НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег Чертов
«___» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Математичне та програмне забезпечення системи покращення якості фотографії»

Виконав:
студент IV курсу, групи КМ-92
Борисенко Богдан Русланович _____

Керівник:
Доцент, канд. техн. наук, доцент
Олефір Олександр Степанович _____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Доцент каф. ПЗКС, канд. техн. наук, доцент
Юрчишин Василь Якович _____

Засвідчую, що в цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.
Студент Борисенко Б. Р. _____

Київ — 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олег Чертов
«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Борисенку Богдану Руслановичу

1. Тема роботи: «Математичне та програмне забезпечення системи покращення якості фотографії», керівник роботи Олефір Олександр Степанови, канд. техн. наук, доцент, затверджені наказом по університету від «31» травня 2023 р. № 2108-С.
2. Термін подання студентом роботи: «12» червня 2023 р.
3. Вихідні дані до роботи: розроблюване забезпечування має вміти покращувати фотографії, що були зроблені при поганому освітлені.
4. Зміст роботи: проведення огляду існуючих рішень, вибирання тих, які найбільше підходять, розробка та реалізація моделі для покрашення якості фотографії. Проведення процесів верифікації та валідації моделі покрашення якості фотографії.
5. Перелік ілюстративного матеріалу: діаграма моделі покращення якості, діаграма діяльності, приклади архітектури, приклад роботи згортки, графіки функцій втрат, зображення тестових даніх.
6. Дата видачі завдання: «06» лютого 2023р.

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вивчення та збір літератури за тематикою дипломної роботи та збір даних	12.11.2022	
2	Підготовка матеріалів першого розділу	14.12.2022	
3	Проведення порівняльного аналізу математичних методів покрашення якості фотографії.	24.12.2022	
4	Підготовка матеріалів другого розділу роботи	01.02.2023	
5	Розроблення математичного забезпечення для покрашення якості фотографії.	01.03.2023	
6	Підготовка матеріалів третього розділу роботи	15.03.2023	
7	Розроблення програмного забезпечення для покрашення якості фотографії.	05.04.2023	
8	Підготовка матеріалів четвертого розділу роботи	15.04.2023	
9	Підготовка графічної частини	03.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	

Студент _____ Богдан БОРИСЕНКО

Керівник роботи _____ Олександр ОЛЕФІР

АНОТАЦІЯ

Дипломну роботу виконано на 50 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 8 найменувань. У роботі наведено 11 рисунків.

Метою роботи є дослідження та розробка математичного та програмного забезпечення покращення якості зображення, що базується на основі моделей машинного навчання, порівняння різних моделей машинного навчання, що існують для розв'язання даної задачі.

У роботі проведено аналіз існуючих методів, та на основі вибраних було розроблено забезпечення для покращення якості зображення, що зроблені в умовах поганої освітленості.

Ключові слова: покращення якості фотографії, машинне навчання, алгоритмом глибокого навчання, покращення фото.

ABSTRACT

The thesis consists of 50 pages, 2 appendices and a list of references of 8 titles. The work contains 11 figures.

The purpose of the work is to research and develop mathematical and software for image quality improvement based on machine learning models, to compare different machine learning models that exist to solve this problem.

The paper analyses the existing methods, and based on the selected ones, develops software to improve the quality of images taken in low light conditions.

Keywords: photo quality improvement, machine learning, deep learning algorithm, photo enhancement.

Зміст

Перелік умовних позначень, скорочень і термінів

GPU – Graphics processing unit

ILSVRC – ImageNet Large Scale Visual Recognition Challenge

ReLU – Rectified Linear Unit

SSIM – structural similarity index measure

UML – Unified Modeling Language

Zero-DCE net – Zero-Reference Deep Curve Estimation Network

Датасет – набір даних

Вступ

Обробка фотографії сьогодні використовується у різноманітних сферах життя і має широкий спектр застосувань. Кожного дня за допомогою фотографій люди зберігають якісь моменти з особистого життя, рекламують свої продукти та документують результати наукових досліджень, але майже завжди при створенні фотографій вночі є велика проблема у вигляді втрати якості зображення за рахунок недостатньої кількості освітлення.

Раніше, при використанні фотоплівки, погана якість зображення була майже патовою ситуація, але за рахунок науково-технічного прогресу, розвитку в напрямку комп'ютерних наук, нових відкриттів у сфері машинного навчання та еволюції обчислювальної техніки за останні роки, перед людством відкрились нові можливості. З'явилась можливість покращувати фотографії електронного формату за допомогою різних програмних засобів, з'явились моделі машинного навчання які вміють робити різноманітні операції над ілюстраціями.

Створення систем покращення якості фотографії дало можливість подивитися під новим кутом на фото які раніше вважалися невдало зробленими. За допомогою моделей машинного навчання науковці змогли отримувати з фотографій більше даних, аналізувати їх краще, та використовувати з максимальною вигодою. Поліція отримала можливість мати фото та відео гарної якості при будь-яких умовах. Звичайні люди змогли покращити якість свої світлин, до такого результату, який закарбувався у них в спогадах. Тобто засоби покращення якості фотографії змінили наше життя, у ліпшу сторону.

Системи, що вміють ефективно обробляти фотографії, в основному являють собою модель машинного навчання, котрі були вже навчені до цього та надають вже готовий функціонал для подальшої роботи. Саме цей підхід і буде використовуватися.

У даній дипломній роботі розглядається задача використання методів машинного навчання для покращення якості фотографій. Увага приділяється фотографіям, які мають погану якість через недостатню кількість освітлення. Для розв'язання цієї задачі необхідно розробити та реалізувати систему покращення якості фотографій, яка здатна автоматично коригувати недоліки фотографій.

1 ПОСТАНОВКА ЗАДАЧІ

Метою роботи є дослідження та розробка математичного та програмного забезпечення покращення якості зображення, що базується на основі моделей машинного навчання, порівняння різних моделей машинного навчання, що існують для розв'язання даної задачі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести огляд наявних засобів покращення зображень.
2. Обґрунтувати та вибрати методи машинного навчання.
3. Розробити систему покращення якості зображень.
4. Верифікувати та валідувати систему.

Реалізована система має задовольняти такі вимоги:

1. мати високі якісні показники покращення якості фотографії;
2. має включати в себе функціонал для прибирання шуму з фотографії;
3. швидко та якісно навчатися на різних вибірках.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПОКРАЩЕННЯ ЯКОСТІ ЗОБРАЖЕННЯ

2.1 Огляд наявних рішень

Задача покращення фотографії вже існує дуже давно, так як майже з самого початку масового використання фотоапаратів з’явилася потреба у різноманітних корекціях фотографії. У випадку цієї роботи введеться пошук вирішення проблеми покращення фотографій, що були зроблені при поганому освітлені, або в нічний час.

Одним із наявних рішень в наш час є застосування функціоналу кривих у програмі Adobe Photoshop, за допомогою яких змінюються різні параметри, такі як кольоровість, контраст та інші. Цей підхід потребує не аби яких навичок у знанні функціоналу програми та вмінні оперувати ним. Результат цього методу завжди є гарним, але потребує від користувача довго вивчення роботи з програмою та забирає багато часу при самій обробці.

Іншим підходом є використання моделі машинного навчання Zero-DCE [1]. Саме таке рішення дозволяє покращити якість фотографії, а потім прибрати зернистість з фото, які були на фото, або утворились під час роботи Zero-DCE моделі.

Задачу можна розбити на 3 основних пункти:

1) попередня обробка до вигляду зрозумілої для моделі машинного навчання та нейронної мережі;

2) обробка фотографії за допомогою моделі машинного навчання Zero-DCE;

3) фінальна обробка та збереження кінцевого фото.

Схожим підходом до попереднього є використання суто математичних підходів, які можна об’єднувати між собою при необхідності. Для вирішення цієї задачі використовують такі методи як:

1) глобальна і локальна адаптивна обробка гамми;

2) фільтрація згладжування;

3) застосування розширення контрастності;

4) алгоритми обробки гістограми;

2.2 Математичні методи

2.2.1 Глобальна і локальна адаптивна обробка гамми

Гамма-корекція є важливим процесом в обробці зображень і пов'язана з відтворенням кольорів на дисплеях або пристроях відображення. Гамма-корекція використовується для компенсації нелінійності відтворення яскравості на пристроях, що мають позитивну степінь яскравості (більші значення яскравості представлені більшими числами). Гамма-корекція може бути реалізована через глобальну адаптивну обробку гамми і локальну адаптивну обробку гамми.

Глобальна адаптивна обробка гамми використовує одну значення гамми для всього зображення. Формула глобальної адаптивної обробки гамми має вигляд:

$$V_{out} = V_{in}^{\gamma} \tag{2.1}$$

де V_{in} - вхідне значення пікселя;

V_{out}- вихідне значення пікселя;

γ - значення гамми. Значення γ зазвичай знаходиться в діапазоні від 0,1 до 5,0, де 1,0 представляє лінійну гамму, а значення менше 1,0 змінюють гамму на меншу ступінь (темніші значення стають ще темнішими, яскраві значення - менш яскравими), а значення більше 1,0 змінюють гамму на більшу ступінь (темні значення стають світлішими, яскраві значення - більш яскравими).

Локальна адаптивна обробка гамми використовує різні значення гамми для різних частин зображення, що дозволяє більш точно адаптуватися до локальних

особливостей зображення. Наприклад, на частині зображення з високою яскравістю може бути застосована більша гамма для збільшення контрастності, тоді як на частині зображення з низькою яскравістю може бути застосована менша гамма для збільшення деталей у темних областях. Локальна адаптивна обробка гамми зазвичай базується на використанні масок або функцій яскравості для визначення рівнів гамми для кожної локальної області.

Формула для локальної адаптивної обробки гамми може виглядати наступним чином:

$$V_{out}(x, y) = V_{in}^{\gamma(x,y)}(x, y) \quad (2.2)$$

де $V_{in}(x, y)$ - вхідне значення пікселя у позиції (x, y);

$V_{out}(x, y)$ - вихідне значення пікселя у позиції (x, y);

$\gamma(x, y)$ - значення гамми, яке залежить від позиції пікселя.

Значення $\gamma(x, y)$ можуть бути обчислені за допомогою масок або функцій яскравості, які відображають локальні властивості зображення.

Глобальна і локальна адаптивна обробка гамми дозволяють відтворити правильні кольори та контраст на дисплеях або в пристроях відображення. Глобальна адаптивна обробка гамми підходить для загальних зображень, де одне значення гамми застосовується до всього зображення. Локальна адаптивна обробка гамми підходить для зображень зі змінною яскравістю або контрастом, де різні значення гамми застосовуються до різних частин зображення.

Ці математичні підходи дозволяють керувати яскравістю, контрастом та деталями на зображенні, забезпечуючи кращу візуальну якість та точніше відтворення кольорів.

2.2.2 Фільтрація згладжування

Фільтрація згладжування є методом обробки зображень, який допомагає зменшити шум і видалити артефакти, спричинені недостатнім освітленням або іншими факторами. Цей метод базується на математичних операціях розмиття і згладжування значень пікселів.

Одним з поширених методів фільтрації згладжування є фільтр Гауса [2]. Він використовує математичну функцію Гауса, яка представляє собою криву згладжування, для розмиття значень пікселів. Формула для фільтра Гауса має вигляд:

$$V_{out(x,y)} = \left(\frac{1}{2\pi\sigma^2}\right) * \sum[i = -k \text{ to } k]\sum[j = -k \text{ to } k]V_{in(x+i,y+j)} * \exp\left(-\left(\frac{i^2+j^2}{2\sigma^2}\right)\right) \quad (2.3)$$

де $V_{in(x,y)}$ – вхідне значення пікселя у позиції (x, y);

$V_{out(x,y)}$ – вихідне значення пікселя у позиції (x, y);

σ – стандартне відхилення функції Гауса, контролюючи ступінь розмиття. Більше значення σ призводить до більшого розмиття;

$\sum[i = -k \text{ to } k]i \sum[j = -k \text{ to } k]$ – суми значень пікселів в околі пікселя;

k – радіус фільтра, тобто кількість сусідніх пікселів, що враховуються при обчисленні результуючого значення пікселя.

Інший поширений метод фільтрації згладжування - це медіанний фільтр. Він замінює значення пікселя медіаною зі значень пікселів в його околі. Формула для медіанного фільтра виглядає наступним чином:

$$V_{out(x,y)} = \text{median}\left(V_{in(x+i,y+j)}\right), \quad (2.4)$$

де $V_{in(x+i,y+j)}$ – значення пікселів в околі пікселя (x, y);

median – виражає медіану цих значень;

$V_{in(x,y)}$ – вхідне значення пікселя у позиції (x, y);

$V_{out(x,y)}$ – вихідне значення пікселя у позиції (x, y).

Кількість пікселів в околі може бути визначена радіусом фільтра.

2.2.3 Алгоритми обробки гістограми

Алгоритми обробки гістограми використовуються для аналізу та модифікації гістограми зображення, що представляє розподіл яскравості пікселів. Ці алгоритми дозволяють покращити контраст, розширити динамічний діапазон, виправити недоекспоновані або переекспоновані області та змінити загальну яскравість зображення. Декілька поширених алгоритмів обробки гістограми включають:

а) глобальне розтягнення гістограми.

Цей алгоритм розтягує діапазон значень яскравості на всьому зображенні, щоб забезпечити оптимальне використання всього діапазону яскравості. Формула для глобального розтягнення гістограми може бути виражена таким чином:

$$V_{out(x,y)} = \left(V_{in(x,y)} - V_{min}\right) * \frac{\left(V_{new_{max}} - V_{new_{min}}\right)}{\left(V_{max} - V_{min}\right)} + V_{new_{min}} \quad (2.5)$$

$V_{in(x,y)}$ – вхідне значення пікселя у позиції (x, y);

$V_{out(x,y)}$ – вихідне значення пікселя у позиції (x, y) після розтягнення гістограми;

V_{min} – мінімальне значення яскравості на зображенні;

V_{max} – максимальне значення яскравості на зображенні;

$V_{new_{min}}$ – нове мінімальне значення яскравості;

$V_{new_{max}}$ – нове максимальне значення яскравості.

б) локальне розтягнення гістограми:

$$V_{out(x,y)} = \left(V_{in(x,y)} - V_{local_{min}}\right) * \frac{\left(V_{new_{max}} - V_{new_{min}}\right)}{\left(V_{local_{max}} - V_{local_{min}}\right)} + V_{new_{min}} \quad (2.6)$$

де $V_{local_{min}}$ – мінімальне значення яскравості у локальному околі пікселя (x, y);

$V_{local_{max}}$ – максимальне значення яскравості у локальному околі пікселя (x, y);

$V_{in(x,y)}$ – вхідне значення пікселя у позиції (x, y);

$V_{out(x,y)}$ – вихідне значення пікселя у позиції (x,y) після розтягнення гістограми;

$V_{new_{min}}$ – нове мінімальне значення яскравості;

$V_{new_{max}}$ – нове максимальне значення яскравості.

2.3 Методи машинного навчання

Методи машинного навчання - це комп'ютерні алгоритми і статистичні моделі, які дають комп'ютеру здатність "навчатися" і вдосконалювати свою продуктивність на основі вхідних даних без явного програмування. Ці методи дозволяють комп'ютеру розпізнавати закономірності, витягувати корисну інформацію і приймати рішення на основі навчальних прикладів.

Для покращення фотографії, що зроблена при поганому освітленні, можна використовувати Zero-DCE net модель.

Zero-DCE net є моделлю машинного навчання, призначеною для покращення контрасту та якості зображень без використання посилань на початкове зображення. Zero-DCE net отримує на вхід зображення з низьким рівнем освітленості, а на виході

створює тональні криві високого порядку. Ці криві потім використовуються для по піксельного регулювання динамічного діапазону вхідного зображення для отримання покращеного зображення. Процес оцінки кривих відбувається таким чином, щоб підтримувати діапазон покращеного зображення і зберігати контрастність сусідніх пікселів. Ця оцінка кривих натхненна коригуванням кривих, що використовується в програмному забезпеченні для редагування фотографій, такому як Adobe Photoshop, де користувачі можуть коригувати точки по всьому тональному діапазону зображення.

Zero-DCE net привабливий тим, що він не вимагає жодних припущень щодо еталонних зображень: йому не потрібні пари вхідних/вихідних зображень під час навчання. Це досягається за допомогою набору ретельно сформульованих функцій похибок, які неявно вимірюють якість покращення і керують навчанням мережі.

Zero-DCE net використовує дві підмережі: енкодер та декодер. Енкодер відповідає за вилучення високорівневих ознак зображення, а декодер відповідає за відтворення виправленого зображення з вилучених ознак.

Нижче наведені формули для Zero-DCE net моделі:

Формула для розрахунку вихідного зображення:

$$O = D\big(E(I)\big), \tag{2.7}$$

де I - вхідне зображення;

E - енкодер (перетворення зображення в високорівневі ознаки);

D - декодер (відтворення виправленого зображення з ознак);

Функціонал втрат (loss function) для навчання моделі:

$$L = L1_{loss(O,I)} + w * SSIM_{loss(O,I)} \tag{2.8}$$

де $L1_{loss}$ – функція втрат L1 (наприклад, середньоквадратичне відхилення);
$SSIM_{loss}$ – функція втрат SSIM, яка враховує якість структури зображення;
w - ваговий коефіцієнт для балансування двох компонентів функціоналу втрат;
Формула для розрахунку локальної контрастної мапи (Local Contrast Map):

$$C = F(O) \tag{2.9}$$

де O – виправлене зображення;
F – функція, яка обчислює локальну контрастну карту на основі виправленого зображення.
Функціонал втрат для навчання моделі з урахуванням локальної контрастної мапи:

$$L = L1_{loss(O,I)} + w1 * SSIM_{loss(O,I)} + w2 * L1_{loss(C,C_{gt})}$$

де C_{gt} – референсна локальна контрастна карта (якщо доступна);
$L1_{loss}, SSIM_{loss}$ – функції втрат для порівняння виправленого зображення з вхідним зображенням та локальною контрастною мапою з референсною мапою:
$w1, w2$ - вагові коефіцієнти для балансування компонентів функціоналу втрат.

2.4 Висновки до розділу

Задача по покращенню фотографій широко розповсюджена, її можна вирішити багатьма способами& Можна використовувати як суто математичні підходи так і алгоритми машинного навчання, також можна скористатися різними комерційними програмами, наприклад Adobe Photoshop, для вирішення цієї задачі. У процесі

написання даного розділу проведений огляд наявних методів розв’язання поставленої задачі покращення фотографій, що були зроблені при поганому освітлені, або в нічний час. Було оглянуто математичну сторону деяких алгоритмів та математичну сторону функцій втрат.

3 МОДЕЛЬ СИСТЕМИ ПОКРАЩЕННЯ ЯКОСТІ ФОТОГРАФІЇ

3.1 Компонентна модель системи покращення якості фотографії

Використовуючи UML мову та техніки бізнес-моделювання Еріксона-Пенкера представимо у вигляді діаграми компонентів структуру системи покращення якості фотографії та представимо інформацію про компоненти та їх імплементацію [4]:

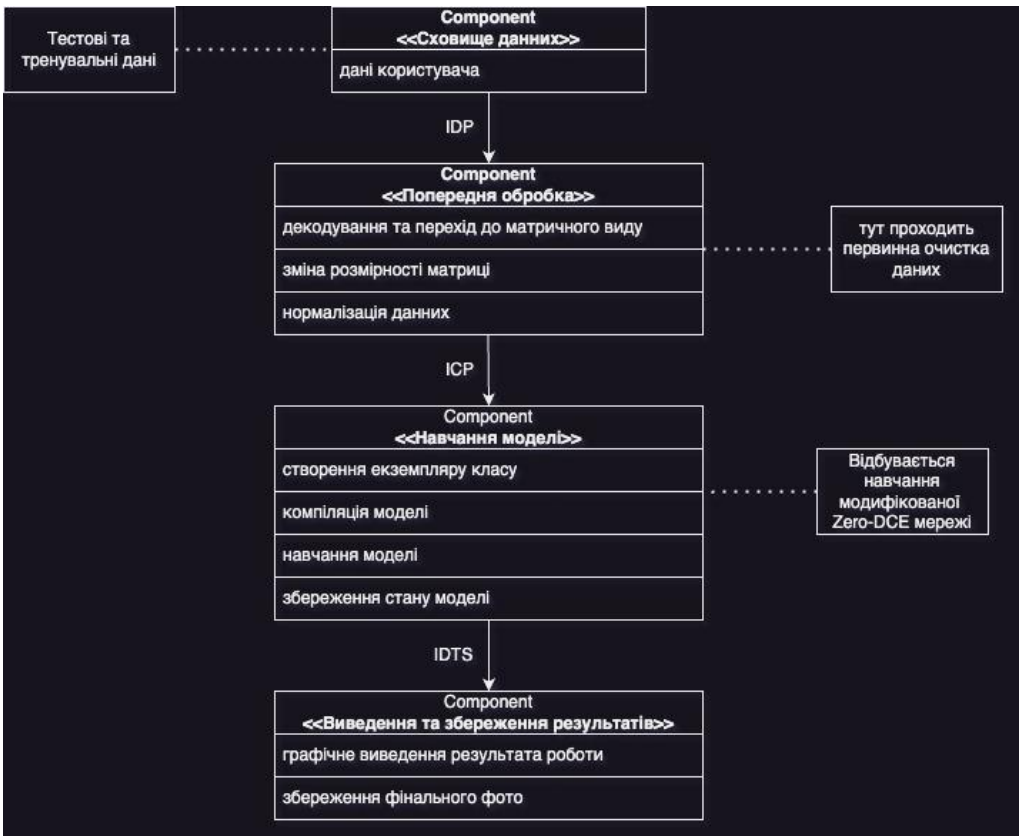

Рисунок 2.1 – Модель системи покращення якості фотографії. Діаграма компонентів у нотації UML

Компонент «Сховище даних» – це сховище, на якому зберігаються набори даних.

Компонент «Попередня обробка» призначений для зміни початкових даних, якщо вони знаходяться не у тому вигляді, що потрібно системі. У ньому виконуються наступні кроки:

а) перехід до матричного вигляду;

б) зміна розширення для дуже великих зображень;

в) нормалізація пікселів;

Компонент «Навчання моделі» призначений для обробки світлин після попередньої обробки, в ньому відбувається основна логіка системи. В цій компоненті проходить навчання моделі.

Компонент «Виведення та збереження результатів» відображає результат навчання мережі – покращену фотографію, графічне виведення з можливістю збереження.

Інтерфейси [3]:

а) інтерфейс IDP (Interface Data Processing) – інтерфейс передання завантажених в оперативну пам’ять сирих наборів даних на вхід процесу попередньої обробки;

б) інтерфейс ICP (Interface Clustering Process) – інтерфейс передавання вихідних (із компонента попередньої обробки) опрацьованих даних на вхід процесу навчання нейронної мережі;

в) інтерфейс IDTS (Interface Development Team Selection) – інтерфейс передавання результатів навчання мережі на вхід процесу виведення результатів для відображення проробленої роботи, тобто відображення покращеної світлини;

Імплементація динамічного представлення системи. На основі компонентної моделі та математичних методів реалізації компонентів розроблений алгоритм роботи системи. Діаграма діяльності (рис. 2.2) – це візуалізація основних циклів функціонування процесів і їхньої взаємодії першого рівня.

Імплементація компонентів Сховища даних, завантаження даних та попередньої обробки даних. Для подальшої покращення світлин необхідно обробити набори даних. Алгоритм збирання та обробки даних:

а) набори даних з фотографіями:

– збирання/добирання сирих даних з будь-яких відкритих ресурсів;

– збереження даних у відповідну теку;

б) попередня обробка даних:

– перехід до матричного вигляду;

– зміна розміру фотографії;

– нормалізація пікселів;

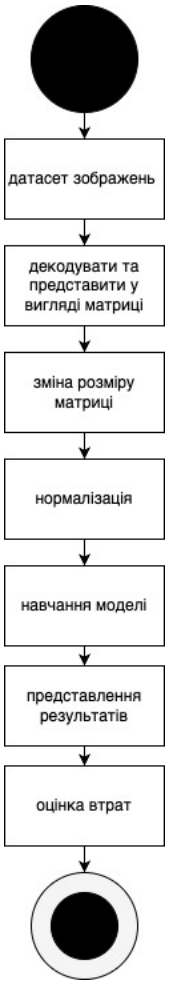

Рисунок 2.2 – Діаграма діяльності системи покращення якості фотографії

Імплементація компонента «Виведення результатів»: відображення результатів роботи нейронної мережі після навчання.

Для нормального функціонування моделі потрібне використання бібліотек мови Python, які є у відкритого доступу, тобто потенційних проблем з зупинкою і повним видаленням модулів не має виникнути.

Були використані такі бібліотеки:

- TensorFlow – надає функціонал для створення власної моделі.
- NumPy – надає функціонал для легкого та швидкого обчислення, містить в собі функціонал для роботи над матрицями, що в нашому випадку дуже зручно, бо фотографія для функціонування та нормальної роботи системи переводиться у матричний вигляд.
- Glob – надає функціонал для зручного отримання декількох світлин з теки.
- PIL – надає функціонал для гнучкого поводження з світлинами, дає можливість легко працювати з ними.
- Matplotlib – надає функціонал для виведення графік і не тільки, в нашому випадку початкової та кінцевої фотографії
- Keras - це високорівнева бібліотека для машинного навчання, яка працює поверх бібліотеки TensorFlow. Вона надає простий та інтуїтивно зрозумілий інтерфейс для створення, навчання та оцінки моделей глибинного навчання.

3.2 Висновки до розділу

У розділі було представлено структуру системи покращення якості фотографій. На основі компонентної діаграми та опису компонентів системи було надано інформацію про їх функціональність та взаємодію.

Компонент "Зберігання даних" відповідає за зберігання наборів даних, що використовуються в системі. Компонент "Попередня обробка" виконує необхідні зміни вихідних даних, такі як перетворення в матричне представлення, зміна розміру та нормалізація пікселів. Компонент "Навчання моделі" виконує основну логіку роботи системи, навчаючи модель на оброблених даних. Компонент "Виведення та збереження результатів" виводить і зберігає покращені фотографії.

Також були описані інтерфейси, які визначають взаємодію компонентів системи. Інтерфейс IDP передає сирі дані в оперативну пам'ять для попередньої обробки. Інтерфейс ICP передає оброблені дані з компонента попередньої обробки в процес навчання моделі. Інтерфейс IDTS передає результати навчання моделі для відображення та зберігання.

Для реалізації системи використовуються різні бібліотеки Python, такі як TensorFlow, NumPy, Glob, PIL, Matplotlib та Keras. Ці бібліотеки надають необхідну функціональність для створення моделей, обробки даних, відображення результатів та маніпулювання зображеннями.

Загалом, у цьому розділі подано огляд структури системи покращення якості фотографій, пояснено функціональність кожного компонента та інтерфейси між ними, а також вказано на бібліотеки, які використовуються для реалізації системи.

4 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Типова архітектура згорткової нейронної мережі

Згорткові нейронної мережі, відомі також як CNN, являють собою варіант нейронних мереж з багатьма шарами, створених з метою розпізнавання візуальних патернів на зображеннях у пікселях. У термінології CNN, термін "згортки" відноситься до математичної операції. Це є різновидом лінійної операції, в якій дві функції множаться з метою отримання нової функції, що показує, як форма однієї функції може бути модифікована іншою. Іншою мовою, це процес, коли два зображення, які представлені у вигляді матриць, множаться для виробництва результуючих даних, що використовуються для витягу інформації з зображення. Хоча CNN мають схожу структуру з іншими нейронними мережами, наявність послідовності згорткових шарів робить їх більш складними. Без шарів, CNN не може працювати належним чином [5].

Для CNN знаходять все нові використання у сфері детекції, обробки та інших випадках.

Шар згортки - є фундаментом CNN і спеціалізується на виконанні конволюційних операцій. У цьому шарі ключовим елементом є ядро, який здійснює саму згортку. Ядро переміщується по зображенню в горизонтальному та вертикальному напрямках залежно від величини кроку, скануючи всю поверхню зображення. Воно має менші розміри, ніж зображення, але більшу глибину. Це означає, що у випадку зображення з трьома каналами, ядро буде мати невеликі просторові розміри, але його глибина буде рівною трьом каналам.

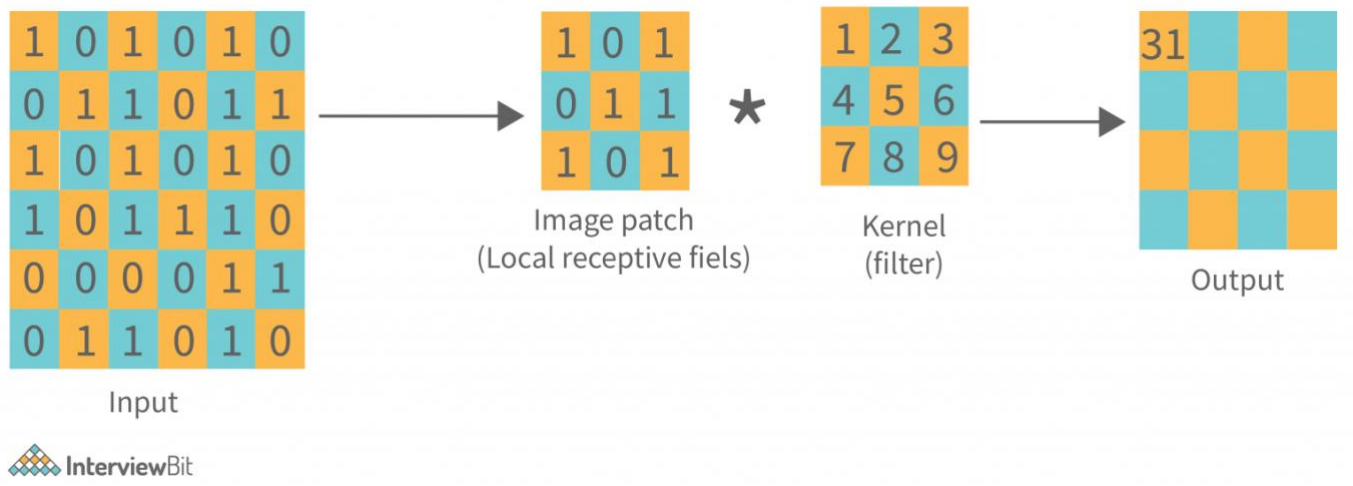

Рисунок 3.1 – Приклад роботи ядра згортки [5]

Крім конволюційних операцій, в конволюційних шарах існує інший суттєвий компонент - це функція нелінійної активації. Результати лінійних операцій, таких як конволюція, проходять через цю функцію нелінійної активації. Хоча раніше зазвичай використовувалися гладкі нелінійні функції, такі як функція сигмоїда або гіперболічний тангенс, які імітують поведінку біологічних нейронів, сьогодні блок лінійної ректифікації, або ReLU, став найпопулярнішою функцією нелінійної активації [5].

Шар з повним зв'язуванням обробляє дані у вигляді витягнутого вектора, при цьому кожен елемент вхідних даних зв'язаний з усіма нейронами в шарі. Далі цей витягнутий вектор передається через декілька FC шарів, в яких відбуваються математичні операції. На цій стадії починається процес класифікації. Зазвичай, шари з повним зв'язуванням розміщені в кінцівці структури CNN [5].

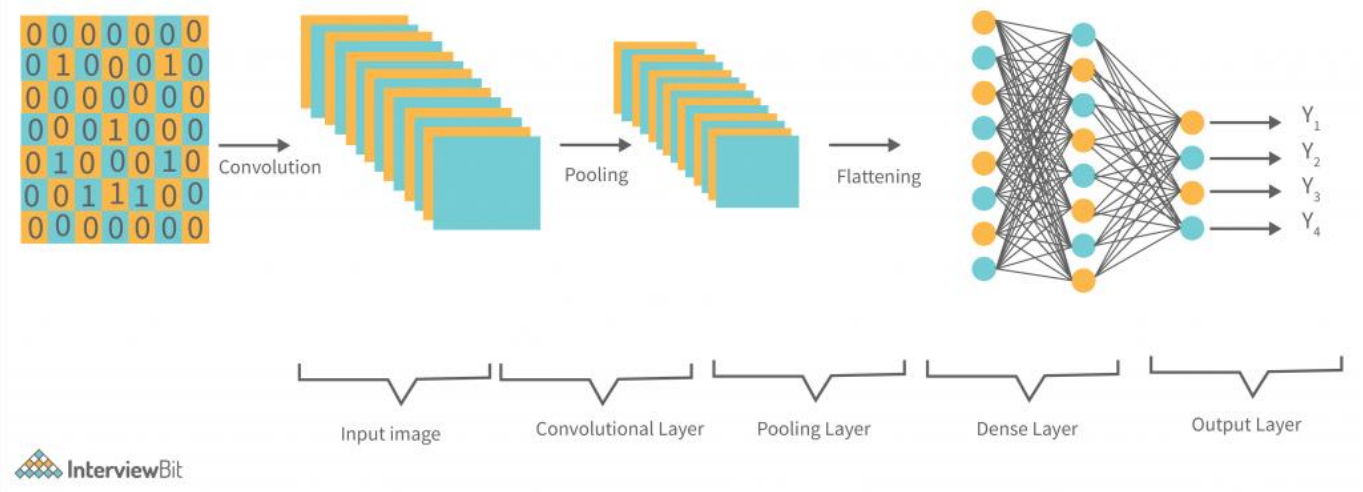

Рисунок 3.2 – Архітектура типової CNN моделі з повним зв'язуванням [5]

4.2 AlexNet архітектура згорткової нейронної мережі

AlexNet – архітектура згорткової нейронної мережі яка була розроблена Алексом Кризевським та стала популярною у 2012 році після того, як посіла перше місце на конкурсі ILSVRC. Перші 5 шарів є згортковими шарами. Вони використовують фільтри з різними розмірами ядра (3x3, 5x5) для виявлення різних рівнів ознак у вхідних зображеннях. Кожен згортковий шар слідується шаром пулінгу, який здійснює підвибіркове зведення для зменшення розмірності. Ця архітектура стала проривною бо вперше було використано функція активації ReLU у великих масштабах та була однією із перших, що використовувала GPU для прискорення. Алекс Кризевський встановив початковий шаблон для глибоких нейронних мереж у сфері зорового розпізнавання об'єктів [**Error! Reference source not found.**].

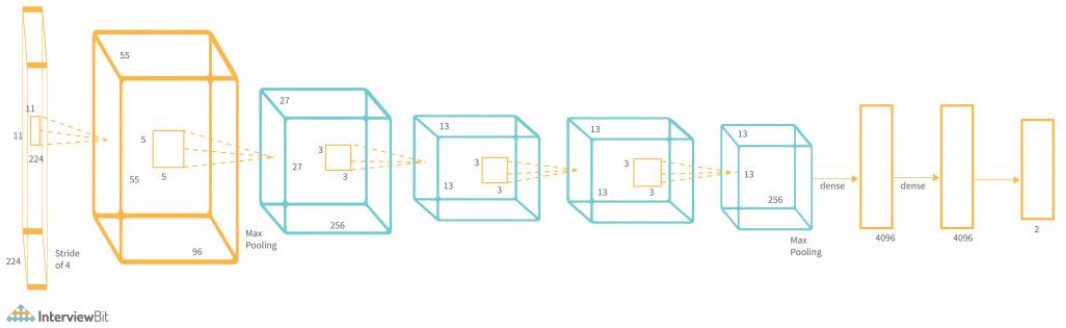

Рисунок 3.3 – Архітектура AlexNet [5]

4.3 Zero-DCE архітектура згорткової нейронної мережі

Zero-DCE має відносну схожість з архітектурою AlexNet, вона має 7 шарів, 6 згорткових та один, останній, повно зв’язний шар. Майже всі згорткові шари у моделі однакової розмірності, також використовується функція активації ReLU, як і в AlexNet, для лінійного моделювання вихідних даних. Але у цих двух архітектур є деякі основні відмінності, а саме:

а) Архітектури були створені для різних цілей. Zero-DCE створена для покращення зображень при недостатньому освітлені, а модель створена Алексом Кризевським для класифікації.

б) Архітектура Zero-DCE є більш простою, в ній менша кількість шарів, немає використання пулінгу та нормалізації партій.

в) Zero-DCE не вимагає пар вхідних/вихідних зображень для навчання, а AlexNet потребує великого набору даних для навчання для своєї задачі.

Тобто при створені Zero-DCE було використано схожий підхід, але було спрощено деякі моменти для вдалого виконання завдання покращення якості зображення при недостатньому освітленні шляхом змінення кривих.

4.4 Недоліки та модифікування Zero-DCE моделі

Zero-DCE архітектура дуже гарно працює у виконання завдання покращення якості зображення при недостатньому освітленні, але в неї є багато недоліків. Так як запропонована у статті [1] модель іноді довго тренується для своїх параметрів, перетворює фотографії у дуже малий розмір для поліпшення, 256x256 пікселів, та використовує малий розмір батчу, тобто використовує малу кількість зразків, які передаються через мережу за один раз.

Збільшення розмірності дасть можливість більш гарного поліпшення фото за рахунок збільшення розмірності шарів. За рахунок цієї зміни, на більш ефективних обчислювальних машинах, можна збільшити ефективність системи. Також можна збільшити розмір батчу, що теж вплине на швидкість тренування та використання системи.

Для більш ефективного покращення результатів можна спробувати такі підходи:

а) Використання ансамблювання моделей: Застосування ансамблювання, тобто комбінації декількох моделей покращення зображень, може дати кращі результати, оскільки кожна модель може мати свої сильні сторони та спроможності.

б) Використання методів передобробки: Перед використанням моделей покращення зображень можна застосувати методи передобробки, такі як нормалізація освітлення, вирівнювання кольорів або зменшення шуму, що допоможе покращити якість вхідних зображень та сприяти кращим результатам після обробки.

в) Використання інших наборів даних для навчання: Застосування різних наборів даних для навчання моделей покращення зображень може

допомогти покращити їх універсальність та здатність до адаптації до різних умов зйомки.

г) Застосування обрізки зображень: Під час обробки зображень можна застосувати метод обрізки, щоб зосередитися на певних регіонах зображення, де поліпшення є найбільш важливим, що може дати кращі результати у визначених областях.

д) Оптимізація параметрів моделей: Тривалість тренування моделей може бути скорочена шляхом оптимізації параметрів моделі та налаштування гіперпараметрів, що дозволить досягти кращої продуктивності та швидкості роботи моделі.

е) Використання апаратного прискорення: Застосування апаратного прискорення, такого як використання графічних процесорів (GPU) або спеціалізованих тензорних процесорів (TPU), може допомогти прискорити обробку зображень та зменшити час, необхідний для покращення фотографій.

ж) Використання оптимізованих алгоритмів інференсу: Під час застосування моделей до реального часу чи масової обробки зображень, використання оптимізованих алгоритмів інференсу може покращити швидкість та продуктивність системи.

Ансамблювання є гарним варіантом покращення моделі, цей підхід заснований на припущенні, що різні моделі можуть мати різні помилки та прихильності. Поєднання моделей з різними архітектурами, алгоритмами навчання або наборами даних дозволяє знизити загальну помилку і покращити універсальність моделі. Також таким способом можна знизити дисперсію прогнозів, покращити універсальність, зменшити помилку та інше.

Можна використати схожий підхід, додати частково функціонал однієї моделі, до іншої, розширити модель використовуючи іншу, але не прямим способом, не

взаємодією між двома моделями, а модифікацією додавання додаткового функціоналу з метою отримання гібриду двох архітектур.

Архітектура Zero-DCE має деякі недоліки, які вже було вище описано, але основним недоліком, який дуже виділяється при практичній роботі це наявність шумів на результуючому фото. Саме тому було знайдене рішення, подібне до ансамблювання, а саме розширення кількості шарів. До 6 згорткових шарів додається ще 3 шари в кінець, але так щоб кінцевим був старий 7 шар, тобто повно зв'язний шар. Ці шари потрібні для реалізації додатково функціонала видалення шуму до функціоналу покращення якості зображення при недостатньому освітленні. Також для нормальної роботи, треба розширити функцію втрат, потрібно включити середньоквадратичну похибку, як похибку для оцінки рівня шуму у зображеннях, в розрахунок загальної функції втрат.

4.5 Методи машинного навчання системи

4.5.1 Функції втрат

Для нормального функціонування моделі, має бути функція втрат, яка буде використовуватися при навчанні моделі. Zero-DCE специфічна модель, і тому тут треба особливий підхід, враховуючи що це ще й не звичайна Zero-DCE модель, а модифікована, бо в ній додані додаткові шари які прибирають можливий шум на фотографіях, який з'являється при роботі системи. В такому випадку потрібно використовувати комплексне вирахування втрат, яке складається з 5 різних похибок:

1) Втрата сталості кольору – втрата константності кольору використовується для виправлення потенційних відхилень кольору в покращеному зображенні [7].

2) Втрата експозиції – обмежує недостатньо/надмірно експоновані області, ми використовуємо втрату контролю експозиції. Він вимірює відстань між середнім значенням інтенсивності локальної області та заданим рівнем експозиції [7].

3) Втрата плавності освітлення – зберігає відношення монотонності між сусідніми пікселями, до кожної карти параметрів кривої додається втрата згладжування освітленості [7].

4) Втрата просторової узгодженості – сприяє просторовій когерентності покращеного зображення, зберігаючи контраст між сусідніми областями на вхідному зображенні та його покращеною версією [7].

5) Середньоквадратична похибка – використовується для оцінки рівня шуму у зображеннях.

Отже в результаті суми цих 5 параметрів ми отримуємо загальну похибку.

Функція color_constancy_loss обчислює втрату константності кольору для покращення зображень. Ця втрата сприяє збереженню сталості кольорів у покращених зображеннях.

Спочатку функція обчислює середнє значення червоного, зеленого і синього каналів (RGB) для кожного пікселя зображення. Для цього вона виконує зведення середнього значення по ширині та висоті зображення.

Далі, функція обчислює різницю між середніми значеннями червоного, зеленого і синього каналів. Це здійснюється шляхом обчислення квадрату різниці для кожної пари каналів. Наприклад, обчислюється різниця між червоним і зеленим каналами (d_rg), червоним і синім каналами (d_rb), та зеленим і синім каналами (d_gb).

На останньому кроці функція обчислює загальну втрату, яка представляє суму квадратів різниць між каналами. Це допомагає оцінити ступінь сталості кольорів у покращених зображеннях.

У результаті, функція color_constancy_loss забезпечує математичний апарат для обчислення втрати константності кольору, який використовується для покращення якості зображень.

Функція exposure_loss обчислює втрату експозиції з метою покращення зображень. Ця втрата допомагає збалансувати рівень освітлення у покращених зображеннях.

Спочатку функція обчислює середнє значення пікселів зображення, шляхом виконання зведення середнього значення по трьох RGB-каналах. Це дозволяє отримати загальне середнє значення зображення.

Далі, функція виконує згладжування середнього значення, використовуючи функцію tf.nn.avg_pool2d з визначеним розміром пулінгу. Це допомагає отримати середнє значення зображення на більшій шкалі.

Після отримання середнього значення зображення, функція обчислює квадрат різниці між цим значенням та заданим цільовим значенням експозиції (mean_val). Це дозволяє визначити ступінь відхилення експозиції у покращених зображеннях.

Загальна втрата обчислюється як середнє значення квадратів різниць між середнім значенням та цільовим значенням експозиції. Це дозволяє визначити загальну втрату експозиції та використовується для покращення якості зображень.

Таким чином, функція exposure_loss забезпечує математичну базу для обчислення втрати експозиції, яка допомагає збалансувати рівень освітлення у покращених зображеннях.

Функція illumination_smoothness_loss обчислює втрату плавності освітлення з метою покращення зображень. Ця втрата сприяє збереженню плавного переходу між різними областями освітлення у покращених зображеннях.

Спочатку функція отримує розміри батчу, висоти та ширини зображення для подальшого використання в обчисленнях.

Далі, функція обчислює кількість горизонтальних та вертикальних переходів між сусідніми пікселями зображення. Це здійснюється шляхом віднімання значень

пікселів від їх сусідів та обчислення квадратів цих різниць. Наприклад, для горизонтального переходу обчислюється різниця між пікселями справа та зліва ($x[:,1:,:,:]-x[:,:h_x-1,:,:]$), а для вертикального переходу - різниця між пікселями знизу та зверху ($x[:,:,1:,:]-x[:,:,:w_x-1,:]$).

Далі, функція обчислює суму квадратів горизонтальних та вертикальних переходів. Це дозволяє оцінити плавність освітлення шляхом вимірювання кількості змін у значеннях пікселів між сусідніми областями.

Останнім кроком функція обчислює загальну втрату плавності освітлення, яка представляє суму квадратів горизонтальних та вертикальних переходів, нормалізовану за розміром зображення та кількістю пікселів. Це дозволяє визначити загальну втрату плавності освітлення та використовується для покращення якості зображень.

Отже, функція illumination_smoothness_loss забезпечує математичну базу для обчислення втрати плавності освітлення.

Клас SpatialConsistencyLoss [7], що являє собою реалізацію використовується для обчислення втрати просторової послідовності. Основна математична основа цього класу полягає в застосуванні згортка для порівняння просторової послідовності між оригінальними та покращеними зображеннями.

В класі ми використовуємо чотири згортки (tf.nn.conv2d) з різними ядрами (self.left_kernel, self.right_kernel, self.up_kernel, self.down_kernel) для обчислення різниці між оригінальними та покращеними зображеннями у різних напрямках (ліво, право, вгору, вниз). Кожна згортка виконує математичну операцію згладжування та порівняння значень пікселів.

Потім використовуються квадратичні функції (tf.square) для обчислення квадрату різниці між оригінальними та покращеними зображеннями у кожному напрямку.

На виході функція повертає суму квадратів різниць усіх напрямків (loss_left + loss_right + loss_up + loss_down), що представляє втрату просторової послідовності.

Отже, клас SpatialConsistencyLoss забезпечує математичний апарат для обчислення втрати просторової послідовності, що використовується в моделі для оцінки якості покращення зображень.

4.5.2 Алгоритм оптимізації

Для даної моделі гарним варіантом буде вибір оптимізатору Adam.

Оптимізатор Adam (Adaptive Moment Estimation) є алгоритмом оптимізації, який використовується для навчання нейронних мереж. Він поєднує в собі ідеї інших популярних оптимізаторів, таких як RMSprop і Momentum.

Оптимізатор Adam використовує оцінки першого та другого моменту градієнтів параметрів моделі для оновлення ваг. Він адаптивно налаштовує швидкість навчання для кожного параметра, що дозволяє ефективніше оптимізувати модель.

У моделі оптимізатор Adam використовується для оновлення ваг моделі Zero-DCE net під час тренування. В коді він ініціалізується за допомогою зазначеного швидкості навчання (learning_rate) і встановлюється як атрибут класу. Під час тренування ваги моделі оновлюються за допомогою оптимізатора Adam з використанням обчислених градієнтів та похибок.

4.6 Висновки до розділу

У розділі було розглянуто деталі та функції втрат моделі Zero-DCE, яка використовується для покращення якості зображень при недостатньому освітленні. Ця модель пропонує комплексне вирахування втрат, яке включає втрати сталості

кольору, експозиції, плавності освітлення, просторової узгодженості та середньоквадратичну похибку для оцінки рівня шуму у зображеннях.

Для оптимізації моделі Zero-DCE використовується алгоритм оптимізації Adam. Цей алгоритм комбінує ідеї інших оптимізаторів, таких як RMSprop і Momentum, і використовує оцінки першого та другого моменту градієнтів для оновлення ваг моделі. Adam дозволяє адаптивно налаштовувати швидкість навчання для кожного параметра, що покращує ефективність оптимізації.

Загальною метою використання цих методів та функцій втрат є покращення якості зображень, збереження кольорової константності, балансування рівня освітлення, збереження плавних переходів між областями освітлення та покращення просторової узгодженості. Використання оптимізатора Adam допомагає покращити ефективність навчання та оптимізації моделі Zero-DCE.

Ці методи та алгоритм оптимізації відіграють важливу роль у розробці та вдосконаленні систем машинного навчання для покращення якості зображень. Розуміння їх математичних основ дозволяє досягти більш точних та стабільних результатів покращення зображень.

5 Програмне забезпечення

5.1 Підготовка зображень

5.1.1 Пошук датасету

Для навчання і тестування було використано датасет Dark Face Dataset з сайту [9]. Набір включає в себе 6 000 реальних зображень, що були зроблені в умовах низької освітленості, отриманих вночі в навчальних закладах, на вулицях, мостах, шляхопроводах, в парках та інших місцях [9]. Для ефективного тренування та валідації було обрано саме цей датасет, так як він має фотографії з різноманітними місцями та різноманітними об’єктами, що дає змогу гарного навчання та подальшого використання моделі. Також датасет має тільки зображення, що були зроблені в умовах низької освітленості, тому що для моделі Zero-DCE net вистачить цих даних, їй не потрібні фото гарної якості для навчання.

5.1.2 Отримання та обробка зображення

В функції load_data(image_path) відбувається зчитування та обробка фотографії. Принцип роботи цієї функції:

а) Зчитування вмісту файлу зображення з вказаного шляху image_path у бінарному форматі.

б) Декодування зображення з формату PNG.

в) Після декодування, зображення змінює свій розмір на нові значення, які вказані у IMAGE_SIZE.

г) Нормалізація зображення: Зображення нормалізується шляхом поділу значень пікселів на 255.0.

д) Повернення обробленого зображення.

Так функція load_data(image_path) готує готові для тренування дані, використовуючи тільки шлях.

5.2 Опис обраного методу покращення якості фотографії

Так як модель Zero-DCE повністю підходить, але має свої недоліки, було вирішено взяти цю модель за основу, додати 3 додаткові згорткові шари для видалення шуму, збільшити розмір картинки яку може приймати модель, збільшити кількість батчів, розширити функцію втрат, включити до неї середньоквадратичну похибку.

На піку можливостей обчислювальної машини вдалось встановити розмір матриці, або інакше кажучи, зображення, 512x512 пікселів. Розмір батчу, або так званий розмір пакету, вдалось підняти з 16 до 32. Була зроблена спроба підвищити розмір фотографій до 1024x1024, але при такій конфігурації моделі вже технічних можливостей комп’ютера не вистачало. Аналогічна ситуація була при спроби виставити розмір пакету 64 для більш швидкого навчання, модель при старті навчання використала всю відео пам'ять GPU, що змусило зупинити програму. Зрештою, ще було додано 3 згорткові шари, які допомагають прибрати шум, на фінальній фотографії. Для повного функціонування системи не вистачає тільки функції втрат для нових доданих шарів. Для оцінки рівню шуму зазвичай використовують середньоквадратичну похибку, тому і візьмемо її, для оцінки.

5.3 Архітектура системи

Архітектура створеної системи представлена нижче у вигляді зображення.

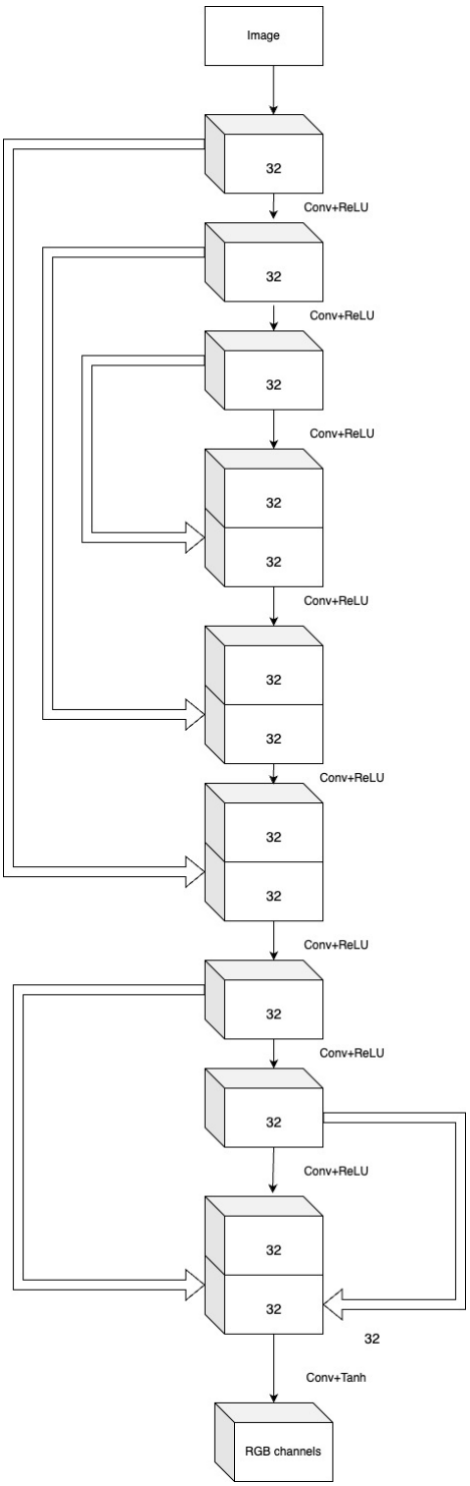

Рисунок 5.1 – Архітектура модифікованої Zero-DCE

5.4 Навчання моделі покращення якості зображення

Після підготовки даних модель готова до навчання, для тренування моделі робляться такі етапи:

а) Створення моделі: Необхідно створити об'єкт класу Model і сконфігурувати його, включаючи встановлення параметрів оптимізатора, функцій втрат та метрик для тренування.

б) Цикл тренування: Проведення циклу тренування, який складається з наступних кроків:

– Подача навчальних зображень на вхід моделі та отримання прогнозованих вихідних значень.

– Обчислення значення втрат за допомогою методу compute_losses.

– Обчислення градієнтів відносно параметрів моделі за допомогою автоматичного диференціювання.

– Оновлення параметрів моделі з використанням оптимізатора за допомогою методу apply_gradients.

в) Моніторинг і збереження метрик тренування та втрат.

г) Оцінка моделі: Після завершення тренування можна оцінити модель на перевірочному наборі даних. Це включає обчислення метрик ефективності та втрат на перевірочних зображеннях, що дозволяє оцінити якість моделі та виявити ознаки перенавчання.

д) Налаштування та вдосконалення: В залежності від результатів оцінки моделі можуть бути внесені зміни у конфігурацію моделі або алгоритмів тренування, таких як зміна швидкості навчання, збільшення обсягу даних або зміна архітектури моделі.

е) Збереження моделі: Після завершення тренування і досягнення задовільної ефективності моделі, її можна зберегти для подальшого використання. Це

включає збереження параметрів моделі та конфігурації для використання в майбутньому.

Кожен ітераційний цикл тренування включає подачу даних, обчислення втрат, обчислення градієнтів та оновлення параметрів моделі з використанням оптимізатора. Цей процес повторюється протягом декількох епох або до досягнення визначеного критерію зупинки, наприклад, зменшення втрат на перевірочних даних або досягнення заданої точності. Загалом, тренування моделі ZeroDCE включає постійне циклічне вдосконалення параметрів моделі з використанням втрат та градієнтного спуску, з метою покращення якості покращення зображень.

5.5 Аналіз результатів

Аналізувати результати та оцінювати їх, це завжди надзвичайно важливо. За допомогою оцінки результатів ми можемо зрозуміти чи працює та чи інша модель в рамках очікуваних результатів, або в рамках випробувань схожих моделей. В цій моделі, як вже було сказано розділами вище використовується складна функція втрат, вона являє собою суму 5 різних втрат, а саме:

1) Втрата сталості кольору.
2) Втрата експозиції.
3) Втрата плавності освітлення.
4) Втрата просторової узгодженості.
5) Середньоквадратична похибка.

Отже модель було навчено в декілька етапів, тому можна оцінити роботу системи на різних проміжках навчання.

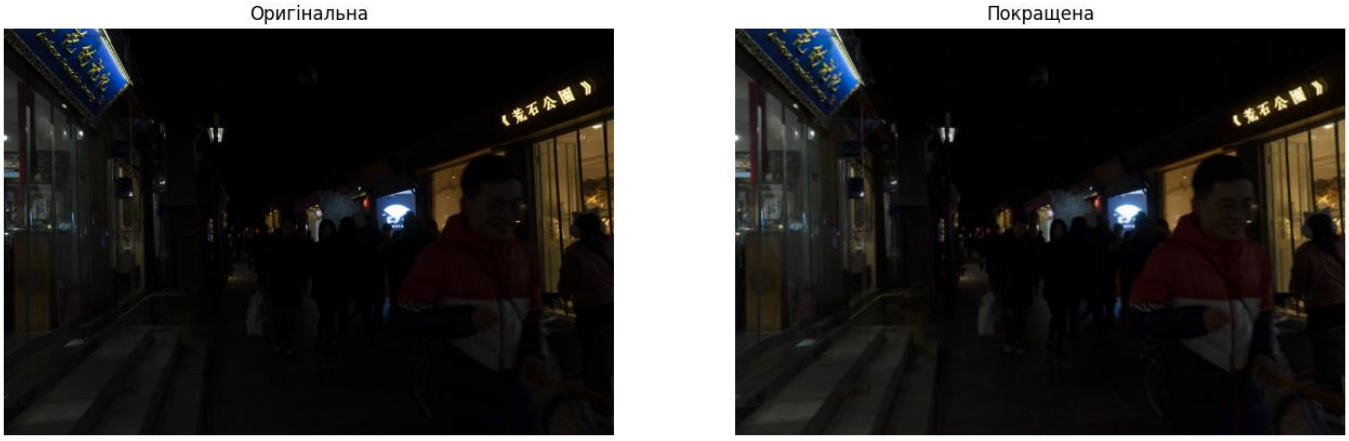

Рисунок 5.2 – Результат тестування системи після 5 епох навчання

Як бачимо після 5 епох навчання результат тестування моделі на тестових даних малопомітний.

На п’ятій епосі значення функції втрат дорівнює 2.7338, а його складові в свою чергу такі:

– illumination_smoothness_loss: 1.1011e-10;
– spatial_constancy_loss: 0.0015;
– color_constancy_loss: 0.0047;
– exposure_loss: 2.7265;
– mse_loss: 0.0011.

Продовжимо навчання та зробимо порівняння. Отже після ще 45 епох навчання ще раз перевіряємо те саме зображення, та значення функції втрат.

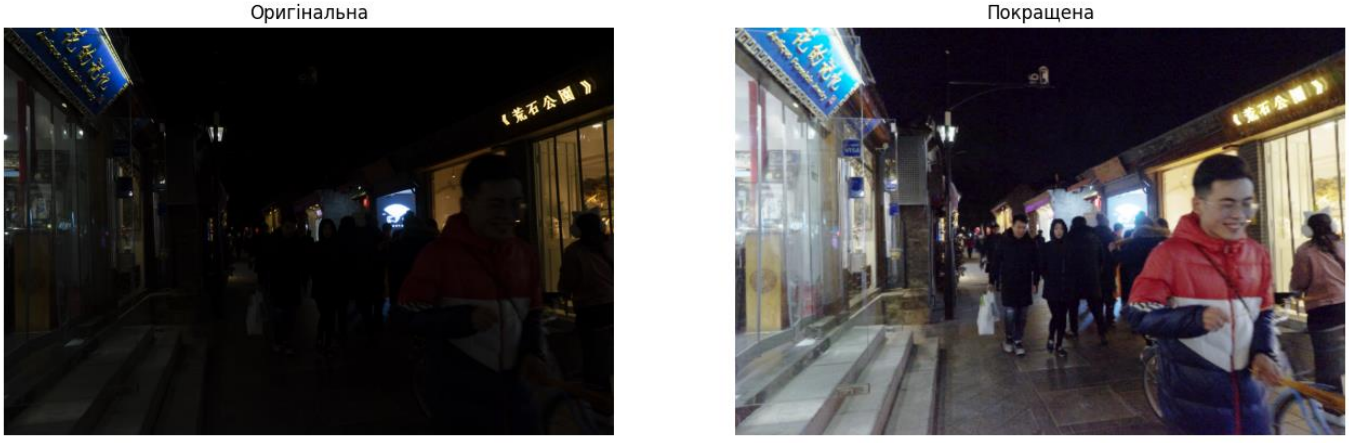

Рисунок 5.3 – Результат тестування системи після 50 епох навчання

Вже після 50 епох навчання система робить непогані успіхи. Значення функції загальної втрати впало і тепер це 1.3348, а його складові в свою чергу такі:

– illumination_smoothness_loss: 2.2355e-11;
– spatial_constancy_loss: 0.2065;
– color_constancy_loss: 0.0610;
– exposure_loss: 0.9093;
– mse_loss: 0.1580.

Продовжуємо навчання ще на 50 епох, щоб зробити аналіз чи є сенс цій моделі дуже довго вчитися, чи можливо є якийсь поріг, після якого навчання стає більш довгим і перестає бути таким вигідним якщо вважати використані ресурси.

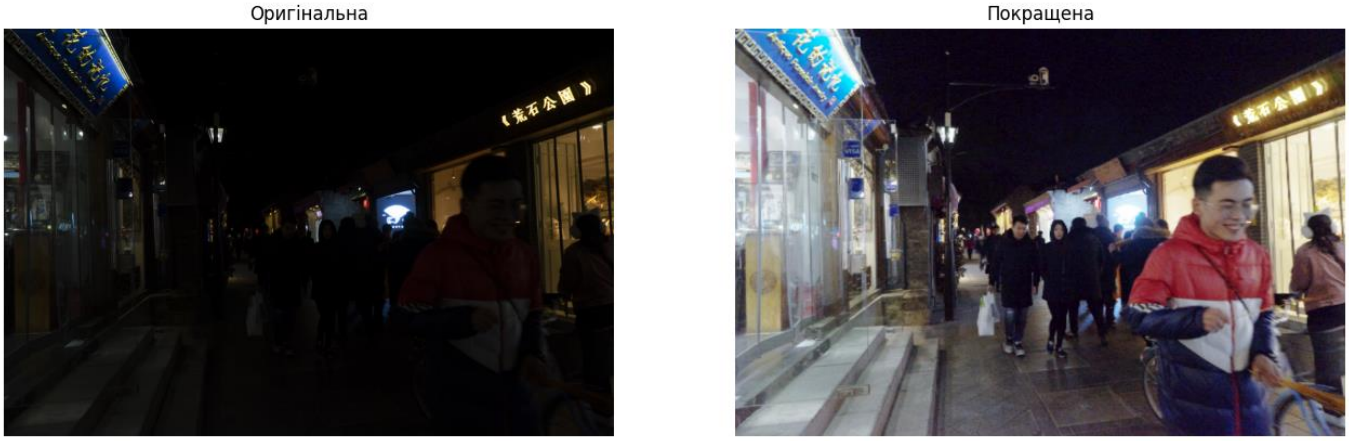

Рисунок 5.3 – Результат тестування системи після 100 епох навчання

Очевидно, що після 100 епох навчання система не дуже краще показує результат. Значення функції загальної втрати впало і тепер це 1.3269, а його складові в свою чергу такі:

– illumination_smoothness_loss: 6.1735e-12;
– spatial_constancy_loss: 0.2296;
– color_constancy_loss: 0.0655;

– exposure_loss: 0.8561;

– mse_loss: 0.1757.

За результатами які видно і графічно, і показниково, за рахунок функцій втрат та порівняння зображень, можна було б припинити навчання, так як система починає уповільнено навчатися. Але вирішено не зупиняти і зробити ще 900 епох навчання. Це останній етап навчання для перевірки результатів.

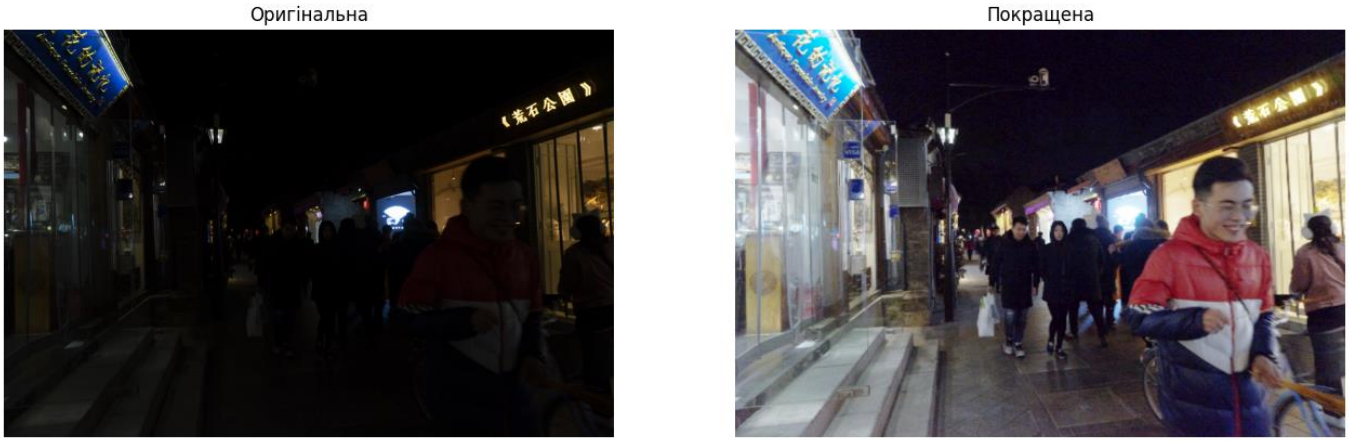

Рисунок 5.4 – Результат тестування системи після 1000 епох навчання

Графічно результат не змінився, а функція втрат залишилась майже такою самою. Значення функції загальної втрати 1.3212, а його складові в свою чергу такі:

– illumination_smoothness_loss: 0.0000e+00;

– spatial_constancy_loss: 0.2324;

– color_constancy_loss: 0.0658;

– exposure_loss: 0.8451;

– mse_loss: 0.1778.

Дивлячись на результати можна спокійно сказати що в якийсь момент, можна було зупиняти навчання, бо результат був би майже такий самий.

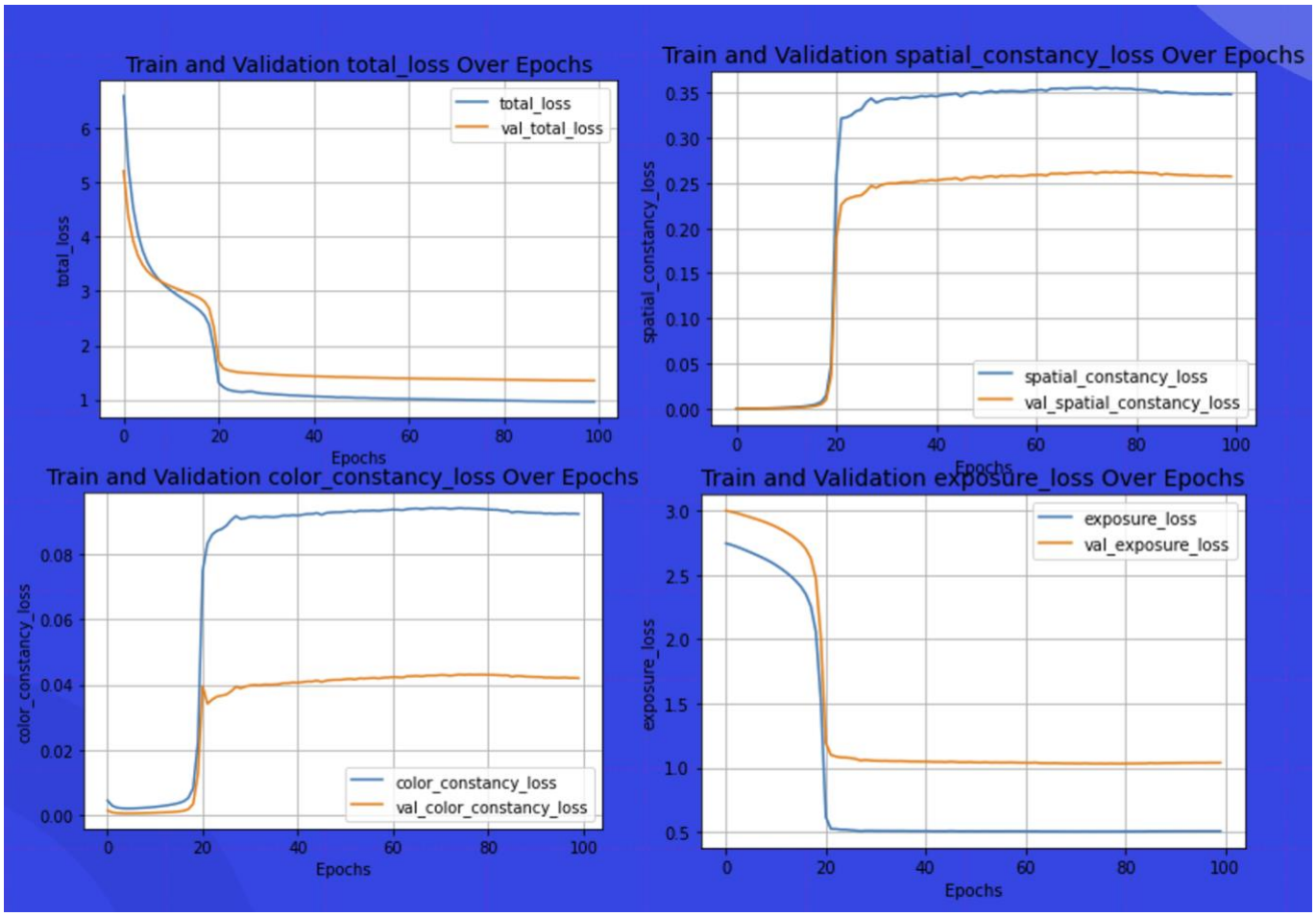

Рисунок 5.5 – Графіки деяких складових функцій втрат

Дивлячись на графіки можна сказати, що тенденція зміни значень функцій втрат якраз починає затухати на проміжку 80-100 перших епох. Тобто цю систему можна навчати всього 80-100, максимум 150 епох, щоб не перенавантажувати електронно обчислювальну машину.

5.6 Верифікація та валідація

Верифікація. Було створено систему покращення якості фотографії, що зроблені при поганому освітлені. Було оглянуті наявні рішення, та було вирішено

модифікувати Zero-DCE модель додавши 3 шари для прибирання шуму в результуючому зображені , збільшити розмір вхідних даних для кращого навчання, та збільшити розмір батчей. Результатом навчання на 1000 епохах є оцінка загальної втрати в 1.3212, якщо врахувати, що на 5 епосі було оцінено в 2.7338, а на 50 вже в 1.3348. Результат потенційно не буде краще, бо модель починає сповільнено навчатися після 100 перших епох.

Валідація. Результати тестування показали, що система ефективна, і навіть на деяких тестових фотографіях з автомобільними номерами можна чітко розглядити її номер, хоча основною метою було не це, а покращення якості фотографії зробленою в умовах поганого освітлення.

Розроблена система може бути корисною як для і для покращення фотографій звичайними людьми, так і для розпізнавання в подальшому різних об’єктів в умовах поганого освітлення, наприклад вночі, наприклад автомобільних номерів, як було виявлено під час дослідження.

5.7 Висновки до розділу

У цьому розділі було розглянуто процес навчання моделі покращення якості зображень на основі модифікованої Zero-DCE моделі. Основні кроки навчання включають створення моделі, цикл тренування, моніторинг та збереження метрик тренування та втрат, оцінку моделі та налаштування параметрів.

Під час аналізу результатів було проведено порівняння значень функції втрат після різних епох навчання. Виявлено, що після певної кількості епох навчання досягнуто точки насичення, коли додаткове навчання не приносить значних покращень. Значення функції втрат практично залишаються на одному рівні після досягнення цієї точки.

Аналіз функцій втрат та порівняння зображень також дозволили визначити, що тренування моделі протягом 80-100 перших епох може бути достатнім для досягнення бажаного рівня покращення зображень. Подальше тренування може вести до перенавантаження та неефективного використання обчислювальних ресурсів.

Усі ці висновки допомагають установити оптимальну кількість епох для навчання моделі, що сприяє покращенню якості зображень. Такий аналіз також важливий для ефективного використання обчислювальних ресурсів та збереження часу під час тренування.

ВИСНОВКИ

В даній роботі було проведено дослідження та розроблено систему покращення якості фотографій, зроблених при поганому освітленні або в нічний час. В процесі роботи були використані різні підходи, включаючи математичні алгоритми та методи машинного навчання.

Спочатку було проведено аналіз проблеми покращення якості фотографій та огляд наявних методів. Виявлено, що покращення контрасту та яскравості зображень можна досягти за допомогою алгоритмів обробки гістограми, таких як глобальне та локальне розтягнення гістограми. Також було виявлено, що для покращення фотографій можна використовувати моделі машинного навчання, зокрема Zero-DCE, яка дозволяє покращити контраст та якість зображень без використання посилань на початкове зображення.

На основі цих висновків була розроблена компонентна модель системи покращення якості фотографій. Модель включає компоненти, такі як сховище даних, попередня обробка, навчання моделі та виведення результатів. Кожен компонент має свою функціональність і взаємодіє з іншими компонентами для досягнення покращення якості фотографій. Для реалізації системи використовуються різні бібліотеки мови Python, такі як TensorFlow, NumPy, PIL, Matplotlib та Keras.

Результати дослідження та розробки підтверджують ефективність використання методів обробки гістограми та моделей машинного навчання для покращення якості фотографій. Застосування розробленої системи може бути корисним у різних сферах, де важлива якість зображень, наприклад, в фотографії, медицині, безпеці тощо.

Однак, слід зазначити, що дослідження в цій роботі має свої обмеження. Деякі алгоритми та моделі можуть бути чутливими до шуму або зайвої обробки, що може призводити до втрати деталей або виникнення артефактів на зображенні. Також, вибір

оптимальних параметрів для кожного конкретного зображення може бути складним завданням. Для подальшого розвитку системи можна розглянути використання більш складних моделей машинного навчання або комбінацію різних методів для досягнення ще кращих результатів.

В цілому, робота висвітлює важливу проблему покращення якості фотографій та пропонує ефективний підхід до розв'язання цієї задачі. Результати дослідження можуть бути використані для подальшого вдосконалення систем покращення якості зображень і сприяти подальшому розвитку області обробки зображень та машинного навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. C. Guo et al., "Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement", IEEE, 2020, August 5.
2. Lowe D. Distinctive image features from scale-invariant keypoints, cascade filtering approach. IJCV №60 – 2004. – p. 91 – 110
3. Maslianko, P. P., Sielskyi, Y. P., “МЕТОД СИСТЕМНОЇ ІНЖЕНЕРІЇ СИСТЕМ НЕЙРОННОГО МАШИННОГО ПЕРЕКЛАДУ”, KPI Science News, 2021, August 31.
4. Penker M., Eriksson H.-E. Business Modeling with UML: Business Patterns at Work. Wiley & Sons, Incorporated, John, 2008. 480 p.
5. Benjamin Planche, Eliot Andres, "Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras", Packt Publishing, 2019, May 30.
6. CNN Architecture – Detailed Explanation [Електронний ресурс] – Режим доступу до ресурсу: https://www.interviewbit.com/blog/cnn-architecture/
7. Shaees et al., "Facial Emotion Recognition Using Transfer Learning", IEEE, 2020, September 20.
8. Zero-DCE for low-light image enhancement [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/examples/vision/zero_dce/
9. RAKSHIT S. Dark Face Dataset, Face Detection in Low Light Condition [Електронний ресурс] / SOUMIK RAKSHIT // kaggle.com. – Mode of access: https://www.kaggle.com/datasets/soumikrakshit/dark-face-dataset.

Додаток А

Лістинги програм

```
import os
import tensorflow as tf
from tensorflow import keras
from keras import layers
from glob import glob
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
import numpy as np
class ImageEnhancer:
    def __init__(self, model):
        self.model = model

    def get_plot(self, images, titles, figure_size=(12, 12)):
        fig = plt.figure(figsize=figure_size)
        for i in range(len(images)):
            fig.add_subplot(1, len(images), i + 1).set_title(titles[i])
            _ = plt.imshow(images[i])
            plt.axis("off")
        plt.show()

    def enhance_image(self, original_image):
        image = keras.preprocessing.image.img_to_array(original_image)
        shape = tf.shape(image)
        height, width = shape[0], shape[1]
        image = image.astype("float32") / 255.0
        image = np.expand_dims(image, axis=0)
        image = tf.image.resize(image, [512, 512])
        output_image = model(image)
        output_image = tf.image.resize(output_image, [height, width])
        output_image = tf.cast((output_image[0, :, :, :] * 255), dtype=np.uint8)
```

```
        output_image = Image.fromarray(output_image.numpy())
        return output_image

    def enhance_images(self, image_paths):
        for image_path in image_paths:
            original_image = Image.open(image_path)
            enhanced_image = self.enhance_image(original_image)
            self.get_plot(
                [original_image, enhanced_image],
                ["Оригінальна", "Покращена"],
                (16, 12)
            )

    def save_enhanced_image(self, enhanced_image, file_name, save_dir):
        if not os.path.exists(save_dir):
          os.makedirs(save_dir)
        save_path = os.path.join(save_dir, 'enhanced_' + file_name+'.jpg')
        enhanced_image.save(save_path)
class SpatialConsistencyLoss(keras.losses.Loss):
    def __init__(self, **kwargs):
        super(SpatialConsistencyLoss, self).__init__(reduction="none")

        self.left_difference_kernel = tf.constant(
            [[[[0, 0, 0]], [[-1, 1, 0]], [[0, 0, 0]]]], dtype=tf.float32
        )
        self.right_difference_kernel = tf.constant(
            [[[[0, 0, 0]], [[0, 1, -1]], [[0, 0, 0]]]], dtype=tf.float32
        )
        self.up_difference_kernel = tf.constant(
            [[[[0, -1, 0]], [[0, 1, 0]], [[0, 0, 0]]]], dtype=tf.float32
        )
        self.down_difference_kernel = tf.constant(
            [[[[0, 0, 0]], [[0, 1, 0]], [[0, -1, 0]]]], dtype=tf.float32
```

```
        )

    def call(self, true_images, predicted_images):
        original_images_mean = tf.reduce_mean(true_images, axis=3, keepdims=True)
        enhanced_images_mean = tf.reduce_mean(predicted_images, axis=3, keepdims=True)

        pooled_original_mean = tf.nn.avg_pool2d(
            original_images_mean, ksize=4, strides=4, padding="VALID"
        )
        pooled_enhanced_mean = tf.nn.avg_pool2d(
            enhanced_images_mean, ksize=4, strides=4, padding="VALID"
        )

        original_left_difference = tf.nn.conv2d(
            pooled_original_mean, self.left_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
        )
        original_right_difference = tf.nn.conv2d(
            pooled_original_mean, self.right_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
        )
        original_up_difference = tf.nn.conv2d(
            pooled_original_mean, self.up_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
        )
        original_down_difference = tf.nn.conv2d(
            pooled_original_mean, self.down_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
        )

        enhanced_left_difference = tf.nn.conv2d(
            pooled_enhanced_mean, self.left_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
        )
        enhanced_right_difference = tf.nn.conv2d(
            pooled_enhanced_mean, self.right_difference_kernel, strides=[1, 1, 1, 1],
padding="SAME"
        )
```

```
            enhanced_up_difference = tf.nn.conv2d(
                pooled_enhanced_mean, self.up_difference_kernel, strides=[1, 1, 1, 1], padding="SAME"
            )
            enhanced_down_difference = tf.nn.conv2d(
                pooled_enhanced_mean, self.down_difference_kernel, strides=[1, 1, 1, 1],
padding="SAME"
            )

            loss_left = tf.square(original_left_difference - enhanced_left_difference)
            loss_right = tf.square(original_right_difference - enhanced_right_difference)
            loss_up = tf.square(original_up_difference - enhanced_up_difference)
            loss_down = tf.square(original_down_difference - enhanced_down_difference)

            return loss_left + loss_right + loss_up + loss_down
        def build_model():
            input_img = keras.Input(shape=[None, None, 3])
            conv1 = layers.Conv2D(
                32, (3, 3), strides=(1, 1), activation="relu", padding="same"
            )(input_img)
            conv2 = layers.Conv2D(
                32, (3, 3), strides=(1, 1), activation="relu", padding="same"
            )(conv1)
            conv3 = layers.Conv2D(
                32, (3, 3), strides=(1, 1), activation="relu", padding="same"
            )(conv2)
            conv4 = layers.Conv2D(
                32, (3, 3), strides=(1, 1), activation="relu", padding="same"
            )(conv3)
            int_con1 = layers.Concatenate(axis=-1)([conv4, conv3])
            conv5 = layers.Conv2D(
                32, (3, 3), strides=(1, 1), activation="relu", padding="same"
            )(int_con1)
            int_con2 = layers.Concatenate(axis=-1)([conv5, conv2])
```

```
        conv6 = layers.Conv2D(
            32, (3, 3), strides=(1, 1), activation="relu", padding="same"
        )(int_con2)
        int_con3 = layers.Concatenate(axis=-1)([conv6, conv1])

        conv7 = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(int_con3)
        conv8 = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(conv7)
        int_con4 = layers.Concatenate(axis=-1)([conv8, conv7])
        conv9 = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(int_con4)

        x_r = layers.Conv2D(30, (3, 3), strides=(1, 1), activation="tanh", padding="same")(
            int_con4
        )
        return keras.Model(inputs=input_img, outputs=x_r)

class DataLoader:
    def __init__(self, image_size, batch_size, max_train_images):
        self.image_size = image_size
        self.batch_size = batch_size
        self.max_train_images = max_train_images

    def load_data(self, image_path):
        image = tf.io.read_file(image_path)
        image = tf.image.decode_png(image, channels=3)
        image = tf.image.resize(images=image, size=[self.image_size, self.image_size])
        image = image / 255.0
        return image

    def generator_data(self, images):
        dataset = tf.data.Dataset.from_tensor_slices(images)
        dataset = dataset.map(self.load_data, num_parallel_calls=tf.data.AUTOTUNE)
        dataset = dataset.batch(self.batch_size, drop_remainder=True)
        return dataset
```

```
    def get_train_datasets(self, path):
        images = sorted(glob(path))
        train_images = images[:self.max_train_images]
        val_images = images[self.max_train_images:]

        train_dataset = self.generator_data(train_images)
        val_dataset = self.generator_data(val_images)

        return train_dataset, val_dataset

class Model(keras.Model):
    def __init__(self):
        super(Model, self).__init__()
        self.model = build_model()


    def compile(self, learning_rate):
        super(Model, self).compile()
        self.optimizer = keras.optimizers.Adam(learning_rate=learning_rate)
        self.spatial_constancy_loss = SpatialConsistencyLoss(reduction="none")

    def get_image_with_improved_quality(self, data, output):
        r1 = output[:, :, :, :3]
        r2 = output[:, :, :, 3:6]
        r3 = output[:, :, :, 6:9]
        r4 = output[:, :, :, 9:12]
        r5 = output[:, :, :, 12:15]
        r6 = output[:, :, :, 15:18]
        r7 = output[:, :, :, 18:21]
        r8 = output[:, :, :, 21:24]
        x = data + r1 * (tf.square(data) - data)
        x = x + r2 * (tf.square(x) - x)
```

```
    x = x + r3 * (tf.square(x) - x)
    enhanced_image = x + r4 * (tf.square(x) - x)
    x = enhanced_image + r5 * (tf.square(enhanced_image) - enhanced_image)
    x = x + r6 * (tf.square(x) - x)
    x = x + r7 * (tf.square(x) - x)
    enhanced_image = x + r8 * (tf.square(x) - x)
    return enhanced_image

def call(self, data):
    model_output = self.model(data)
    return self.get_image_with_improved_quality(data, model_output)

def illumination_smoothness_loss(self, x):
    batch_size = tf.shape(x)[0]
    h_x = tf.shape(x)[1]
    w_x = tf.shape(x)[2]
    count_h = (tf.shape(x)[2] - 1) * tf.shape(x)[3]
    count_w = tf.shape(x)[2] * (tf.shape(x)[3] - 1)
    h_tv = tf.reduce_sum(tf.square((x[:, 1:, :, :] - x[:, : h_x - 1, :, :])))
    w_tv = tf.reduce_sum(tf.square((x[:, :, 1:, :] - x[:, :, : w_x - 1, :])))
    batch_size = tf.cast(batch_size, dtype=tf.float32)
    count_h = tf.cast(count_h, dtype=tf.float32)
    count_w = tf.cast(count_w, dtype=tf.float32)
    return 2 * (h_tv / count_h + w_tv / count_w) / batch_size

def color_constancy_loss(self, x):
    mean_rgb = tf.reduce_mean(x, axis=(1, 2), keepdims=True)
    mr, mg, mb = mean_rgb[:, :, :, 0], mean_rgb[:, :, :, 1], mean_rgb[:, :, :, 2]
    d_rg = tf.square(mr - mg)
    d_rb = tf.square(mr - mb)
    d_gb = tf.square(mb - mg)
    return tf.sqrt(tf.square(d_rg) + tf.square(d_rb) + tf.square(d_gb))
```

```
def exposure_loss(self, x, mean_val=0.6):
    x = tf.reduce_mean(x, axis=3, keepdims=True)
    mean = tf.nn.avg_pool2d(x, ksize=16, strides=16, padding="VALID")
    return tf.reduce_mean(tf.square(mean - mean_val))

def compute_losses(self, data, output):
    enhanced_image = self.get_image_with_improved_quality(data, output)
    loss_illumination = 200 * self.illumination_smoothness_loss(output)
    loss_spatial_constancy = tf.reduce_mean(
        self.spatial_constancy_loss(enhanced_image, data)
    )
    loss_color_constancy = 5 * tf.reduce_mean(self.color_constancy_loss(enhanced_image))
    loss_exposure = 10 * tf.reduce_mean(self.exposure_loss(enhanced_image))

    loss_mse = tf.reduce_mean(tf.keras.losses.mean_squared_error(data, enhanced_image))

    total_loss = (
        loss_illumination
        + loss_spatial_constancy
        + loss_color_constancy
        + loss_exposure
        + loss_mse
    )


    return {
        "total_loss": total_loss,
        "illumination_smoothness_loss": loss_illumination,
        "spatial_constancy_loss": loss_spatial_constancy,
        "color_constancy_loss": loss_color_constancy,
        "exposure_loss": loss_exposure,
        "mse_loss": loss_mse
```

```
        }


    def train_step(self, data):
        with tf.GradientTape() as tape:
            output = self.model(data)
            losses = self.compute_losses(data, output)
        gradients = tape.gradient(
            losses["total_loss"], self.model.trainable_weights
        )
        self.optimizer.apply_gradients(zip(gradients, self.model.trainable_weights))
        return losses

    def test_step(self, data):
        output = self.model(data)
        return self.compute_losses(data, output)

    def save_weights(self, filepath, overwrite=True, save_format=None, options=None):
        self.model.save_weights(
            filepath, overwrite=overwrite, save_format=save_format, options=options
        )

    def load_weights(self, filepath, by_name=False, skip_mismatch=False, options=None):
        self.model.load_weights(
            filepath=filepath,
            by_name=by_name,
            skip_mismatch=skip_mismatch,
            options=options,
        )
image_size = 512
batch_size = 32
max_train_images = 400
```

```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)

data_loader = DataLoader(image_size, batch_size, max_train_images)
train_dataset, val_dataset = data_loader.get_train_datasets("/content/drive/My Drive/Диплом/lol_dataset/our485/low/*")

model = Model()
model.compile(learning_rate=0.001)

model.fit(train_dataset, validation_data=val_dataset, epochs=5)

last_test_low_light_images = sorted(glob("/content/drive/My Drive/Диплом/test/*"))
enhanceService = ImageEnhancer(model)
enhanceService.enhance_images(last_test_low_light_images[:10])

model.fit(train_dataset, validation_data=val_dataset, epochs=45)

enhanceService = ImageEnhancer(model)
enhanceService.enhance_images(last_test_low_light_images[:10])

model.fit(train_dataset, validation_data=val_dataset, epochs=50)

enhanceService = ImageEnhancer(model)
enhanceService.enhance_images(last_test_low_light_images[:10])

model.fit(train_dataset, validation_data=val_dataset, epochs=900)

model.save('/content/drive/My Drive/Диплом/zero-dce-saved-model')

model_loaded = keras.models.load_model('/content/drive/My Drive/Диплом/zero-dce-saved-model')
last_test_low_light_images_2 = sorted(glob("/content/drive/My Drive/Диплом/mytest/*"))
```

```
enhanceService = ImageEnhancer(model_loaded)
image = Image.open(last_test_low_light_images_2[-1])
out = enhanceService.enhance_image(image)
enhanceService.save_enhanced_image(out,'abc','/content/drive/My Drive/Диплом/saved')

enhanceService = ImageEnhancer(model_loaded)
enhanceService.enhance_images(last_test_low_light_images[:10])
```

Додаток Б

Ілюстративний матеріал