лістинг 3.3 Зворотня конвертація зображення:

```
nI = ntsc2rgb(nI)
plt.imshow(nI)
plt.show()
plt.imsave(out_name, nI)
```

Опис функції getColorExact:

Крок 1. Створюються матриці та вектори для подальшого використання у внутрішній логіці функції.

Лістинг 3.4 Ініціалізація та підготовка даних:

```
nI = np.zeros_like(ntscIm)
nI[:, :, 0] = ntscIm[:, :, 0]
m = ntscIm.shape[0]
n = ntscIm.shape[1]
img_size = m * n
img_mat = np.arange(img_size)
lbl_idxs = img_mat.reshape(m, n)[np.where(colorIm)]
lbl_idxs.sort()
img_mat = img_mat.reshape(m, n)
```

Крок 2. Функція розглядає кожен піксель та враховує його локальний контекст для оптимізації процесу колоризації. Обмеження додаються до системи лінійних рівнянь, враховуючи області, де колоризація не потрібна.

Лістинг 3.5 Побудова системи лінійних рівнянь та оптимізація:

```
for i in range(m):
    for j in range(n):
        if not colorIm[i, j]:
        else:
            row_inds[length] = consts_len
            col_inds[length] = img_mat[i, j]
            vals[length] = 1
            length += 1
            consts_len = consts_len + 1
```

Крок 3. Створюється розріджена матриця A, яка описує систему лінійних рівнянь. Розв’язанням цієї системи є значення пікселів колоризованого зображення.

Лістинг 3.6 Розв’язання системи лінійних рівнянь:

```
A = sparse.coo_matrix((vals, (row_inds, col_inds)), shape=(consts_len, img_size)).tocsr()
    b = np.zeros(A.shape[0])
    for t in range(1, 3):
        curIm = ntscIm[:, :, t]
        b[lbl_idxs] = curIm.reshape(img_size)[lbl_idxs]
        new_vals = sparse.linalg.spsolve(A,  b)
        nI[:, :, t] = np.reshape(new_vals, (m, n))
```

Для застосування цього методу необхідно додатково обробити зображення, як було описано в підрозділі 2.1. Приклади попередньої обробки надано на рисунках 3.6 – 3.10.

Рисунок 3.6 – Попередня обробка монохромного зображення їжі

Рисунок 3.7 – Попередня обробка монохромного зображення тварини

Рисунок 3.8 – Попередня обробка монохромного зображення людини

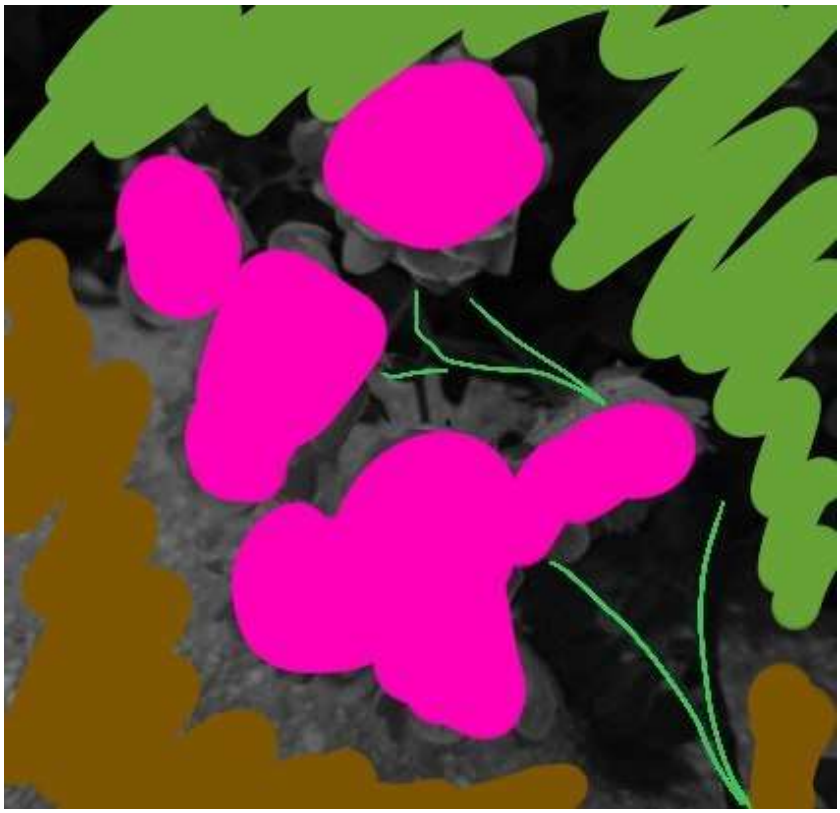

Рисунок 3.9 – Попередня обробка монохромного зображення рослини

Рисунок 3.10 – Попередня обробка монохромного зображення міста

Як можна побачити, під час обробки не завжди точно відтворюються необхідні кольори. До того ж, це досить часомістка задача. Хоча необхідність власноруч вказувати кольори на зображенні також є досить корисним способом колоризації, адже за таких умов можна покращувати вже колоризовані зображення. Наприклад, додавати необхідні деталі.

3.2.2 Метод Color2Embed

Цей метод призначений для тренування моделі з колірними автокодерами та U-Net [29]. Він включає в себе оптимізацію з допомогою Adam, обчислення втрат на основі L1-втрати та втрати ознак. Результати тренування можуть зберігатися для подальшого використання.

Для реалізації використовувалися наступні власні модулі:

– mkdirss: створення каталогів;

– data_sampler: визначення семплера для даних;

– requires_grad: встановлення параметра requires_grad для параметрів моделі;

– sample_data: функція для безкінечного отримання даних з джерела;

– Lab2RGB_out: конвертація Lab* в RGB для виводу;

– RGB2Lab, Normalize, numpy2tensor, tensor2numpy, preprocessing: допоміжні функції для обробки зображень.

Більшість інших бібліотек вже були описані в попередніх пунктах.

До вхідних зображень застосовуються певні трансформації, які включають конвертацію в тип float та перетворення в тензор PyTorch.

Лістинг 3.7 Конвертація:

```
transform_torch = albumentations.Compose(
[       albumentations.ToFloat(),
            albumentations.pytorch.ToTensorV2(),])
```

Для читання та підготовки зображень використовуються наступні кроки:

Крок 1. Зображення зчитуються за допомогою OpenCV.

Крок 2. Зображення може бути зменшено до розмірів MAX_SIZE для ефективності обчислень.

Крок 3. Кольорове зображення (color_src) змінюється розміром та конвертується за допомогою аугментацій.

Крок 4. Чорно-біле зображення (grayscale_dst) перетворюється в простір кольорів Lab та аугментується.

Лістинг 3.8 Перетворення зображення в простір кольорів:

```
MAX_SIZE = 1024
ref_image = cv2.imread(ref_path)
target_image = cv2.imread(target_path)
```

```
delta = max(target_image.shape)/MAX_SIZE
if delta > 1.0:
   target_image = cv2.resize(target_image, (0, 0), fx=1/delta, fy=1/delta)
color_src = ref_image
grayscale_dst = target_image
color_src =
cv2.resize(color_src,grayscale_dst.shape[:2],
interpolation=cv2.INTER_LINEAR)
color_src = transform_torch(image=color_src)['image']
grayscale_dst = cv2.cvtColor(grayscale_dst, cv2.COLOR_BGR2LAB)
grayscale_dst = transform_torch(image=grayscale_dst)['image']
grayscale_dst = grayscale_dst[0, ...].unsqueeze(0)
```

Для попереднього передбачення ab-каналів та створення кольорованого зображення:

Крок 1. Модель застосовується до чорно-білого зображення для передбачення ab-каналів в просторі кольорів Lab.

Крок 2. Зображення знову перетворюється, щоб отримати кольоровий вихід.

Крок 3. Результати зображення зберігаються у відповідній формі.

Лістинг 3.9 Попереднє передбачення ab-каналів:

```
with torch.no_grad():
pab       =
model(grayscale_dst.to('cuda').unsqueeze(0),
color_src.to('cuda').unsqueeze(0))
merged_lab   =
torch.cat((grayscale_dst.to('cuda').unsqueeze(0), pab), 1)
prgb = merged_lab.cpu().numpy()
prgb = np.transpose(prgb, (0, 2, 3, 1))
```

```
for i in range(prgb.shape[0]):
   prgb[i] =
cv2.cvtColor(np.clip(prgb[i]*255, 0, 255).astype(np.uint8),
cv2.COLOR_Lab2BGR)
   merged_lab[i]= transform_torch(image=prgb[i])['image'].to('cuda')/255.
```

Для кращого розуміння результатів вони відображаються разом з оригінальними зображеннями та джерелом кольорів.

Лістинг 3.10 Відображення результатів:

```
result_image =
np.transpose((merged_lab.cpu().numpy()[0]*255)
.astype(np.uint8), (1, 2, 0))
figure(figsize=(14, 12), dpi=80)
plt.subplot(1, 3, 1)
plt.axis('off')
plt.imshow(target_image[...,::-1])
plt.title('Target')
plt.subplot(1, 3, 2)
plt.axis('off')
plt.imshow(ref_image[...,::-1])
plt.title('Color source')
plt.subplot(1, 3, 3)
plt.axis('off')
plt.imshow(result_image[...,::-1]) = plt.title('Result')
```

Для навчання цього методу використовується всього одне кольорове зображення – приклад. Для наданих вище зображень приклади показані на рисунках 3.11 – 3.15.

Рисунок 3.11 – Приклад кольорового зображення для колоризації їжі

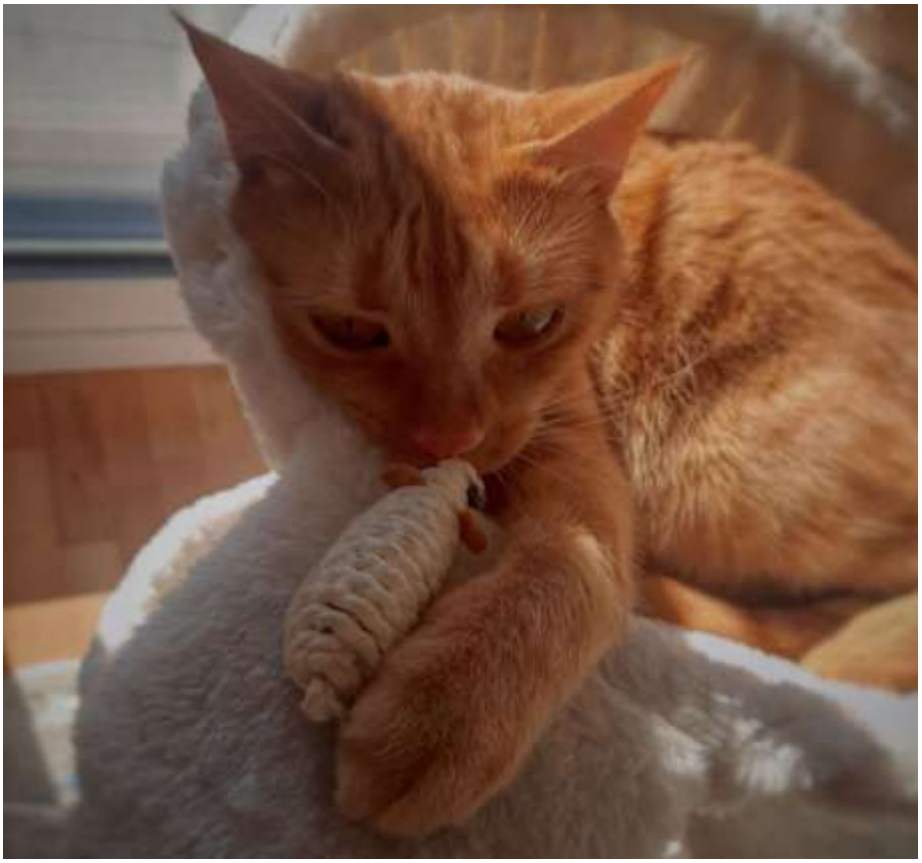

Рисунок 3.12 – Приклад кольорового зображення для колоризації тварини

Рисунок 3.13 – Приклад кольорового зображення для колоризації людини

Рисунок 3.14 – Приклад кольорового зображення для колоризації рослини

Рисунок 3.15 – Приклад кольорового зображення для колоризації міста

3.2.3 Метод DeOldify

Для реалізації цього методу використовуються наступні методи бібліотеки DeOldify.

Лістинг 3.11 Імпорт бібліотек та налаштування:

```
from deoldify import device
from deoldify.device_id import DeviceId
from deoldify.visualize import *
from deoldify.distributed import *
```

Далі обирається модель колоризації, можливо з використанням передвчених ваг (`artistic=True`).

Лістинг 3.12 Ініціалізація та завантаження моделі:

```
colorizer = get_image_colorizer(artistic=True)
```

Визначаємо URL чорно-білого зображення та використовуємо `plot_transformed_image` для генерації кольоризованого зображення.

Лістинг 3.13 Вхідні та вихідні дані:

```
source_url='https://www.example.com/path/to/black_and_white_image.jpg'
 colorizer.plot_transformed_image(path=source_url,
render_factor=10, display_render_factor=True,
figsize=x,y), results_dir=Path('./results'))
```

Лістинг 3.14 Відображення результатів:

```
show_image_in_notebook('./results/colorized.jpg', figsize=(8,8),
title='Colorized Image')
```

Також за допомогою цього метода можливо реалізувати колоризацію відео.

Лістинг 3.15 Приклад коду для колоризації відео

```
video_path = colorizer.colorize_from_file_name('video.mp4')
show_video_in_notebook(video_path)
```

Більшість операцій зводяться до використання методів та функцій, наданих DeOldify, що робить процес досить зручним та інтуїтивно зрозумілим.

3.3 Тестування та порівняння реалізованих методів

Після успішної реалізації обраних методів колоризації, наступним етапом є їх тестування та подальше порівняння за основними критеріями.

Перший критерій, який важливий для оцінки ефективності методів, – це швидкість обробки. Вимірювання часу виконання для кожного методу дозволяє зрозуміти, наскільки ефективно він працює в реальному часі та його можливу придатність для швидкісного використання, наприклад, в обробці великого обсягу зображень або в реальному часі при поточній генерації зображень.

Другий критерій порівняння – складність попередньої підготовки. Це включає в себе будь-які передпроцесингові етапи або вимоги до вхідних даних, які можуть вплинути на зручність та універсальність використання методу. Наприклад, методи, які вимагають масштабування чи попередньої обробки вхідних зображень, можуть мати обмеження в порівнянні з тими, що працюють безпосередньо з невеликими чорно-білими зображеннями.

Третій критерій – якість кінцевого результату. Оцінка точності та реалістичності колоризації є ключовою для визначення ефективності методу. Це включає в себе порівняння з оригінальними кольоровими зображеннями, а також визначення, наскільки добре метод враховує контекст та текстурні особливості для створення природного та гармонійного кольорового відтінку.

Отже, проведення тестування та порівняння за цими критеріями надасть об’єктивний погляд на ефективність та придатність кожного методу колоризації в конкретному контексті використання.

3.3.1 Метод Colorization using optimization

Для наданих зображень з попередньою обробкою для використання методу Colorization using optimization було отримано результати, показані на рисунках 3.16 – 3.20.

Рисунок 3.16 – Результат колоризації зображення їжі за допомогою Colorization using optimization

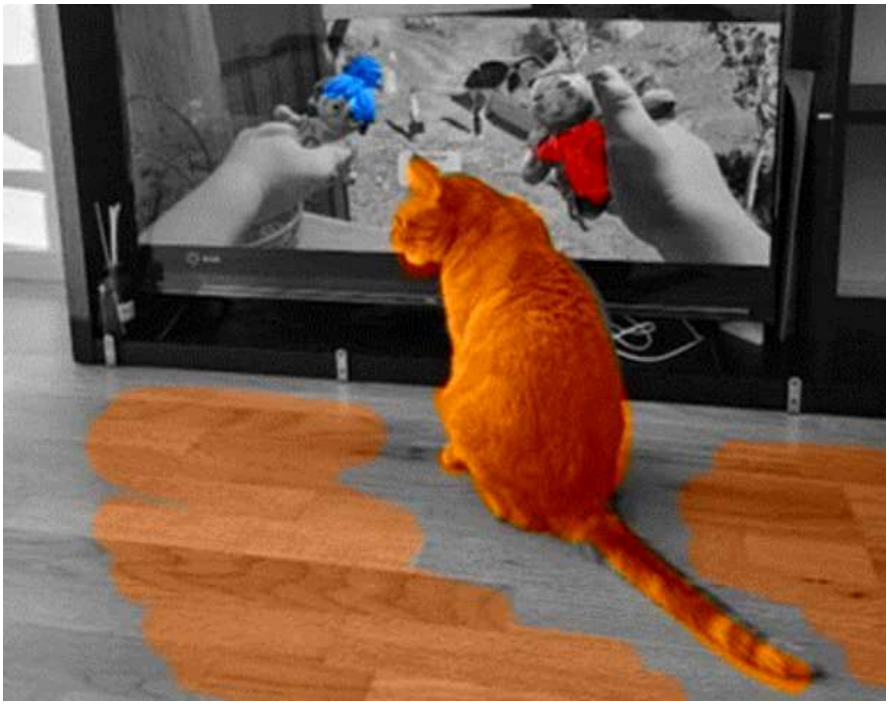

Рисунок 3.17 – Результат колоризації зображення тварини за допомогою Colorization using optimization

Рисунок 3.18 – Результат колоризації зображення людини за допомогою Colorization using optimization

Рисунок 3.19 – Результат колоризації зображення рослини за допомогою Colorization using optimization

Рисунок 3.20 – Результат колоризації зображення міста за допомогою Colorization using optimization

У використанні цього метода є безумовний плюс – можна власноруч налаштувати будь-які кольори. Проте є недоліки у затраті часу на надання кольору. Наприклад, для створення опорного зображення для колоризації міста треба опрацювати дуже багато дрібних деталей. Час обробки зображень програмою та оцінка результатів (від 1 до 5) наведені у таблиці 3.1.

Таблиця 3.1 – Витрати часу на обробку зображень методом Colorization using optimization

Назва зображення у програмній реалізації	Час, затрачений на обробку	Оцінка результатів
Cupckake_colored.bmp	23 секунди	4
Cat_colored.bmp	0,94 секунди	2
Human_colored.bmp	0,78 секунд	3
Plant_colored.bmp	0,64 секунд	1
City_colored.bmp	0,58 секунди	3

Як можна побачити з таблиці 3.1: на обробку першого зображення пішло доволі багато часу. Проте і саме зображення можна оцінити краще, ніж ті, на які затрати часу значно менші.

Час виконання оптимізаційних алгоритмів може сильно варіюватися в залежності від розміру зображення, складності алгоритму та продуктивності апаратного забезпечення. Є декілька можливих причин для того, чому ви отримали такий час обробки:

– розмір зображення: Великі зображення вимагають більше часу для обробки. Якщо ви працюєте з великими вихідними зображеннями, це може вплинути на час обчислення;

– ефективність алгоритму: Оптимізаційні алгоритми можуть вимагати значної кількості обчислень, особливо якщо вони використовують ітераційні методи або оптимізаційні задачі з великою кількістю змінних;

– реалізація коду: Якщо код не оптимізований або має низьку ефективність, це також може призвести до збільшення часу виконання. Ефективніше написаний код може значно покращити продуктивність.

3.3.2 Метод Color2Embed

Метод Color2Embed є дещо зручнішим у порівнянні з попереднім методом, особливо в аспекті попередньої підготовки зображень. Замість того, щоб власноруч додавати необхідні кольори для кожного чорно-білого зображення, модель Color2Embed може автоматично взяти інформацію з аналогічних зображень та використовувати її для колоризації.

Цей підхід робить використання методу Color2Embed більш гнучким та менш часо- та ресурсомістким. Замість того, щоб ретельно готувати вхідні дані або вручну вказувати кольори для кожного окремого випадку, модель може використовувати здобуті знання з попередніх зображень для автоматичного підбору кольорів для нових чорно-білих зображень.

Результати застосування методу Color2Embed можна побачити на рисунках 3.21 – 3.25.

Рисунок 3.21 – Результат колоризації зображення їжі за допомогою Color2Embed

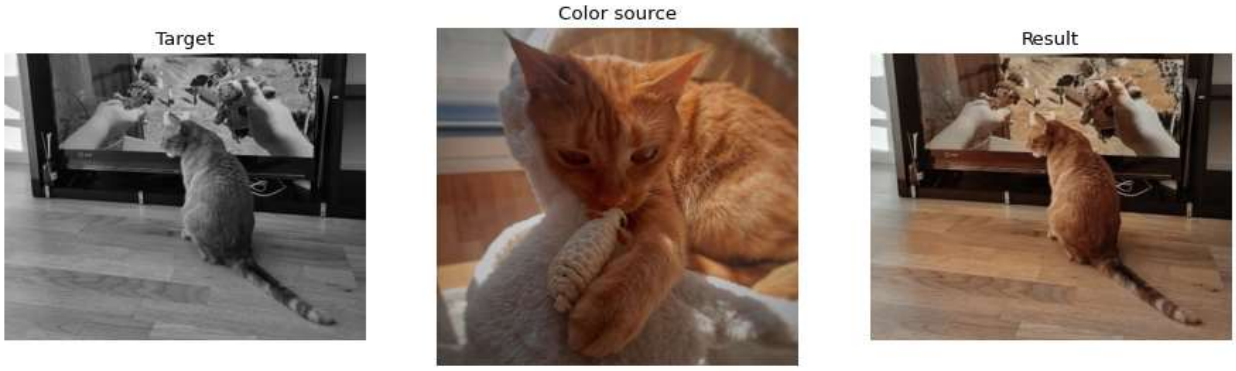

Рисунок 3.22 – Результат колоризації зображення тварини за допомогою Color2Embed

Рисунок 3.23 – Результат колоризації зображення людини за допомогою Color2Embed

Рисунок 3.24 – Результат колоризації зображення рослини за допомогою Color2Embed

Рисунок 3.25 – Результат колоризації зображення міста за допомогою Color2Embed

Час обробки зображень програмою та оцінка результатів (від 1 до 5) наведені у таблиці 3.2.

Таблиця 3.2 – Витрати часу на обробку зображень методом Color2Embed

Назва зображення у програмній реалізації	Час, затрачений на обробку	Оцінка результатів
Cupckake_colored.bmp	22 секунди	2
Cat_colored.bmp	17 секунд	4
Human_colored.bmp	25 секунд	3

Продовження таблиці 3.2

Plant_colored.bmp	19 секунд	2
City_colored.bmp	22 секунди	1

Як можна побачити з відображених результатів, час на колоризацію зображень є більш рівномірним, ніж при використанні Colorization using optimization. Проте результати все ще не можна назвати ідеальними. Це може бути пов’язано з тим, що метод Color2Embed потребує деякого доопрацювання.

3.3.3 Метод DeOldify

Для використання метода DeOldify не потрібно ніяких додаткових зображень. Вся робота базується на архітектурі GAN [7] для колоризації зображення. Особливості реалізації методу були описані вище, проте для тестування використовуватиметься вебсайт з відкритим доступом [30].

Результати обробки зображень продемонстровано на рисунках 3.26 – 3.30.

Рисунок 3.26 – Результат колоризації зображення їжі за допомогою DeOldify

Рисунок 3.27 – Результат колоризації зображення тварини за допомогою DeOldify

Рисунок 3.28 – Результат колоризації зображення людини за допомогою DeOldify

Рисунок 3.29 – Результат колоризації зображення рослини за допомогою DeOldify

Рисунок 3.30 – Результат колоризації зображення міста за допомогою DeOldify

Час обробки зображень програмою та оцінка результатів (від 1 до 5) наведені у таблиці 3.3.

Таблиця 3.3 – Витрати часу на обробку зображень методом DeOldify

Назва зображення у програмній реалізації	Час, затрачений на обробку	Оцінка результатів
Cupckake_colored.bmp	4 секунди	3
Cat_colored.bmp	1 секунда	4
Human_colored.bmp	3 секунди	2
Plant_colored.bmp	2 секунди	1
City_colored.bmp	4 секунди	4

Як можна побачити, загальні витрати часу на обробку чотирьох зображень менші, ніж в перших двох методах, проте якість результатів все ще не можна назвати задовільною. На оброблених зображення переважно обмежена кількість кольорів з невисокою яскравістю.

Більше результатів обробки зображень після додаткового опрацювання показані у додатку А.

ВИСНОВКИ

У рамках кваліфікаційної роботи було досліджено методи колоризації напівтонових зображень.

У процесі виконання роботи було реалізовано для проведення дослідження наступні методи: Colorization using optimization, Color2Embed та DeOldify, як три методи з різними підходами до вирішення завдання колоризації.

Однією з ключових задач роботи було вивчення ефективності традиційних методів колоризації, зокрема на основі оптимізаційних підходів. Аналізуючи алгоритм Colorization using optimization, ми визначили його переваги та недоліки. Основною перевагою є можливість відновлення кольору зображень без потреби у великих обчислювальних ресурсах. Однак, ці методи можуть бути обмежені у вирішенні складних сценаріїв та вимагають деякої додаткових знань у виборі параметрів оптимізації.

Також в роботі були розглянуті сучасні методи, зокрема ті, що використовують глибоке навчання. Зазначено, що ці методи, хоча і вимагають значних обчислювальних ресурсів для тренування, здатні досягати вражаючих результатів у відтворенні кольору та зменшенні часу обробки.

Для кращого відображення процесу колоризації було підібрано декілька зображень з різним контекстом.

Результати проведених досліджень підтверджують, що традиційні методи оптимізації, використовуючи контекст та геометричні особливості, можуть вирішувати задачу колоризації, але при цьому вони можуть бути обмежені обчислювальними ресурсами та часом виконання.

Глибокі нейронні мережі, особливо архітектури, засновані на генеративних засобах, як наприклад GAN, представляють сучасний та досить ефективний підхід до задачі колоризації. Моделі, такі як DeOldify, виявилися досить успішними в автоматичній колоризації зображень, зокрема для відтворення деяких кольорових деталей.

З протестованих обраних методів можна побачити, що різні підходи можна використовувати для колоризації зображень з різним контекстом. Наприклад: для колоризації зображень їжі або яскравих елементів, які не мають великої кількості дрібних деталей може підійти метод Colorization using optimization. Це може бути дещо ресурсозатратно, проте надає можливість отримати максимально задовільні результати.

З колоризацією зображень із тваринами та людьми найкраще впорався метод Color2Embed. Його зручно використовувати, якщо є зображення – приклад, в якому наявні схожі з монохромним деталі.

За допомогою DeOldify було добре виконано колоризацію зображення міста, хоча у результуючому зображенні явно присутня певна тьмяність кольорів. Також зображення тварини та їжі можна назвати задовільними.

Проте на всіх продемонстрованих результатах чітко видно необхідність доопрацювання. Це відноситься як до попередньої обробки зображень, так і до покращення роботи самих методів. Крім того, додаткових досліджень потребує питання колоризації великої кількості зображень.

В умовах активного розвитку технологій у комп’ютерного зору та глибокого навчання існує багато перспективних напрямків для подальших досліджень та покращень методів колоризації. Це може бути збільшення швидкості обробки зображень, або оптимізація витрат часу, вдосконалення глибоких нейронних мереж для колоризації з використанням більш складних архітектур, адаптованих до різних видів зображень та сценаріїв тощо.

Подальші дослідження в цій області сприятимуть розвитку нових інноваційних методів, а також розширять область застосування колоризації для відновлення та покращення зображень у різних галузях, включаючи медицину та мистецтво.

Результати дослідження апробовано у вигляді тез доповідей під час VIII Міжнародної науково-практичної конференції «Distance learning in universities and modern problems» [46].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. In ACM SIGGRAPH 2004 Papers (pp. 689-694).

2. Zhao, H., Wu, W., Liu, Y., & He, D. (2021). Color2Embed: Fast Exemplar-Based Image Colorization using Color Embeddings. arXiv preprint arXiv:2106.08017.

3. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

4. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2020). Advances in Spatio-Temporal Segmentation of Visual Data. Springer.

5. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative based clustering of long multivariate sequences with different lengths. In 2018 IEEE second international conference on Data Stream Mining & Processing (DSMP) (pp. 545-548). IEEE.

6. Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. ISPRS journal of photogrammetry and remote sensing, 173, 24-49.

7. Shahin, A. I., Aly, W., & Aly, S. (2023). MBTFCN: A novel modular fully convolutional network for MRI brain tumor multi-classification. Expert Systems with Applications, 212, 118776.

8. Sun, Y., Cheng, H., Zhang, S., Mohan, M. K., Ye, G., & De Schutter, G. (2023). Prediction & optimization of alkali-activated concrete based on the random forest machine learning algorithm. Construction and Building Materials, 385, 131519.

9. Wang, H., Li, G., & Wang, Z. (2023). Fast SVM classifier for large-scale classification problems. Information Sciences, 642, 119136.

10. Sigut, J., Castro, M., Arnay, R., & Sigut, M. (2020). OpenCV basics: A mobile application to support the teaching of computer vision concepts. IEEE Transactions on Education, 63(4), 328-335.

11. Haghighat, M.; Zonouz, S.; Abdel-Mottaleb, M. (2013). "Identification Using Encrypted Biometrics". Computer Analysis of Images and Patterns. Lecture Notes in Computer Science. Vol. 8048. pp. 440–448. doi:10.1007/978-3-642-40246-3_55. ISBN 978-3-642-40245-6.

12. Through The Eyes of Gabor Filter. URL: https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97 (дата звернення 14.10.2023).

13. Karanwal, S., & Diwakar, M. (2021). OD-LBP: Orthogonal difference-local binary pattern for Face Recognition. Digital Signal Processing, 110, 102948.

14. Serdyuk, M. E., Syryk, S. F., & Sokol, O. O. (2019). Automatic colorization of digital halftone images using neural networks. Питання прикладної математики і математичного моделювання.

15. Chen, T., Cheng, Y., Gan, Z., Liu, J., & Wang, Z. (2021). Data-efficient gan training beyond (just) augmentations: A lottery ticket perspective. Advances in Neural Information Processing Systems, 34, 20941-20955.

16. Sluzek, A. (2023). On unguided automatic colorization of monochrome images.

17. Yu, Y., Wang, C., Fu, Q., Kou, R., Huang, F., Yang, B., ... & Gao, M. (2023). Techniques and challenges of image segmentation: A review. Electronics, 12(5), 1199.

18. Rajesh, M., Senapati, B., Das, R., & Martha, S. (2023). Identifying Colorectal Tumor For Single Cell RNA Sequence Using Rectified Linear Unit With Stochastic Gradient Descent. Procedia Computer Science, 218, 189-198.

19. Kadhim, D. J., Saleh, M. H., & Abou-Loukh, S. J. (2023). Evaluation of massive multiple-input multiple-output communication performance under a proposed improved minimum mean squared error precoding. IAES International Journal of Artificial Intelligence, 12(2), 984.

20. Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. Neural Computing and Applications, 1-18.

21. Xiao, Q., Shen, H., Yin, W., & Chen, T. (2023, April). Alternating projected sgd for equality-constrained bilevel optimization. In International Conference on Artificial Intelligence and Statistics (pp. 987-1023). PMLR.

22. Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14 (pp. 649-666). Springer International Publishing.

23. Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4401-4410).

24. Gao, R., Hou, X., Qin, J., Chen, J., Liu, L., Zhu, F., ... & Shao, L. (2020). Zero-VAE-GAN: Generating unseen features for generalized and transductive zero-shot learning. IEEE Transactions on Image Processing, 29, 3665-3680.

25. VAE-GAN. URL: https://www.linkedin.com/pulse/vae-gan-christiano-faig/?originalSubdomain=pt (дата звернення 28.10.2023).

26. Jin, Z., Xie, X., Geng, M., Wang, T., Hu, S., Deng, J., ... & Liu, X. (2023, June). Adversarial data augmentation using vae-gan for disordered speech recognition. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

27. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE *Access*, 11, pp. 126938-126949.

28. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

29. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

30. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

31. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, International Journal of Academic Engineering Research, 7(9), pp. 64-72.

32. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

33. Learn how to Colorize images with Neural Filters. URL: https://helpx.adobe.com/photoshop/using/colorize.html (дата звернення 02.11.2023).

34. Salmona, A., Bouza, L., & Delon, J. (2022). Deoldify: A review and implementation of an automatic colorization method. Image Processing On Line, 12, 347-368.

35. Vei, R. (2020). NoGAN: Deblurring Images without Adversarial Training (Doctoral dissertation, Ukrainian Catholic University).

36. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer International Publishing.

37. Image Colorization. URL: https://deepai.org/machine-learning-model/colorizer (дата звернення 07.11.2023).

38. Zhao, H., Wu, W., Liu, Y., & He, D. (2021). Color2Embed: Fast Exemplar-Based Image Colorization using Color Embeddings. arXiv preprint arXiv:2106.08017.

39. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

40. He, M., Chen, D., Liao, J., Sander, P. V., & Yuan, L. (2018). Deep exemplar-based colorization. ACM Transactions on Graphics (TOG), 37(4), 1-16.

41. Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.

42. Bahng, H., Yoo, S., Cho, W., Park, D. K., Wu, Z., Ma, X., & Choo, J. (2018). Coloring with words: Guiding image colorization through text-based palette generation. In Proceedings of the european conference on computer vision (eccv) (pp. 431-447).

43. Ghosh, S., Roy, P., Bhattacharya, S., Pal, U., & Blumenstein, M. (2022). Tic: Text-guided image colorization. arXiv preprint arXiv:2208.02843.

44. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

45. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 5907-5915).

46. Кирилова, В. А. (2023, November). ДОСЛІДЖЕННЯ МЕТОДІВ КОЛОРИЗАЦІЇ НАПІВТОНОВИХ ЗОБРАЖЕНЬ. In The 8th International scientific and practical conference “Distance learning in universities and modern problems”(November 07-10, 2023) Budapest, Hungary. International Science Group. 2023. 314 p. (p. 281).