МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ПОГОРІЛА ЮЛІЯ ВОЛОДИМИРІВНА

Допускається до захисту:
в. о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____О. В. Зелінська
«___» _____ 20___ р.

# РОЗРОБКА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ТА ВІДНОВЛЕННЯ ПОШКОДЖЕНИХ ЗОБРАЖЕНЬ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота
(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
Т. В. Січко, доцент кафедри
інформаційних технологій,
канд. техн. наук, доцент

_____
(підпис)

Оцінка: _____ /_____ /_____
(бали / за шкалою ЄКТС / за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2024 рік

## АНОТАЦІЯ

Погоріла Ю. В. Розробка алгоритмів машинного навчання для автоматичного виявлення та відновлення пошкоджених зображень. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні технології обробки даних (Data Science)». Донецький національний університет імені Василя Стуса. Вінниця, 2024. 96 с.

Магістерське дослідження присвячено застосування технологій машинного навчання для обробки пошкоджених зображень. Проведено аналіз зображень, а також проведено практичну реалізацію алгоритмів.

Ключові слова: алгоритми машинного навчання, обробка, пошкоджені зображення

Рис. 31, табл. 3, джерел 74, додатків 1.

## ABSTRACT

Pohorila Yu. V. Development of machine learning algorithms for automatic detection and restoration of damaged images. Specialty 122 “Computer Science”, educational program “Computer Data Processing Technologies (Data Science)”. Vasyl Stus Donetsk National University. Vinnytsia, 2024. 96 p.

The master’s thesis is devoted to the application of machine learning technologies for processing damaged images. The analysis of images was carried out, as well as the practical implementation of algorithms.

Keywords: machine learning algorithms, processing, damaged images

Fig. 31, table 3, sources 74, applications 1.

# ЗМІСТ

# ВСТУП

**Актуальність дослідження.** Досить важливою для розгляду є дослідження теми застосування машинного навчання у різних сферах (виробництво, освіта, медицина), особливо при обробці містких та великих даних, таких як зображення. Зображення становлять вагоме значення для особистісного розвитку та самопрезентації людини, її професійного й кар'єрного зростання, а також наукового прогресування, оскільки питання дослідження обробки та виявлення пошкоджених зображень має місце для ще більшого розвитку. Межі вивчення використання машинного зображення також досить широкі, оскільки щоденно провідними науковцями вдається виявити проблемні моменти та нові предмети дослідження для власних розробок. Машинне навчання (ML) забезпечує чіткий та поглиблений аналіз великих обсягів даних, що робить його незамінним у майже кожній сфері. Важко уявити людство до винайдення машинного навчання й після, оскільки це дві абсолютно різні соціальні групи. Автоматизація обробки зображень зменшило помилковість людського фактору та знизило рівень відхилень у їх опрацюванні, і це важливий аспект, оскільки допущення помилок, наприклад, у зображеннях медичної категорії неприпустимо, оскільки це може вплинути на подальшу якість життя пацієнта. Ще одним важливим аспектом актуальності машинного навчання (ML) є швидкість обробки зображень, оскільки у сучасній картині світу швидкість виконання поставлених завдань вартує значних коштів. Час – це важливий показник, який економить машинне навчання (ML).

Вивченням особливостей виявленню та відновленню пошкоджених як динамічних, так і статичних зображень, займалися українські науковці та світові дослідники, такі як Колбасін В. О. [1], Тищенко В. С. [2], Алтухов В. О. [3], Кириченко І. В. [4], Захарченко Р. М. [5], Олещенко Л. М. [6], Назаркевич М. А. [7], Переяславська, С. О. [8]., Мельник К. [9], Багрій Р. І. [10], Депіка Гай [11], Клавдія Кавалларо [12] та інші.

**Мета кваліфікаційної роботи** полягає у тому, аби дослідити ефективність використання машинного навчання задля відновлення пошкоджених областей фото.

**Об’єктом** є процес обробки фото. Суть об’єкту, який є набагато ширшим поняттям, аніж предмет, полягає у виконанні усіх послідовностей дій: від формування ідеї й до реалізації процесу обробки зображень, застосовуючи машинне навчання.

**Предмет** є застосовувані під час дослідження методи машинного навчання, бібліотеки, функції, які допомогли виявити та відновити пошкоджені статичні фото.

Під час виконання кваліфікаційної роботи було використано **методи наукового дослідження**. До них належать:

1) аналіз наукових досліджень та проблематики;

2) метод алгоритмізації також було застосовано, оскільки саме він дозволяє визначити правильно завдання та виокремити шляхи для його вирішення;

3) метод формалізації застосовано для написання коду, проте із застосуванням існуючої мови програмування.

У процесі дослідження виникає необхідність розгляду **завдань**, необхідних для детального розкриття теми. Разом із практичною частиною, необхідно виділити наступні ключові моменти:

1) визначити проблемний бік дослідження та задачі, які необхідно вирішити, наприклад, якщо мова йде про «виявлення», необхідно розуміти, яке зображення виступає у ролі такого, яке варто опрацювати;

2) провести пошук та відбір зображень, що відповідають заданій умові;

3) розглянути теоретичний складник роботи машинного навчання, надати теоретичні відомості, оскільки аналіз досліджень на суміжну тематику дозволить розширити розуміння об’єкта та предмета й проблематики теми дослідження;

4) провести навчання моделі, оскільки це питання є найбільш затратним за часовими рамками.

**Наукова новизна** як невід'ємна частина будь-якого дослідження полягає у використанні об'єкта та предмета дослідження для подальшого застосування у різноманітних галузях, у яких часто використовують обробку фото: фотозйомка, модельна галузь, галузь медицини, військова справа тощо. Проблема виявлення пошкоджених зображень постала із посиленням військового стану, оскільки знизилася безпека візуально-аудіальних матеріалів, які використовують ЗМІ, тому викривлення та підробка зображень – найбільш суттєва проблема сьогодення, яку необхідно вирішити.

**Апробація дослідження**. Тези доповіді для ІІІ Міжнародної науково-практичної конференції «Прикладні аспекти сучасних міждисциплінарних досліджень» (01 листопада 2024 року, м. Вінниця) на тему: «Аналіз різновидів машинного навчання для виявлення зниження якості зображень» (Погоріла Ю. В., Січко Т. В.)

Структура кваліфікаційної роботи складається із таких частин: вступ, три розділи, 7 підрозділів, висновки, список використаних посилань (50 джерел), рисунків – 35, таблиць – 3, кількість сторінок (основна частина) – 72, всього – 96.

# РОЗДІЛ 1
# ТЕОРЕТИЧНІ ОСНОВИ МАШИННОГО НАВЧАННЯ ПРИ РОБОТІ ІЗ ЗОБРАЖЕННЯМИ

## 1.1. Основні положення Machine Learning: принципи роботи алгоритмів

Розвиток інформаційних технологій неможливий без дослідження теми обробки зображень, оскільки на сьогодні це питання стосується переважної більшості сфер життєдіяльності людини, у кожній з яких способи та засоби обробки, відновлення, редагування зображень відіграють важливу роль. Обробка зображень – це одна із основних тем, що стосується галузі машинного навчання, особливо станом на сьогодні, коли окрема частина інформації та даних може бути викривлена або до якої застосовуються методи зміни якості зображень, їх роздільної здатності тощо.

Сфери застосування такого роду алгоритмів для відновлення зображень абсолютно різні, у тому числі, якщо мова йде про відновлення фото, які мають історичне, політичне або навіть державне значення. Поліпшення якості фото, які з розвитком інформаційних технологій втратили цю властивість, а також відновлення медичних знімків, їх автоматичне формування, застосування візуальних ефектів до фото тощо – це ті аспекти, що посилюють необхідність використання машинного навчання. Йдеться як про вирішення естетичного питання предмета дослідження, так й про вирішення складних завдань обробки фото, з якими людина без спеціальних знань не зможе упоратися.

Необхідною умовою для вдалої комп’ютерної реставрації зображень є розробка алгоритмів, суть якої полягає у виконанні операцій із зображенням без повторюваного використання людських зусиль. Тобто, використовуючи методи програмування не лише для кожного конкретного випадку (за умови, якщо цього не вимагає поставлене завдання, оскільки випадків викривлених та пошкоджених фото може бути велика кількість, тому необхідно застосовувати алгоритми, які будуть ефективними у конкретних випадках також), а для пакету

фото. Наприклад, розробник може стикнутися із наступним формулюванням завдання:

1) замовник надає певну кількість фото, на яких частково втрачено інформацію;

2) фото втратили якість у зв’язку із їх стисненням;

3) було забагато використано спеціальних ефектів, які знизили якість зображення тощо.

Випадки можуть бути кардинально різні, проте частою проблемою й водночас перевагою є те, що одна проблема якості зображень випливає із іншої, наприклад, низька роздільна здатність зображень викривлює текстову інформацію на фото. Також необхідно зазначити, що процес відновлення світлин не є єдиним методом опрацювання світлин, оскільки виникають випадки, коли необхідно правильно виявити та ідентифікувати фото, віднайти конкретне зображення або навчити програму розуміти набір світлин.

Саме поняття «машинного навчання» постає як своєрідне запозичення методів «штучного розуму», він є складовою цього складного процесу та розробляє алгоритми, які допомагають комп’ютерам навчатися, вирішувати складні завдання, бачити закономірності та їм наслідувати. Наприклад, якщо виникає потреба, аби програма зрозуміла різницю між людиною та твариною, необхідно надати програмі набір даних, задати умову й спостерігати за тим, як комп’ютер на основі алгоритму вчиться виявляти різницю між людиною та твариною.

Враховуючи те, що з кожним роком кількість цифрових зображень зростає та проникає у всі можливі сфери, а для деяких платформ великий обсяг світлин становить основу, наприклад, для Instagram, який зосереджений на візуальному поданні інформації, тому у таких умовах важче знаходити потрібні зображення, інформацію. Цей процес став важчим при виконанні «вручну». Важчим став і процес пошуку текстової інформації у пошукових системах, наприклад, Mozilla Firefox, при введенні запиту в якому проблематично одразу знайти корисну й актуальну інформацію.

Машинне навчання у такому випадку стане у нагоді, оскільки воно здатне самостійно отримувати відомості, проте за умови постійного навчання моделі. Машинне навчання – це також досить широке поняття, здатне допомогти людині, наприклад, здійснювати розпізнавання мови, аналізувати текстове зображення тощо. Роботу машинного навчання можна розділити на кілька важливих етапів:

1) підготовка та підбір правильних даних;

2) здійснення перетворення даних, аби вони були більш зрозумілими для ML;

3) вибір правильної архітектури;

4) тестування зібраних даних для того, аби підбити підсумок – правильно модель зрозуміла поставлене завдання чи ні, можливо, необхідне доопрацювання;

5) оптимізування роботи моделі (за потреби);

Машинне навчання має послідовну структуру, зображену на рис. 1.1.

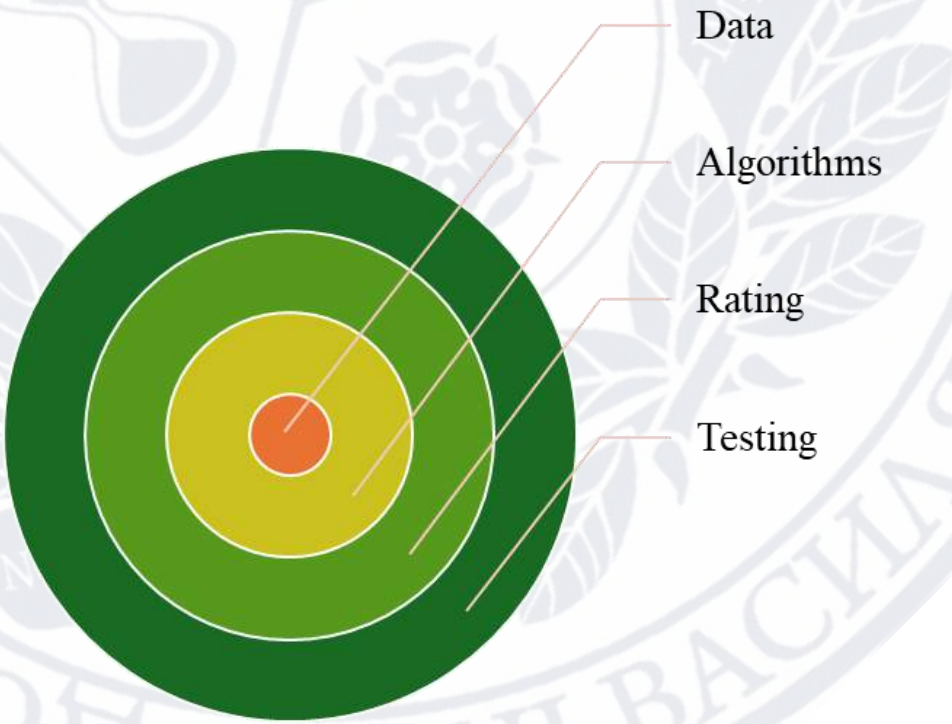


Рис. 1.1 – Схематична структура роботи машинного навчання [13]

Необхідно зауважити, що трапляються випадки, коли при обробці світлин може здійснювати помилки, проте у такому випадку цю особливість можна виправити та зробити роботу моделі машинного навчання більш якісною. У процесі роботи можуть виникати й проблемні аспекти, наприклад, якщо програмою було взято до уваги багато незначних аспектів, наприклад, реагує на незначні зміни або чіткість її роботи знижена, у такому разі програма досить активно обробила дані й відбулося перенавантаження. Аби не виникало таких проблем, необхідно, аби модель аналізувала та підбивала підсумки роботи, тобто, могла правильно аналізувати отримані дані, а не систематично та послідовно запам’ятовувала їх.

При використанні ML у роботі зі світлинами необхідно дотримуватися такого правила – якість роботи залежить від якості наданих даних, які обробляє алгоритм. Проте це не єдиний критерій, оскільки існують наступні (рис. 1.2).

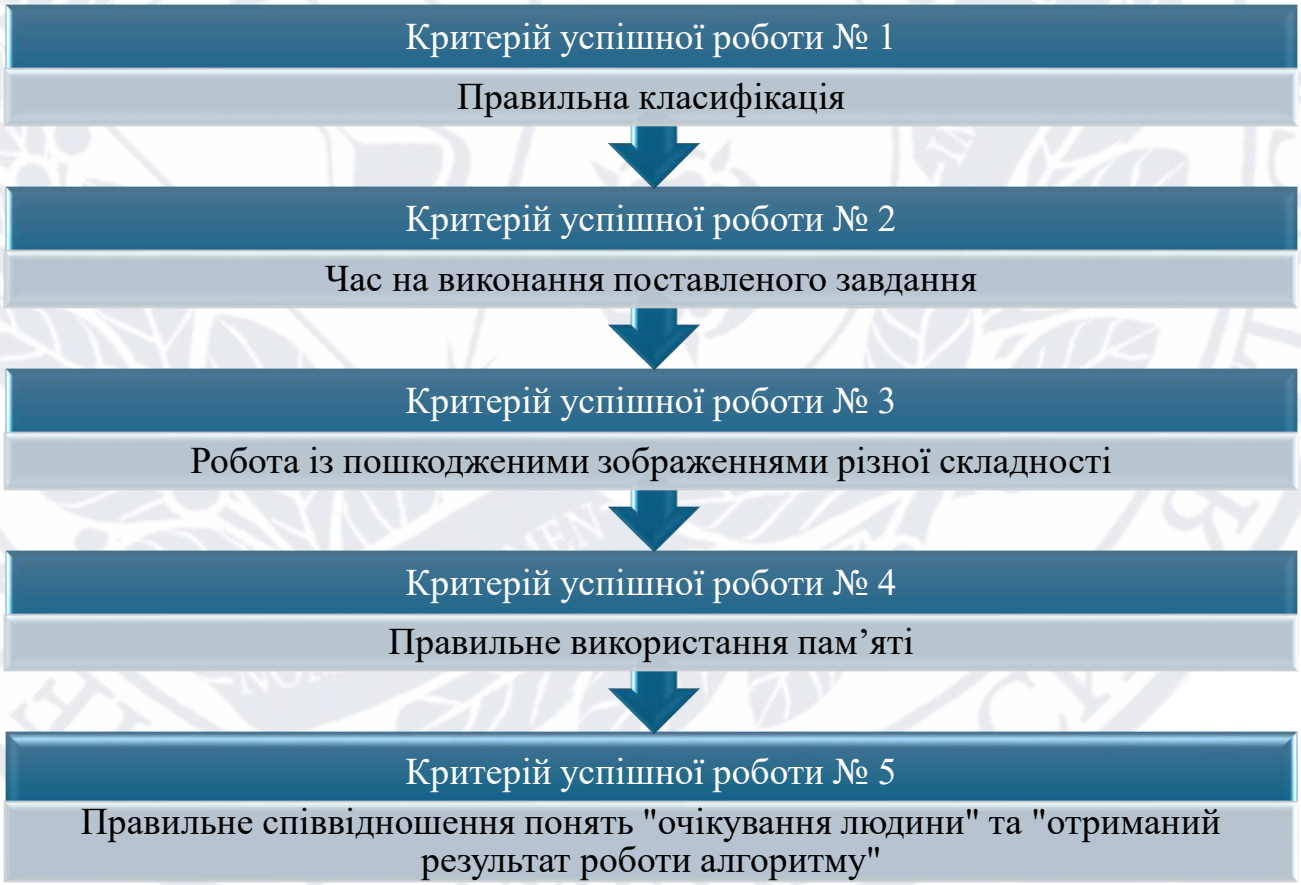


Рис. 1.2 – Критерії оцінки якості роботи алгоритму

Необхідною умовою для роботи із машинним навчанням є уміння відрізняти його від «штучного розуму», адже ці поняття досить наближені один до одного та часто перетинаються. Досить важке, проте одне із основних завдань ML є навчання програми, яка зможе конкурувати або навіть дорівнювати штучному розуму, у найбільш складних випадках – перевершувати здібності людського головного мозку. Необхідно окремо наголосити на понятті «controlled algorithms», оскільки при використанні цього виду навчання алгоритм намагається обробити дані, які мають мітки (тобто, дані, які мають мітки, завідомо мають власне цільове призначення для застосування). Проте існують й інші завдання, які можливо вирішити шляхом такого виду навчання, кожне з яких становить окремий клас як у програмуванні, так й математичному моделюванні, алгебраїчних перетворення тощо (рис. 1.3).

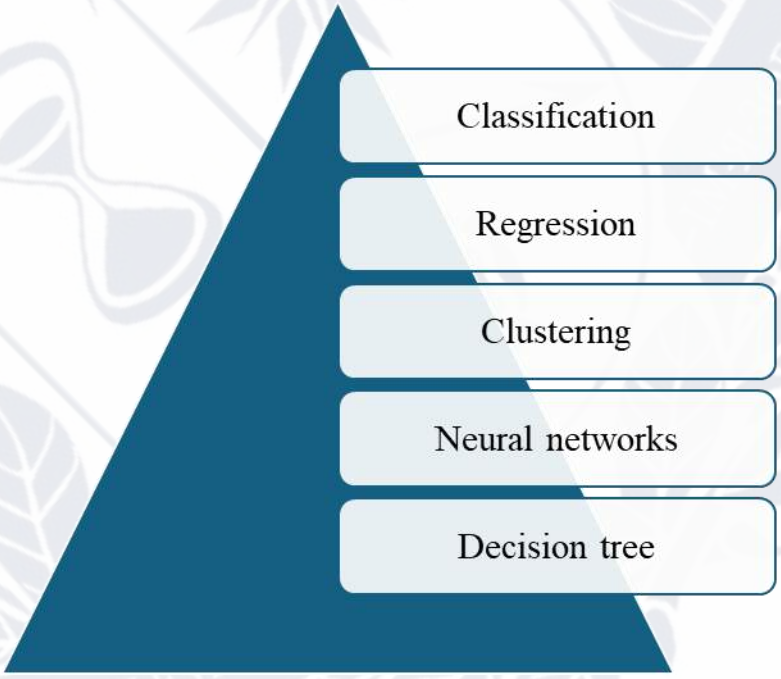


Рис. 1.3 – Групи ML

Найпершим необхідно розглянути алгоритм Баєса, які допомагають передбачити вплив виникнення однієї події на іншу [14]. Цей алгоритм називають «наївним», він став одним із перших алгоритмів, застосовуваним для машинного навчання. Раніше його застосовували для фільтрування непотрібної інформації, яка засмічує електронну пошту. Суть алгоритму полягала у тому, що провідними

інженерами було зібрано достатню кількість інформації у листах, яка не несе інформативності, а має лише активний рекламних характер, наприклад, «безкоштовно», «знижки», «увага». Згодом зібраній інформації навчали алгоритм, який автоматично сортував листи в окремі теки.

Далі розглянемо регресію як підвид машинного навчання, яку визначено як механізм, який шляхом аналізу вхідної змінної дає потенціал для розгляду однієї вихідної змінної [15]. Цю методику використовують, наприклад, під час прогнозування цін на продовольчі ресурси, нерухомість, ліки тощо на основі певних підібраних параметрів до них. Приклад застосовуваних параметрів подано у таблиці 1.1.

Таблиця 1.1 – Параметри для прогнозування через лінійну регресію

| № | Найменування досліджуваного об’єкта | Параметри |
|---|---|---|
| 1. | Зернові культури | 1) швидкість достигання;<br>2) кількість використаних пестицидів;<br>3) якість продукції;<br>4) термін зберігання;<br>5) витрати на збір та обробку зерна. |
| 2. | Нерухомість | 1) термін експлуатації будинку;<br>2) якість ремонту;<br>3) якість сантехніки;<br>4) місцерозташування;<br>5) інфраструктура;<br>6) наявність об’єктів життєвої необхідності |
| 3. | Фармацевтична продукція | 1) попит на ліки за певний період часу;<br>2) інфляція;<br>3) іноземний товар;<br>4) ліки життєвої необхідності |

Отже, за наявними критеріями можна спрогнозувати вартість тих чи інших товарів та розуміти, що саме вплинуло на зростання цін. Алгоритм переважно застосовують з метою прогнозування чисел, спираючись на лінійну залежність. Наприклад, цю техніку використовувати у багатьох випадках, особливо, якщо мова йде про попереднє визначення ціни житла. Існує два типи лінійної регресії (ЛР): проста ЛР та множинна ЛР.

Проста ЛР включає одну незалежну змінну та одну залежну (формула 1.1).

$$y = \beta_0 + \beta_1 X \tag{1.1}$$

1) Y залежна змінна
2) X незалежна змінна
3) β0 точка перетину
4) β1 нахил

Множинна ЛР включає більше однієї незалежної змінної, проте все одно одна повинна бути залежною (формула 1.2).

$$y = \beta_0 + \beta_1 X1 + \beta_2 X2 + \cdots \beta_n Xn \tag{1.2}$$

1) Y залежна змінна
2) X1, X2, Xn незалежні змінні
3) β0 точка перетину
4) β1, β2, βn нахили

Основну мету найбільш доцільно прийнято трактувати як таку, яка визначає найбільш привабливу лінію, за якої похибка між прогнозом та фактичним результатом буде мінімальною.

Ще однією не менш важливою складовою машинного навчання вважають кластеризацію, яка значно спрощує вихідні дані та поділяє їх на різні групи або кластери, групує їх за наявністю певних ознак [16]. Краще, коли дані згруповані в різні розміри та форми, і для них не надається жодних специфікацій, оскільки вони вивчаються без нагляду. Щоб полегшити спрощення та обробку великих і

складних даних у машинному навчанні, після завершення кластеризації їм надається ідентифікатор. Спрощене подання методу кластеризації зображене на рис. 1.4.

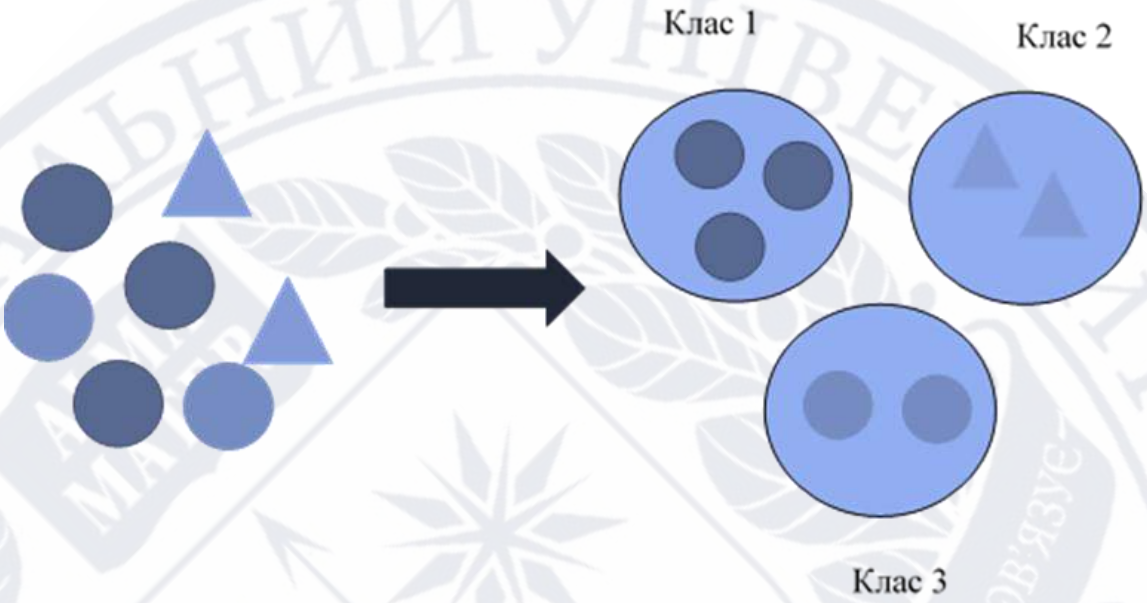


Рис. 1.4 – Схематичне зображення кластеризації

Наступним при розгляді виступає дерево рішень, застосовуване для моделювання та прогнозування. Цей алгоритм має на меті структурувати рішення, спираючись на вхідні дані, які роблять його здатним вирішувати завдання одразу кількох двох інших алгоритмів, таких як класифікація та регресія. Дерево рішень має назву «дерева», оскільки його структура має схожу будову, при якій кожен внутрішній вузол здійснює перевірку атрибуту, а кожна гілка становить значення атрибуту, а вже листовий вузол представляє крайнє рішення прогнозування [17].

Сама будова цього алгоритму подібна до природнього дерева, має ієрархічну структуру. З практичної точки зору використання дерева рішень є складним. Його реалізують за допомогою ентропії та отримання інформації. Ентропія вимірює частоту розбиття на рівні вузла

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \quad (1.3)$$

Наприклад, необхідно обчислити чистоту наступного вектору:

S = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1]

$$n_0 = 7$$

$$n_1 = 3$$

Обрахунок формули має наступний вигляд (формула 1.4.):

$$E(S) = -0{,}7\log_2(0{,}7) - 0{,}3\log_2(0{,}3) = 0{,}8813 \quad (1.4.)$$

Для більш детального ознайомлення необхідно провести схоже обчислення з використанням VS Code (рис. 1.5).

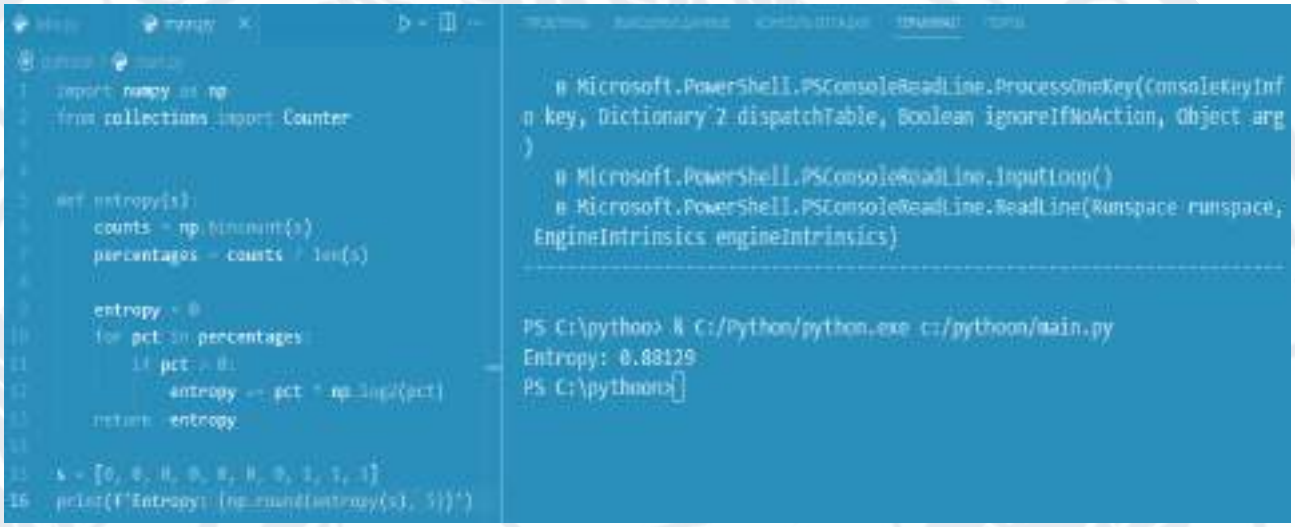


Рис. 1.5 – Приклад ентропії

Проведемо огляд отримання інформації, суть його полягає у тому, аби представити середнє значення ентропії на основі конкретного розподілу [18]. Значення повинно бути вище, значення інформації, у такому разі розподіл рішень буде кращим. Розглянемо подібний приклад з обрахованою ентропією (табл.я 1.2).

Таблиця 1.2 – Приклад обрахунку приросту інформації

| **Дані 1** | **Дані 2** | **Дані 3** |
|---|---|---|
| S = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1] $n_0 = 12$ $n_1 = 8$ | S = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1] $n_0 = 10$ $n_1 = 2$ | S = [0, 0, 0, 0, 1, 1, 1, 1] $n_0 = 4$ $n_1 = 4$ |
| $E(S) = 0.97095$ | $E(S) = 0.65002$ | $E(S) = 1$ |

Оскільки значення ентропії є відомим, можливим є обрахунок приросту інформації за наступною формулою:

$$Gain(S, A) = E\ (S) - \sum_{u \in Values(A)} \frac{|S_u|}{|S|} E(S_u) \quad (1.5.)$$

$$Gain(S, A) = \ 0.97095 - \frac{12}{20} \cdot 0.65002 - \frac{8}{20} \cdot 1 = 0.18094 \quad (1.6.)$$

Змінна з найменшою ентропією виступає у ролі кореневого вузла, який становить основу дерева рішень, тобто, це функція, яка займається поділом даних на частини. Разом із кореневим вузлом, дерево рішень має вузли прийняття рішень та листові вузли. Необхідно наголосити на тому, що не лише математичні обрахунки становлять основу алгоритму. Є багато інших складних та цікавих методів, що можуть слугувати об'єктом для аналізу.

І варто перейти до огляду нейронних мереж, які належать до розряду математичних моделей, проте будову має подібну до людського мозку – складається з нейронів – вузлів, які між собою групуються у певні шари [19]. Вони пов'язані синапсами, теж окремими з'єднаннями. Проте людським мозок є геніальним витвором, оскільки він працює, забезпечуючи людині не лише нормальне життя, а й відповідає за рухову здатність, здатність мислити та аналізувати, а також самонавчатися, як це робить машинне навчання. Тому при створенні нейронних мереж науковці у першу чергу дбали про те, як наблизитися до будови людського мозку і створити унікальну систему, яка спростить життя мільйонам людей.

Це геніальне відкриття вченим здійснити вдалося. Нейронні мережі працюють таким чином, що дані, які взаємодію із нейронною мережею, проходять багато шарів та постійно видозмінюються. При такому підході дані, які мають застарілу інформацію, можуть бути відсіяними, а ті, які мають вагоме значення, – передані далі для обробки.

Один нейрон має власну вагу, яка не є статичною, а динамічно змінюється у процесі навчання мережі. Якщо вага нейрону дуже велика, тоді зв'язок між нейронами стає більш міцним. Зрозуміти, як приблизно працює нейронна мережа, можна за допомогою практичного розгляду прикладу. Якщо виникає

необхідність визначити стать особи або груп осіб, можна скористатися деяким алгоритмом. Наприклад, згідно умови маємо відомості по зріст та вагу осіб, Сашко позначений як чоловік – одиницею, Аліса та Жанна як жінки – двійкою (рис. 1.6).

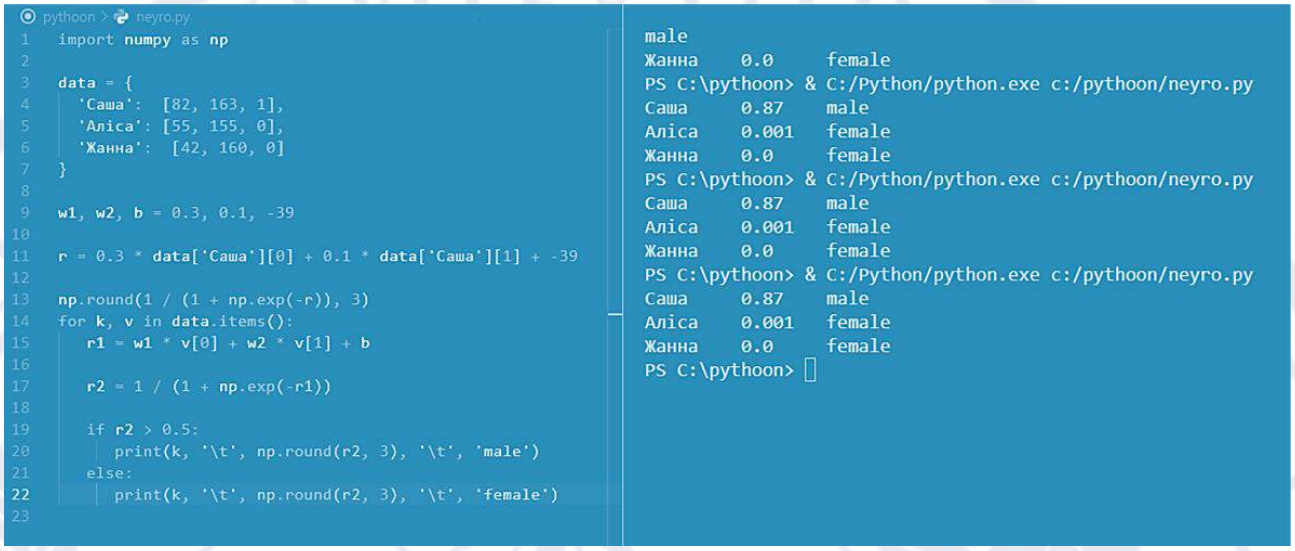


Рис. 1.6 – Схематичний приклад роботи мережі

Щодо визначення статі, результат, отриманий про Сашка, наближений до одиниці, отже, він – чоловік. Результати Аліси та Жанни максимально близькі до нуля, тому вони – жінки.

Приблизно за таким принципом працюють нейронні мережі та їх допомогу можна використовувати у нескладних або навіть складних випадках будь-якої сфери життєдіяльності людини. Вони допомагають у прийнятті рішень, розпізнаванні інформації, також застосовуються активно на виробництві, наприклад, якщо необхідно визначити, чи є партія певного товару гарної або поганої якості. Зібравши дані, систематизувавши їх та надавши їх нейронній мережі – вона зможе допомогти. Загалом використання машинного навчання на сьогодні можливе у різних сферах життєдіяльності людини (рис. 1.7).


Рис. 1.7 – Сфери застосування машинного навчання [20]

Проте необхідно не забувати про те, як важливо тримати баланс між перевагами та недоліками, присутніми у використанні цього виду навчання у тому числі (рис. 1.8, 1.9).

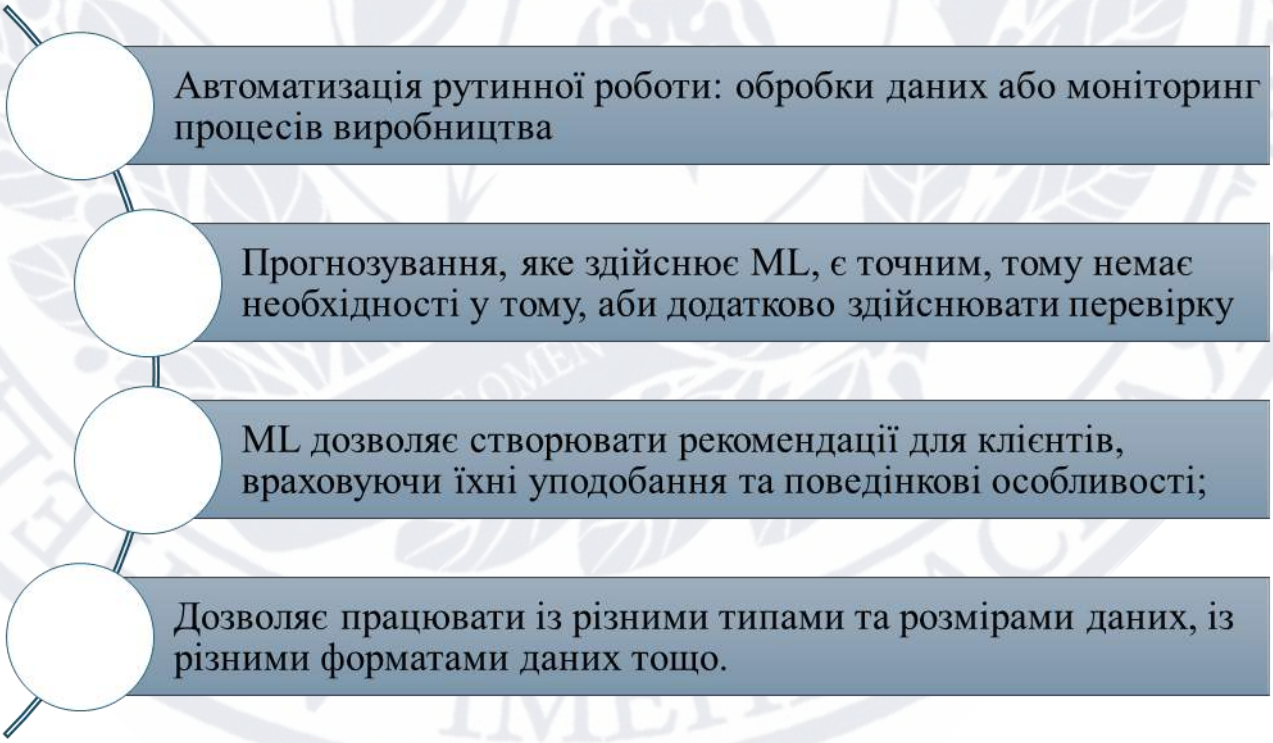


Рис. 1.8 – Переваги машинного навчання

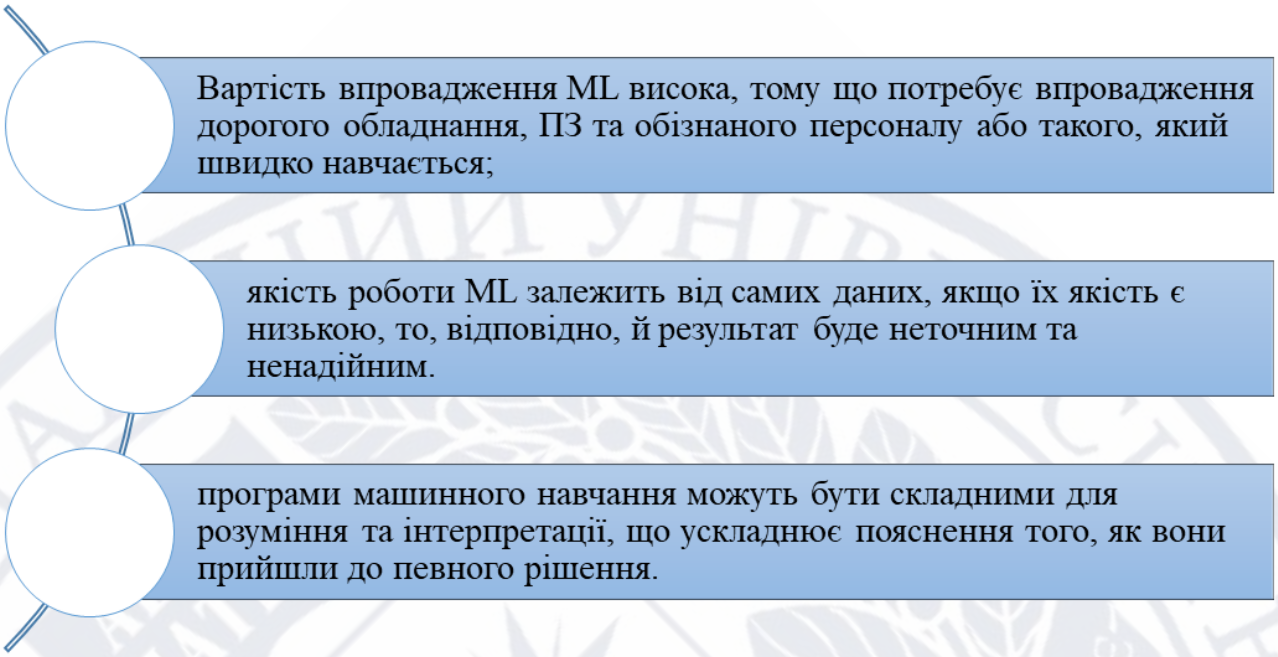


Рис. 1.9 – Недоліки машинного навчання [21]

Отже, підсумовуючи, потрібно наголосити на тому, що машинне навчання покликане покращити та пришвидшити роботу будь-якого підприємства, відповідно, зростатимуть й доходи. Зрозумілим є те, що для потрібно витрачати кошти на впровадження, але це буде результативне рішення. Якщо мова йде про обробку зображень, то машинне навчання значно зекономить час та підвищить ефективність обробки, відповідно, збільшить кількість клієнтів, які купують світлини, як товар.

## 1.2. Характеристика світлин: методи та способи опрацювання

Саме поняття «зображення» або інший його відповідник – «світлина» варто розуміти як візуальне представлення інформації у цифровому форматі. Із цифровими зображеннями працюють за допомогою використання ПК та інших пристроїв. Проте перед тим, як людство змогло дійти до оцифровування фото та інформації загалом, світлини мали виключно матеріальний (фізичний або друкований) характер та створювалися за допомогою габаритних фотоапаратів [22]. Перші фото оброблялися вручну, тому проблему виправлення дефектів тогочасні фотографи вирішували також вручну із застосуванням інструментів, спеціальної рідини та налаштованого освітлення.

Освітлення й на сьогодні також відіграє важливу роль як при створенні самої цифрової світлини, так й при обробці її із застосуванням вбудованих функцій обладнання, так і з використанням машинного навчання. Якість будь-якої світлини залежить від наступних трьох правил – майстерність фотографа, освітлення в приміщенні, а також майстерність самої вторинної обробки. Кожен із аспектів є незамінний, оскільки при ньому застосовується багато людських зусиль, які цінують й на сьогодні також. Створити фото вручну – це праця, яка потребує додаткових знань, постійної практики й самонавчання.

Створити цифрове зображення можна за допомогою спеціальних програм із застосуванням обладнання для фото-зйомки. У ролі пристрою для створення зображень найзручніше використовувати графічний планшет або будь-яке інше обладнання, яке допоможе з нуля створити шедевр.

Усі цифрові світлини містяться на електронних носіях. Цей вид інформації має певну специфіку, тому графічні файли від інших видів файлів за своїм форматом. Формат файлу – спосіб організації інформації на електронних носіях у файлі. Інформація у файлі повинна бути певним способом організована, щоб програми обробки могли правильно інтерпретувати дані, що містяться у файлі, і коректно будувати закодоване в ньому зображення.

Називати формати прийнято відповідно до призначення за промовчанням розширенням імені файлу (тип файлу). Наприклад, файли формату JPEG мають розширення .jpg, формату TIFF – розширення .tif.

Як правило, графічний файл будь-якого формату має дві основні частини: заголовок та тіло. У заголовку файлу розміщується вся службова інформація, необхідна для правильного розуміння вмісту тіла файлу, а тіло містить закодовані пікселі зображення.

Виділяють такі типи графічних файлів – растрові та векторні (рис. 1.10).

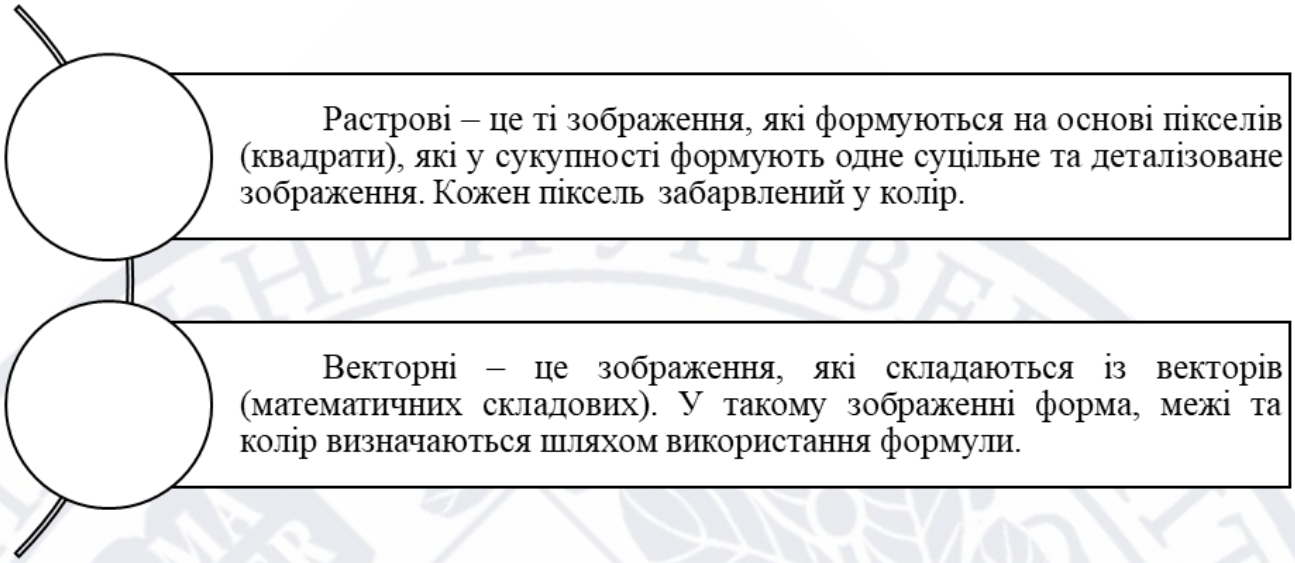


Рис. 1.10 – Типи графічних зображень [23]

Визначивши два основних напрями зображень є необхідність перейти до форматів (розширень) світлин, їх є досить велика кількість, об’єднаних спільними характеристика, проте різних.

Перший формат варто йменувати як JPEG [24]. Це найбільш використовуваний формат у навчанні, фотозйомці, підготовці презентаційних матеріалів тощо. Його також активно використовують для веб-зображень. У перекладі цей формат йменують як такий, що «об’єднаний в єдину групу з фотографіями». Такий формат світлини як правило піддають стисненню, тому важко знайти зображення у повноцінному обсязі. Проте у деяких випадках, особливо, коли обсяг пам’яті обмежений, можна застосовувати цей формат, проте варто пам’ятати, що при стисненні цього формату втрата відбувається інформації. Для окремих випадків це не є недоліком, але вартує уваги при виборі формату світлини для різних приводів.

Другий формат – це GIF [25]. Належить до розряду динамічних зображень, проте залежно від різновиду зображення можуть бути й статичними. Це найбільш використовуваний формат у світі та є найбільш поширюваним в Інтернеті. Кольорова палітра цього формату є обмеженою, проте під час стискання він зберігає власну структуру без втрати якості. Єдине, що варто отримати після стиснення – це зменшений розмір. Використовують такий формат для передачі емоцій, для логотипів, мультфільмів тощо.

Третім форматом варто вважати PNG [26]. Найчастіше цей формат файлу зображення застосовують для рекламування продукції, оскільки вони є максимально якісними та не втрачають якість при завантаженні або стисненні. Локалізовані переважно на веб-сторінках, соціальних мережах, лендінгах, різних форматах блогів тощо. Цей формат вдало підтримує ефект «прозорості».

Четвертим прийнято вважати формат TIFF [27]. Він не є поширеним у буденному або побутовому середовищі, оскільки у порівнянні з попередніми форматами цей формат має дуже великі розміри та високу роздільну здатність. Ці файли мають розширення .tif. Незважаючи на великий розмір, такий файл є активно використовуваним у таких сферах, як графічний дизайн, обробка фото та навіть у видавничій діяльності. Їх значною перевагою є те, що вони можуть слугувати місцем для зберігання інших типів файлів, схоже до теки або хмари для зберігання.

П’ятим форматом є BMP [28]. Цей формат також, як і формат GIF, не має широкого вибору колірної палітри, а також не піддаються процедурі стиснення, тобто, апріорі займають досить багато місця. Як вид растрового формату, BMP підходить для зображення у соціальних мережах та веб-сайтах, оскільки завжди є чіткими та гарно сприймаються людським оком.

Шостий формат має назву EPS [29]. Цей формат належить до розряду «векторні», підходить у тому випадку, якщо виникає потреба відтворити його у кількох розмірах.

Сьомий формат – це SVG [30]. Це світлини, які мають двовимірний формат. Тобто, при наведенні курсору на зображення, його рухливість збільшиться, проте роздільна здатність залишиться без змін. Часто цей формат зображень застосовують для персональних сайтів, які потребують просуванню, оскільки SVG написаний мовою, яка зберігає текстову інформацію. Пошукова система краще здійснює пошук інформації у текстовому вигляді, тому зображення і сторінка разом із ними буде потрапляти на очі користувачам частіше, що, власне, підніматиме рейтинг її привабливості та використання.

Восьмим виступає формат світлини такий як PDF [31]. Цей формат підходить більше для документів, застосовується у тому випадку, якщо є необхідність уберегти файл від змін. Додатково можливо удосконалити файл шляхом переведення у формат PDF/A, проте, як правило, у цьому немає необхідності. На усіх пристроях (планшет, смартфон, ПК) цей документ виглядатиме абсолютно однаково. Може відкриватися будь-якою програмою. Завдяки тому, що PDF-файли зберігають форматування та макет, їх використовують для друку, оскільки вони не викривлюють інформацію та вона залишається незмінною.

Дев'ятий формат – це AI [32]. Це формат компанії Adobe, який призначений для векторних світлин. Переважно цей формат застосовують для логотипів, банерів, рекламних постерів.

Десятий формат становить PSD [33]. Цей формат є продуктом іншої компанії – Photoshop. Файл PSD містить світлини, поділені на шари. Кожен шар у файлі накладається один на одного, що дозволяє користувачам створювати та редагувати зображення з кількома рівнями прозорості.

Одинадцятий формат – це RAW [34]. Має невідредаговане та не стиснене подання. Мають дані світлин, які зібрано певним сенсором, і це робить їх корисним та популярним форматом для обробки. Ці файли редагують в такій програмі, як Photoshop для налаштування експозиції, балансу кольорів тощо.

Дванадцятий формат – це WebP [35]. Належить до тих форматів файлів, які стискають світлини без втрат. Стискання без втрат має на меті перетворення вихідного зображення зі збереження якості навіть при зменшенні розміру.

Тринадцятий формат HDR [36] – це формат файлів, який дозволяє більш деталізовано отримати враження про глибину кольору, ніж традиційні файли зображень. Його особливістю є те, що він у змозі підтримувати якість зображень як статичних, так і динамічних, а також легко виправляти наявні дефекти у них. Світлини у такому форматі більш інформативні у відблисках та тінях, а також більш чіткі у кольорі в цілому.

Чотирнадцятим форматом є INDD [37] – це продукт, розроблений для користувачів Adobe, в якому є змога зберігати частини певного проекту: вміст сторінки, стилі, зразки тощо. З усім форматів цей має найбільш знижену політику безпеки та конфіденційності, локально немає змоги переглядати та редагувати такі зображення, також він не підходить для публікування на веб-сайтах. Проте, не зважаючи на те, що це більш текстовий формат, його застосовують і для створення електронних публікацій, презентацій.

XCF – це тип файлу світлин, який використовується лише однією програмою GIMP і в межах однієї організації [38]. Сама програма є збірником відредагованих зображень. Проте GIMP може працювати лише зі стисненими зображеннями.

Отже, така велика кількість різних форматів вказує на те, що зображення різної кольорової гами, якості, розміру тощо – завжди будуть цікавими й актуальними для дослідників. Постійно з’являються нові формати, а перелік їх поповнюється. Окремі компанії створюють власні формати для світлин, а деякі навіть виконують роль накопичувачів фото.

Тому, розглянувши основні різновиди зображень, варто перейти до особливостей використання обробки зображень, які на сьогодні існують, здійснимо огляд найбільш популярних аспектів, методів та сучасних технологій у обробці, їх актуальність, перспективи до подальшого розвитку.

В інформаційному середовищі цифрові зображення відіграють особливу роль, оскільки з їх допомогою будь-який контент, розміщений на сайтах, стає більш розбавленим, краще сприймається інформація, відбувається унаочнення будь-якої текстової інформації за допомогою графічного подання – і це важливий аспект.

Оскільки більшу частину інформації людина схильна забувати, увага від сприйняття та розуміння великої текстової інформації знижується, тоді на допомогу приходять фото, картинки, графіки, які унаочнюють та спрощують сприйняття інформації.

Обробка зображень постає у головній ролі під час дослідження покращення зображень і працює шляхом налаштування різних параметрів зображення та додавання функцій до зображення. Відомо, що обробка зображень є підмножиною комп'ютерного зору. Під час обробки зображень трансформації застосовуються до вхідного зображення, яке надає користувач, а потім до нього застосовується обробка зображення, щоб отримати вихідне зображення.

Така обробка зображень бере зображення як вхідні дані та застосовує до нього розпізнавання шаблонів, щоб зрозуміти шаблон для вхідного зображення, а потім надсилає його до конкретної моделі даних і геометрії для вдосконалення, звідси воно повертається до обробки зображення як вихідне зображення.

Розглянемо схему обробки зображень (рис. 1.11):


Рис. 1.11 – Принцип обробки зображень

На сьогодні існують такі методики обробки зображень – це компресія, фільтрація шумів, комп'ютерна графіка, розпізнавання образів, цифрова обробка тощо. Кожен з них має власні особливості.

Загальне представлення світлин в комп'ютері нагадує вектор пікселів. Кожен піксель кодується певною кількістю бітів, які визначають інтенсивність кольору: у випадку чорно-білого зображення – в градаціях сірого, а для кольорового – через три канали RGB.

Стиснення зображень необхідне для зменшення їх розміру, що дозволяє економити час при передачі фотографій та значно знижує вимоги до місця для зберігання [39].

Щоб зрозуміти важливість стиснення, слід розглянути роздільну здатність. Вона показує кількість пікселів, що припадає на кожен дюйм зображення під час його друку, і вимірюється в пікселях на дюйм.

Розглянемо приклад зображення, яке має роздільну здатність 1000:1000, кожен піксель – 8 біт. Необхідна кількість бітів становить:

$$R = 1000 * 1000 * 8 = 8000000 - 1 \text{ біт / зображення} \quad (1.7.)$$

Тепер для того, аби визначити, скільки необхідно бітів, аби зберегти, наприклад, 5-секундне відео, кількість кадрів на 1 секунду якого становить 50 кадрів, нам необхідно визначити загальну кількість бітів для цього відео.

$$R = 5 * 50 * 8000000 = 2000000000 - \text{біти} \quad (1.8.)$$

Отже, для того, аби зберегти 5-секундне відео, необхідно багато бітів. Тому необхідно знайти спосіб, який би допоміг зберігати зображення без втрати його якості. Тому в такому випадку застосовують стиснення.

Поетапне проведення стиснення (рис. 1.12):

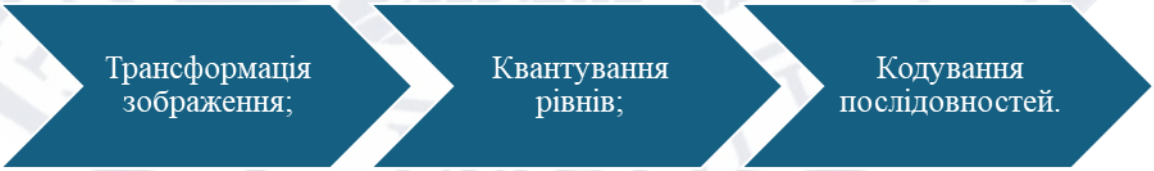


Рис. 1.12 – Поетапне проведення стиснення

З математичної перспективи, трансформація зображення є функцією, яка відображає один векторний простір в інший [40]. Якщо мова йде про зображення, то воно також може виступати у ролі функції із розташуванням пікселів, $I(x, y)$, x та y – це координати пікселя на зображенні. Перетворення зображень відбувається із просторової області в частотну.

Перетворення зображень здійснюється з просторової області в частотну. Цей процес є важливим, оскільки дозволяє легше виявити основні складові зображення, сприяє стисненню даних і спрощує обчислення.

Далі йде квантування, оскільки при його дії рівні інтенсивності згруповуються в окреме об'єднання, спираючись на математичні основи, що визначає пікселі. Зазвичай новий рівень визначається за допомогою фіксованого розміру фільтра «m», шляхом ділення кожного з елементів фільтра на «m», округлення й додаткове множення на «m».

Представлена функція:

$$\left[\frac{\text{pixelvalue}}{m}\right] \cdot m \qquad (1.9)$$

Тому, кількість задіяних рівнів у світлинах активно зменшується, відповідно, разом із цим знижується надмірність інтенсивності. У такому випадку без квантування неможливо, бо саме воно допомагає зменшити ці рівні.

Розглянемо приклад квантування, m = 6 (рис. 1.13).

| | | |
|---|---|---|
| 8 | 5 | 12 |
| 25 | 32 | 31 |
| 2 | 15 | 11 |

[ 25 / 6 ] · 6 = 18

[ 18 / 6 ] · 6 = 18

Рис. 1.13 – Квантування

Отже, ці представлені значення округлили до 18, тому було зменшено кількість рівнів (залучених символів) у специфікації зображення.

Наступний етап – створення символу, іншими словами – перетворення у код, він проводиться над групою символів світлин і не лише світлин. Визначити, скільки потрібно бітів, можна шляхом розуміння того, як часто з'являється символ.

Тому, було розглянуто особливості методу компресії зображень та його складники. Варто зазначити, що компресія застосується також у багатьох сферах життєдіяльності людини (рис. 1.14).

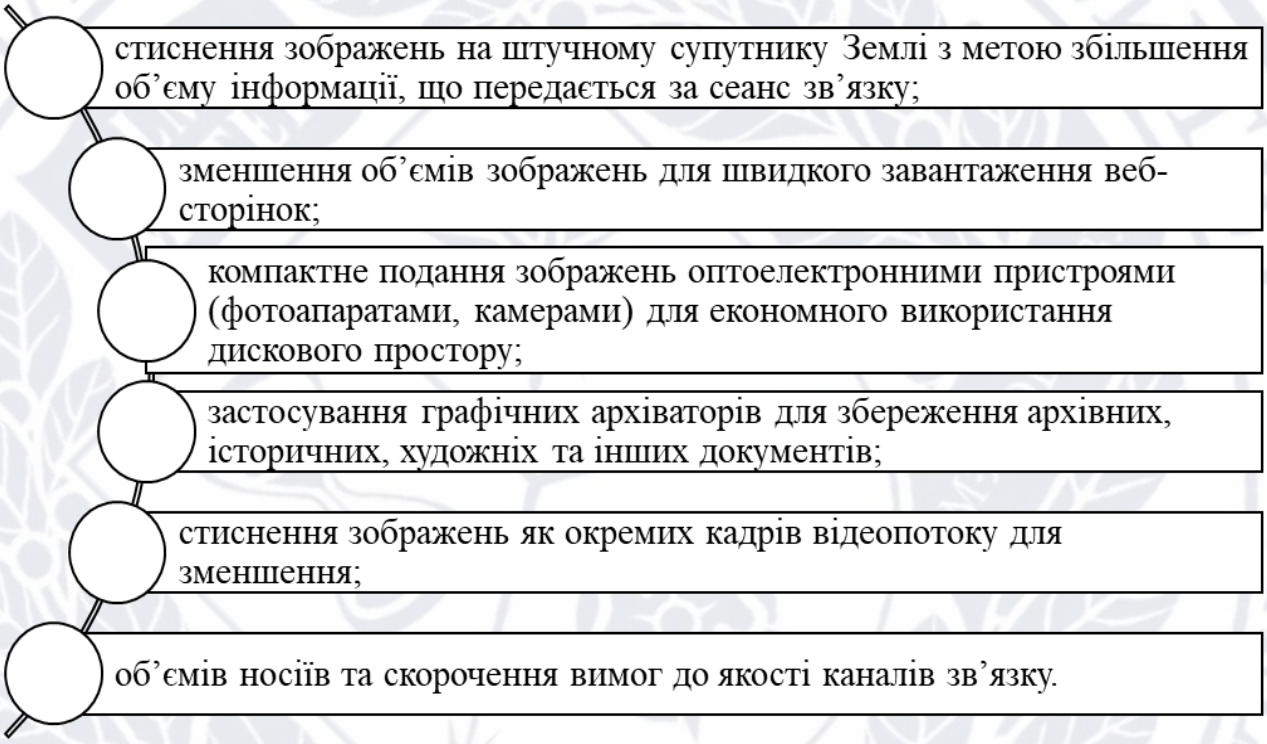


Рис. 1.14 – Сфери застосування компресії [41]

**Фільтрація шумів**. Алгоритми фільтрації зображень генерують вихідні пікселі, спостерігаючи за околицями певного вхідного пікселя на зображенні. Якщо операція є лінійною, значення вихідного пікселя обчислюється лінійним об'єднанням значень набору пікселів поблизу відповідного вхідного пікселя на основі їх відносних позицій відповідно до правил алгоритму. Цей процес можна використовувати для покращення або зменшення певних функцій зображення, зберігаючи інші.

Алгоритми фільтрації зображень формують значення вихідних пікселів, аналізуючи сусідні пікселі конкретного вхідного пікселя на зображенні. У випадку лінійних операцій, значення вихідного пікселя визначається як лінійна

комбінація значень сусідніх пікселів, з урахуванням їхніх відносних позицій відповідно до алгоритмічних правил. Цей підхід можна використовувати для покращення або зменшення певних характеристик зображення, при цьому зберігаючи інші.

Обробка зображень пов'язана з використанням матриць. У випадку зображення в градаціях сірого, де кожен піксель відображає інтенсивність світла в певній точці, матриця має двовимірну структуру. Ядро – це матриця з заздалегідь визначеними значеннями для кожного індексу. Результат згортки – це перетворене зображення вихідного зображення.

Цифрові зображення часто містять шум, який призводить до спотворення значень пікселів. Фільтрація зображень – це процес, спрямований на усунення цих спотворень. Згортка зашумленого зображення з відповідним ядром ефективно усуває шум.

Фільтр Гауса забезпечує підвищений контроль і точніші результати, але може виникати похибка при обчисленні зваженої суми пікселів на краях зображення. Він згладжує пікселі на краях, що призводить до зменшення варіацій інтенсивності. Фільтр Гауса надає вагу сусіднім пікселям виключно залежно від їх відстані до поточного пікселя. Форма гаусіанської функції визначається лише стандартним відхиленням. Рівняння Гауса має такий вигляд:

$$F(x) = ae^{\frac{(x-b)^2}{2x^2}} \tag{1.10}$$

Комп'ютерна графіка. Комп'ютерна графіка – це процес генерації зображень на комп'ютерах, які потім використовуються на різних носіях. Зображення, що використовуються в графічному дизайні друкованих матеріалів, зазвичай створюються за допомогою комп'ютерних технологій, так само як статичні та анімовані зображення в коміксах і анімації. Реалістичні зображення, які можна переглядати та редагувати в електронних іграх і комп'ютерному моделюванні, не можуть бути створені або підтримані без потужних можливостей сучасної комп'ютерної графіки. Ця технологія також є важливою для наукової візуалізації – сфери, що використовує зображення та кольори для моделювання складних явищ, таких як повітряні потоки та електричні поля, а

також для автоматизованого проектування та дизайну, де об’єкти створюються та аналізуються в комп’ютерних програмах.

Таблиця 1.2 – Порівняння комп’ютерної графіки та обробки зображень

| Комп’ютерна графіка | Обробка зображень |
|---|---|
| Зосереджений на створенні та обробці зображень для візуального виведення. | Обробка зображень зосереджена на аналізі та покращенні якості та допомагає отримати інформацію або покращити якість. |
| Основною метою є створення візуального вмісту, зрозумілого людям, часто з інформаційною метою. | Основною метою є зміна або витяг інформації з цифрових зображень, як правило, за допомогою алгоритмів і методів. |
| Комп’ютерна графіка використовується у фільмах, відеоіграх та авіасимуляторах для графічних цілей | Обробка зображень використовується в медичних зображеннях, супутникових зображеннях, розпізнаванні обличчя, редагуванні тощо для кращого зображення. |
| Комп’ютерна графіка створює нові зображення з нуля або змінює існуючі, щоб передати певне повідомлення. | Обробка зображень змінює або вдосконалює існуючі зображення, переважно без фундаментальної зміни їхнього вмісту. |
| Деякими прикладами комп’ютерної графіки є фільми Pixar, Adobe Photoshop і графічні движки для відеоігор. | Деякими прикладами обробки зображень є програмне забезпечення для аналізу медичних зображень , фільтри Instagram і системи розпізнавання облич. |
| Часто починається з геометричних описів або моделей об’єктів тощо. | Він починається з цифрових зображень, отриманих із різних джерел, таких як камери, сканери чи супутники. |
| Для комп’ютерної графіки часто потрібні потужні графічні процесори (GPU) і обладнання для візуалізації. | Він може бути реалізований на стандартному обчислювальному обладнанні, але може потребувати паралельних процесорів для ефективності. |
| Може створювати дуже складні візуальні сцени з кращими деталями та динамічними ефектами, такими як освітлення та тіні. | Результати часто простіші, зосереджені на покращенні або аналізі конкретних аспектів зображення, таких як контраст чи текстура. |

Комп’ютерна графіка застосовується в комп’ютерах для обробки, створення та зберігання різноманітних об’єктів і зображень. Ось кілька її основних застосувань: за допомогою комп’ютерної графіки можна створювати різні види мистецтва, включаючи комерційне та образотворче, а також анімаційні проекти. Вона також дозволяє створювати візуалізації для різних звітів, таких як фінансові, статистичні та економічні дані.

Крім того, комп’ютерна графіка активно використовується в індустрії розваг, зокрема в іграх та кінематографі. Завдяки їй можна розробляти креслення для будівель, а також проекти літаків і автомобілів.

Обробка зображень також застосовується в комп’ютерах для маніпуляцій з різними характеристиками зображень. Нижче наведено кілька програм, що спеціалізуються на обробці зображень. Ця технологія дозволяє підвищити різкість зображень і відновити втрачені деталі.

У медичній сфері обробка зображень використовується для виявлення шаблонів у медичних звітах та покращення якості зображень, що провокує отримання якісних результатів. Крім того, вона застосовується в передачі та кодуванні даних, що забезпечує швидшу та безперебійну передачу зображень.

**Розпізнавання образів.** Система розпізнавання образів повинна швидко та точно виявляти знайомі шаблони, а також розпізнавати і класифікувати нові об’єкти. Вона повинна забезпечувати точне розпізнавання форм і предметів з різних ракурсів, а також виявляти візерунки та об’єкти, навіть якщо вони частково закриті. Крім того, система повинна оперативно ідентифікувати візерунки.

Серед завдань, які вирішує розпізнавання образів, можна виділити:

1) класифікацію;
2) виявлення підроблених біометричних даних;
3) розпізнавання візерунків на тканині для людей з вадами зору;
4) можливість ідентифікації окремих об’єктів з різних кутів.

Проте, розпізнавання образів має й свої недоліки:

1) реалізація синтаксичного розпізнавання є складною і потребує багато часу;

2) для досягнення вищої точності може знадобитися більший обсяг даних;

3) система не може пояснити, чому певний об’єкт був ідентифікований.

Сфери застосування технології розпізнавання образів є досить різноманітними. Ця технологія дозволяє виділяти основні характеристики з наданих зразків зображень або відео. Вона активно використовується в комп’ютерному зорі для виконання різних завдань, таких як аналіз біологічних і медичних зображень. Крім того, розпізнавання образів застосовується для виявлення, створення зображень та інтерпретації часових шаблонів у сейсмічних даних, а також у програмах класифікації радіолокаційних сигналів, наприклад, для виявлення та ідентифікації мін. Технологія також широко використовується в розпізнаванні мовлення, де різні алгоритми намагаються подолати труднощі, пов’язані з використанням фонем, і розглядають більші одиниці, такі як слова, як шаблони. У сфері розпізнавання відбитків пальців застосовуються різноманітні методи, включаючи розпізнавання образів.

Отже, було проведено аналіз характеристик цифрових зображень та існуючих методів їх обробки. Варто зазначити, що обробка необхідна та без неї на сьогодні неможливо обійтися, адже з її допомогою було зменшено кількість злочинності, було спрощено багато організаційних та управлінських процесів, пришвидшилося навчання тощо.

### 1.3. Обробка зображень: практичний складник

Почнемо із визначення поняття «зображення». Звична для сприйняття картинка або світлина – це відтворення бажаної деталі, предмета, людини шляхом застосування світла, приладів здійснення фото та відтворення, а також впливу реагентів для відтворення.

Із одним зображення можна зробити багато цікавих процедур, тобто, обробити фото, зробити його більш розмитим, збільшити або зменшити розмір, обрізати, накласти спеціальні ефекти, видалити із картинки певні об’єкти тощо.

На сьогодні існують вбудовані редактори фото, які без додаткових зусиль допомагають змінювати форму подання зображення, проте існують й способи виконання цього завдання за допомогою програмування, використання вбудованих бібліотек, алгоритмів, функцій.

Необхідно побудувати таким чином код, аби можна було виконувати як більш легкі операції, так і більш складні. Для проведення практичного експерименту візьмемо зображення троянди та спробуємо змінити його зовнішній вигляд (рис. 1.15).

Рис. 1.15 – Зображення троянди

Далі необхідно обрати середовище для роботи із кодом, зупинимося на PyCharm. PyCharm є однією з ключових платформ для розробки програмного забезпечення. PyCharm є продуктом такої мови як Python, він необхідний для розробки програм.

Далі необхідно визначити, які саме дії будуть виконуватися над зображенням – це буде перетворення у ч/б формат та перетворення у ефект розмиття або іншими словами – ефект мультиплікації.

Також необхідно обрати бібліотеки, які підійдуть для цього випадку. Це будуть CV2, Matplotlib та PIL. Зупинимося більш детально на кожній з них.

«cv2» є однією з бібліотек. Створена для реалізації різних можливостей, спрощення інтегрування та допомоги обладнанню краще сприймати візуальну інформацію так, як це робить людина.

Ще однією бібліотекою виступає Matplotlib, вона потрібна для візуалізації даних, що дозволяє створювати як двовимірні, так і тривимірні графіки.

Бібліотека Python для обробки зображень, відома як PIL, є важливою для роботи з графікою в Python. Її покращена версія, Pillow, надає інструменти для маніпуляцій та обробки зображень, подібні до тих, що використовуються в програмному забезпеченні, такому як Photoshop. Деякі з новітніх бібліотек для обробки зображень у Python базуються на Pillow і часто пропонують розширену функціональність. Подамо виконаний код у наступних зображеннях 1.16-1.21:

```
from tkinter import *
from tkinter import filedialog

import cv2
import matplotlib.pyplot as plt
from PIL import Image, ImageTk

window = Tk()
window.geometry('600x600+200+300')

class ImageEditor:
    path = ''
    img = None
```

Рис. 1.16 – Приклад коду 1

```
    def create_image(self):
        if self.path:
            self.img = cv2.imread(self.path)
            self.img = cv2.cvtColor(self.img, cv2.COLOR_BGR2RGB)


image_editor = ImageEditor()


def open_img():
    file = filedialog.askopenfilename()
    if file:
        image_editor.path = file
        image_editor.create_image()

        img = Image.open(file)
        image = ImageTk.PhotoImage(img.resize((700, 500),
Image.Resampling.NEAREST))

        label_image = Label(window, image=image)
        label_image.image = image
        label_image.place(x=0, y=300, anchor="w")
```

Рис. 1.17 –Код частина 2

```
def cartoon_func():
    if image_editor.img is None:
        print("Будь ласка, відкрийте зображення.")
        return

    gray = cv2.cvtColor(image_editor.img, cv2.COLOR_RGB2GRAY)
    gray = cv2.medianBlur(gray, 5)
    edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 9, 9)
    color = cv2.bilateralFilter(image_editor.img, 9, 250, 250)
    cartoon = cv2.bitwise_and(color, color, mask=edges)

    cartoon_resized = cv2.resize(cartoon, (700, 500),
interpolation=cv2.INTER_AREA)
    img_pil = Image.fromarray(cartoon_resized)
    image = ImageTk.PhotoImage(img_pil)

    label_image = Label(window, image=image)
    label_image.image = image
    label_image.place(x=0, y=300, anchor="w")
```

Рис. 1.18 – Код частина 3

```
    plt.figure(figsize=(10, 10))
    plt.imshow(cartoon)
    plt.axis("off")
    plt.title("Cartoon Image")
    plt.show()


def black_white_func():
    if image_editor.img is None:
        print("Будь ласка, відкрийте зображення.")
```

Рис. 1.19 – Код частина 4

```
return

gray = cv2.cvtColor(image_editor.img, cv2.COLOR_RGB2GRAY)
gray = cv2.medianBlur(gray, 5)

gray_resized = cv2.resize(gray, (700, 500), interpolation=cv2.INTER_AREA)
img_pil = Image.fromarray(gray_resized)
image = ImageTk.PhotoImage(img_pil)

label_image = Label(window, image=image)
label_image.image = image
label_image.place(x=0, y=300, anchor="w")


plt.figure(figsize=(10, 10))
plt.imshow(gray, cmap="gray")
plt.axis("off")
plt.title("Grayscale Image")
plt.show()
```

Рис. 1.20 – Код частина 5

```
button_open = Button(window, text='Відкрити картинку', command=open_img)
button_open.place(x=0, y=0)

button_bw = Button(window, text='Зробити картину в чб',
command=black_white_func)
button_bw.place(x=200, y=0)

button_cartoon = Button(window, text='Мультиплікаційний стиль картинки',
command=cartoon_func)
button_cartoon.place(x=400, y=0)

window.mainloop()
```

Рис. 1.21 – Код частина 6


Рис. 1.22 – Кінцевий результат

Отже, у результаті проведеного експерименту було розглянуто багато питань, які стосуються визначення особливостей самих зображень, способів їх обробки, сфер використання.

У результаті вдалося отримати із оригінального зображення троянди зображення у чорно-білому ефекті та розмите зображення. Розглянули досить багато методів обробки, які допомагають створити більш чітке уявлення про можливості обробки зображень. Також розкрили питання, що таке машинне навчання, розглянули детального його особливості.

# РОЗДІЛ 2
# АНАЛІЗ ПРОЦЕСУ ВИЯВЛЕННЯ ТА ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ

## 2.1. Критерії якості даних для забезпечення точної роботи алгоритму

Для навчання моделі машинного навчання, яка буде призначена для виявлення та відновлення пошкоджених зображень, потрібно запропонувати систему класифікації для оцінки зображень.

Класифікація ґрунтуватиметься на таких критеріях, як роздільна здатність, рівень шуму, різкість, динамічний діапазон, точність кольору, колірна гама, артефакти стиснення, експозиція, контраст, глибина різкості та індивідуальні вподобання. Хоча останній критерій має найменший вплив, він також важливий, оскільки для людей з особливими смаками це може мати значення.

Розглянемо кілька зображень з різними роздільними здатностями та спробуємо візуально оцінити їхні відмінності.

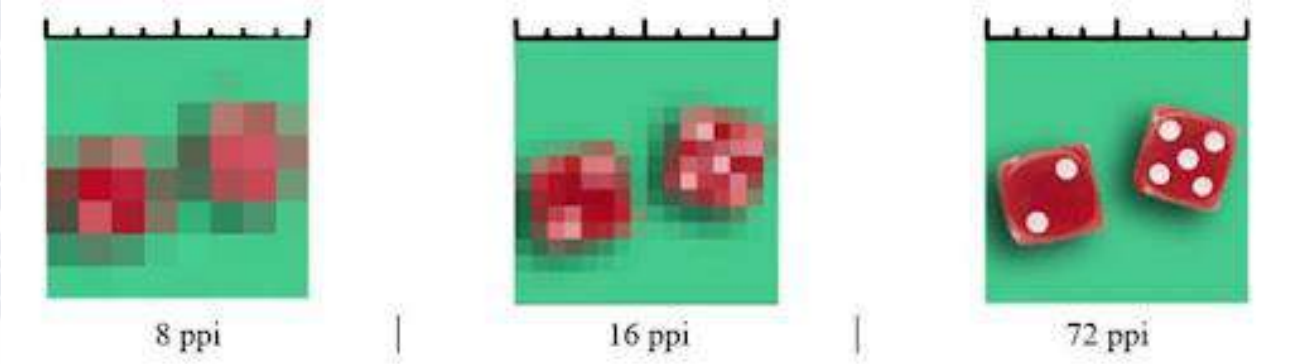


Рис. 2.1 – Порівняльна характеристика зображень за роздільною здатністю

Перше зображення має найнижчу роздільну здатність, має лише вісім ppi, відповідно, останнє зображення має найбільш оптимальну, проте теж не наближену до ідеалу роздільну здатність.

Вимоги до зображень досить суворі, оскільки їх якість впливає на оптимізацію сайту, кількість відвідувань та повернень, що також є не менш важливим фактором. Це знижує привабливість сайту та знижує продажі. Тому необхідно відповідально ставитися до опрацювання фото та розуміти, яке з них краще підходить до конкретного випадку. Наприклад, якщо мова йде про сайти, роздільна здатність має починатися від 72 ppi і коливатися до 150 ppi, але, якщо

зображення повноекранне відкриття, тоді роздільна здатність має бути близько 1920x1080. Проте щодо сайту, варто враховувати також розмір фото, аби сторінка не була перенавантаженою.

Якщо мова йде про розміщення фото у соціальних мережах, тоді роздільна здатність має приблизно становити 1080x1080 або трохи більше. Якщо мова йде про зображення, зроблені на фотоапарат, то в такому випадку РЗ має становити орієнтовно 6000x4000. Найбільш підходящий варіант – це роздільна здатність 1080x1080.

Щодо *рівня шуму*, то це важливий показник якості, оскільки шум може суттєво погіршити чіткість та сприйняття деталей. В ідеалі, для того щоб зображення було якісним, рівень шуму повинен бути мінімальним, але все залежить від конкретних вимог і області застосування.

Для відео та зображень на екрані більш високий рівень шуму може бути прийнятним, особливо якщо зображення використовується в цифровому вигляді або для відео з низькою роздільною здатністю. Проте, щоб зображення було якісним, шум повинен бути мінімальним і майже непомітним для очей. Шуми бувають:

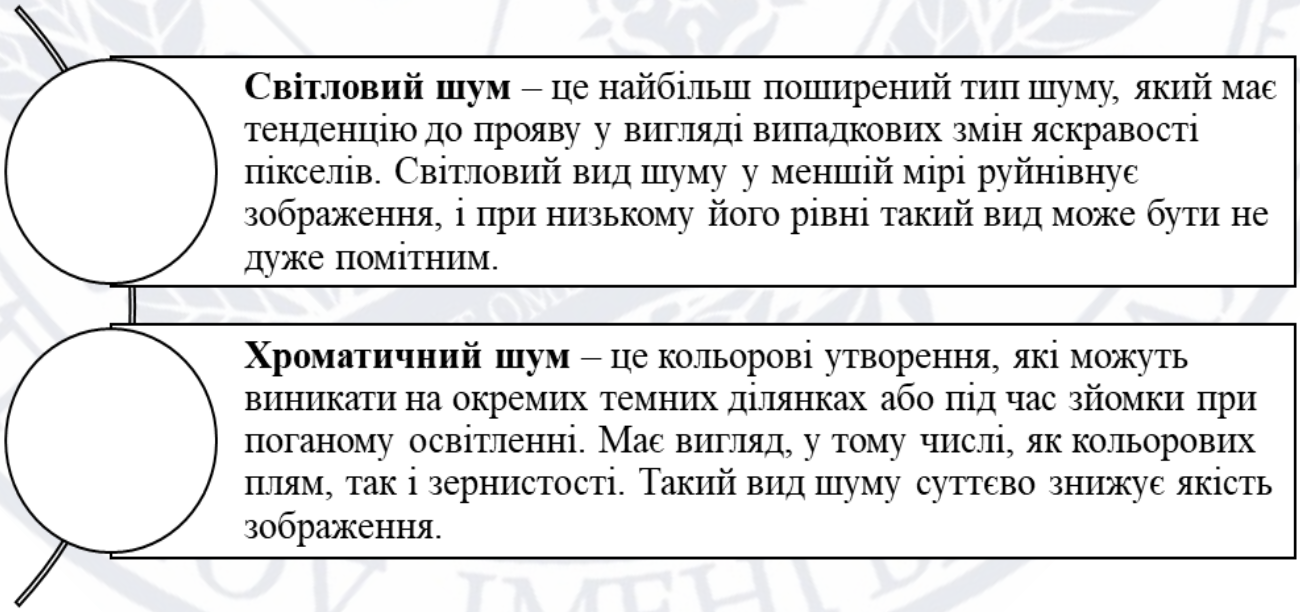


Рис. 2.2 – Різновиди шумів

Для веб-сайтів і соціальних мереж допустимий невеликий рівень шуму, особливо яскравого, оскільки на малих екранах і при перегляді з відстані він залишається непомітним. У такому випадку невеликий шум (до 2-3% за спеціальними показниками) не вплине на загальне сприйняття зображення.

Оптимальний рівень шуму – це той мінімум, який не заважає чіткості та передачі деталей. Для професійних фотографій і друкованих зображень шум має бути практично непомітним, тоді як для цифрових зображень невеликий шум може бути прийнятним, але бажано підтримувати його на якомога нижчому рівні.

Різкість зображення – це характеристика, що визначає чіткість деталей і контурів об’єктів. Якісне зображення повинно мати добре збалансовану різкість, щоб передавати максимальну кількість деталей без надмірного акцентування країв, що може призвести до спотворень.

Ідеальною різкістю вважається така, коли всі об’єкти на зображенні виглядають чітко, краї добре окреслені, але при цьому немає штучних контурів або шумів навколо них.

Рис. 2.3 – Зміна різкості

На правому зображенні представлено картину з ідеальною різкістю, тоді як на лівому – з низькою. Текстури та деталі квітки справа добре видно, без надмірного акцентування контурів. Зображення зліва виглядає занадто "м’яким" і розмитим, що призводить до втрати деталізації та загальної чіткості, негативно впливаючи на його якість.

Фактори, що впливають на різкість зображень:

1) різкість можна покращити за допомогою графічних редакторів, таких як Adobe Photoshop. Використання фільтрів для підвищення різкості дозволяє зробити краї більш чіткими.

2) Деякі камери мають можливість коригувати різкість під час зйомки завдяки вбудованим алгоритмам.

Проте для вебу та соціальних мереж невелика кількість різкості часто є прийнятною, оскільки зображення переглядаються на маленьких екранах і можуть бути стиснутими. Важливо, щоб ключові деталі залишалися чіткими.

Отже, для досягнення якісного зображення різкість повинна бути на достатньому рівні, щоб передавати деталі без спотворень. Вона має бути збалансованою, щоб об’єкти виглядали чітко, але не були перевантажені контрастом чи штучними контурами. Залежно від кінцевого призначення (веб, друк, цифрові носії) необхідний рівень різкості може варіюватися, але головним завжди залишається чіткість ключових елементів зображення.

Динамічний діапазон зображення визначає різницю між найтемнішими і найсвітлішими частинами, які можна відобразити без втрати деталей. Він вимірюється в ступенях експозиції або стопах. Чим більший діапазон, тим краще камера чи сенсор здатні захоплювати яскраві області без пересвічування (перенасичення білим) і темні області без затінення (повністю чорного кольору).

Для отримання якісного зображення важливо, щоб оптимальний динамічний діапазон забезпечував передачу максимальної кількості деталей як у тінях, так і в світлих ділянках. Для камер широкий динамічний діапазон зазвичай коливається від 10 до 14 EV. Професійні камери, такі як DSLR або бездзеркальні

моделі, можуть досягати 14-15 EV, що дозволяє зберігати високу деталізацію навіть у складних умовах освітлення.

Формати RAW, як правило, мають більший динамічний діапазон у порівнянні з JPEG. RAW-файли зберігають більше інформації як у тінях, так і в світлих зонах, що дозволяє коригувати експозицію під час постобробки без втрати деталей.

Види динамічного діапазону:

1) вузький динамічний діапазон (менше 8 EV): зображення виглядає плоским, деталі в тінях втрачаються, а світлі ділянки можуть бути пересвіченими.

2) широкий динамічний діапазон (10-14 EV): дозволяє зберігати більше деталей у всіх частинах зображення, навіть у сценах з високим контрастом між світлом і тінню.

3) динамічний діапазон для мобільних телефонів: багато сучасних смартфонів обладнані технологіями HDR, що забезпечує більший діапазон, хоча загалом він все ще поступається професійним камерам.

Для отримання якісного зображення динамічний діапазон повинен бути достатньо широким, щоб охоплювати деталі як у тінях, так і в світлих областях. Для більшості завдань оптимальним вважається діапазон 10-14 EV. Ширший діапазон дозволяє передати більше деталей у складних умовах освітлення та забезпечити високу якість зображень у фотографії та відео. Щоб досягти максимальної якості, рекомендується знімати у форматі RAW або використовувати HDR.

Точність кольору – це здатність зображення адекватно відображати реальні кольори без спотворень. Для забезпечення високої якості зображення важливо, щоб кольори максимально відповідали тим, що ми спостерігаємо в природі. Ця характеристика є особливо важливою в таких сферах, як фотографія, друк, кінематограф, медична візуалізація та дизайн, де кожен відтінок і кольорова гармонія мають велике значення.

Точність кольору визначається тим, наскільки коректно пристрій (камера, монітор, принтер) відтворює кольорову палітру без спотворення відтінків. Для оцінки точності кольору використовуються спеціальні метрики та тести, зокрема ΔE (Delta E), яка вимірює відхилення між відтвореним кольором і еталонним.

ΔE (Delta E) – це одиниця виміру різниці між двома кольорами:

- ΔE < 1 – людське око не помітить різниці.
- ΔE від 1 до 2 – різниця майже непомітна.
- ΔE 2-5 – різниця помітна, але може бути прийнятною в деяких випадках.
- ΔE > 5 – різниця кольорів стає очевидною, і це свідчить про низьку точність кольору.

Кольорова температура суттєво впливає на сприйняття кольорів у зображеннях. Для досягнення точного відтворення кольорів важливо, щоб кольорова температура джерела світла (освітлення) відповідала умовам, в яких було створено або відтворено зображення.

Основні кольорові простори:

- sRGB: Найбільш розповсюджений стандарт для більшості моніторів, веб-сайтів та мобільних пристроїв. Він охоплює обмежений діапазон кольорів, але підходить для повсякденного використання.
- Adobe RGB: Ширший колірний простір, що використовується в професійній фотографії та друці. Забезпечує більшу кількість відтінків зеленого та синього кольорів.
- ProPhoto RGB: Один з найширших колірних просторів, який застосовується в професійній обробці фотографій, особливо для друку з високою точністю кольорів.
- DCI-P3: Стандарт, що використовується в кіноіндустрії та для дисплеїв з підтримкою HDR, який охоплює ширший спектр кольорів у порівнянні з sRGB.

Для досягнення високої якості зображення важливо забезпечити максимальну точність кольору, особливо в професійних сферах, таких як фотографія, друк і дизайн.

Стандартом для професійної якості вважається значення $\Delta E < 2$. Використання розширених кольорових просторів (наприклад, Adobe RGB, ProPhoto RGB, DCI-P3) разом із правильним калібруванням моніторів і принтерів сприяє точній передачі кольорів. Також важливо враховувати кольорову температуру та умови освітлення, щоб кольори виглядали природно і відповідали реальності.

Артефакти стиснення – це спотворення або дефекти зображення, які виникають внаслідок використання алгоритмів стиснення з втратою якості (наприклад, JPEG). Вони можуть негативно впливати на якість зображення, роблячи його менш чітким і природним. Щоб зберегти високу якість зображення, артефакти стиснення повинні бути мінімальними або непомітними.

Типи артефактів:

1) блокові артефакти виглядають як помітні квадрати або блоки на зображенні. Вони виникають внаслідок стиснення зображення на окремі піксельні блоки, що призводить до спотворення оригінальних деталей через різкі переходи між ними.

2) Зниження різкості проявляється у втраті чіткості, особливо на краях об’єктів, що робить зображення менш деталізованим.

3) Кольорові артефакти виникають, коли кольори «перетікають» з однієї частини зображення в іншу, створюючи розмиті межі або невідповідність кольорів.

4) Гало-ефект характеризується світлими або темними обводками навколо контрастних об’єктів, що виникає через некоректну компресію.

5) Втрата градацій – це зникнення плавних переходів у кольорах, коли складні відтінки і градієнти спрощуються до різких переходів.

Експозиція – одним із основних чинників, що впливають на загальну якість зображення. Вона визначає кількість світла, яке потрапляє на сенсор камери під час зйомки, і впливає на те, наскільки яскравими або темними будуть зображення.

Контраст визначає різницю між найсвітлішими та найтемнішими частинами зображення. Правильне налаштування контрасту впливає на сприйняття зображення, його деталізацію та передачу настрою. Щоб зображення виглядало якісно, контраст повинен бути збалансованим, без надмірного підвищення або зниження, що забезпечує чіткість деталей і збереження природності кольорів.

Глибина різкості – це діапазон відстаней, в межах якого об’єкти на зображенні виглядають чіткими та різкими. Вона залежить від налаштувань камери і є важливим фактором у створенні якісного зображення, оскільки визначає, які елементи кадру будуть у фокусі, а які – розмиті. Глибина різкості впливає на загальний вигляд і атмосферу зображення.

Отже, проаналізувавши критерії оцінки якості зображень, можна визначити класифікаційні ознаки ідеального зображення, що дозволить у подальшому мати надійні дані для орієнтації.

Контраст визначає різницю між найсвітлішими та найтемнішими частинами зображення. Правильне налаштування контрасту впливає на сприйняття зображення, його деталізацію та передачу настрою.

Щоб зображення виглядало якісно, контраст повинен бути збалансованим, без надмірного підвищення або зниження, що забезпечує чіткість деталей і збереження природності кольорів.

Глибина різкості – це діапазон відстаней на зображенні, в межах якого об’єкти виглядають чіткими та різкими. Вона залежить від налаштувань камери і є важливим фактором у створенні якісного зображення, оскільки визначає, які елементи кадру будуть у фокусі, а які – розмиті. Глибина різкості впливає на загальний вигляд і атмосферу зображення.

Отже, проаналізувавши критерії оцінки якості зображень, можна визначити класифікаційні ознаки ідеального зображення, що дозволить у подальшому мати надійні дані для орієнтації (табл. 2.1).

Таблиця 2.1 – Класифікаційні ознаки ідеальних показників зображень

| Класифікаційна ознака | Ідеальні показники |
|---|---|
| роздільна здатність | 1080x1080 ppi |
| рівень шуму | Мінімальні |
| різкість | +30% |
| динамічний діапазон | 10-14 EV |
| точність кольору | ΔE від 1 до 2 |
| артефакти стиснення | При низькому рівні стиснення |
| експозиція | +30% |
| контраст | Ближче до високого |
| глибина різкості | і низьке, і високе оптимально |

### 2.2. Огляд інструментарію для відновлення зображень

Вибір мови програмування для завдання відновлення зображень визначається кількома факторами: ефективністю обчислень, доступністю бібліотек для обробки зображень і машинного навчання, а також особистим досвідом програміста.

Python є одним з найпопулярніших виборів для завдань, пов’язаних з обробкою зображень та машинним навчанням, завдяки великій кількості бібліотек та інструментів.

Серед бібліотек, доступних у Python, варто виділити:

- OpenCV: бібліотека для комп’ютерного зору, яка пропонує безліч функцій для обробки зображень, фільтрації, зменшення шуму, корекції кольорів, а також для відновлення пошкоджених зображень.
- Pillow: інструмент для базової обробки зображень, що включає зміни розміру, поворот, обрізку тощо.
- scikit-image: бібліотека для обробки зображень, що містить готові алгоритми для покращення, сегментації та відновлення зображень.

- TensorFlow і PyTorch: бібліотеки для машинного навчання, які можуть бути використані для створення нейронних мереж.

Переваги:

- Простота використання та низький поріг входження.
- Велика кількість готових алгоритмів і навчальних прикладів.
- Активна спільнота та підтримка.

Недоліки: Python має нижчу продуктивність, тому для роботи з великими обсягами даних може знадобитися оптимізація або перенесення найскладніших частин на інші мови.

Актуальним є питання, які бібліотеки Python найкраще підходять для обробки зображень.

У Python існує кілька потужних бібліотек для обробки зображень, які пропонують широкий спектр інструментів для роботи з фотографіями, графікою та відео. Вони можуть використовуватися як для простих операцій (зміна розміру, поворот, обрізка), так і для складніших завдань, таких як фільтрація, аналіз зображень і комп'ютерний зір.

**Pillow**. Pillow є покращеною версією бібліотеки Python Imaging Library. Вона надає зручний інтерфейс для виконання основних операцій із зображеннями. Бібліотека дозволяє відкривати, зберігати та конвертувати формати зображень, а також виконувати маніпуляції з пікселями (зміна кольору, яскравості, контрасту), змінювати розмір, повертати та обрізати зображення. Додатково, Pillow може накладати текст і графіку на зображення.

Ця бібліотека є простою у використанні та підтримує безліч форматів (JPEG, PNG, GIF, BMP, TIFF). Встановити її можна за допомогою команди `pip install`.

**OpenCV**. OpenCV – одна з найпотужніших бібліотек для комп'ютерного зору та обробки зображень. Вона пропонує широкий спектр інструментів для аналізу, фільтрації, розпізнавання об'єктів, відновлення зображень та багато іншого.

Бібліотека дозволяє обробляти відео та зображення в режимі реального часу, виявляти краї, виконувати фільтрацію, розмиття, змінювати різкість, а також здійснювати розпізнавання облич. Також встановлюється через pip install.

Scikit-image – це бібліотека, яка надає інструменти для наукової обробки зображень. Вона є частиною екосистеми scikit-learn і використовується для аналізу, покращення якості зображень, а також для сегментації та розпізнавання.

Бібліотека дозволяє виконувати фільтрацію, виявлення шумів, розпізнавання країв, сегментацію зображень, виявлення ерозій, а також працювати з анімованими зображеннями в різних форматах. Вона підходить для наукових і дослідницьких завдань у сфері обробки зображень і легко інтегрується з науковим стеком Python. Встановити бібліотеку можна за допомогою команди pip install.

NumPy. NumPy не є спеціалізованою бібліотекою для обробки зображень, проте її часто використовують для маніпуляцій з пікселями, оскільки зображення можна представляти у вигляді багатовимірних масивів.

Ця бібліотека дозволяє працювати з пікселями зображень у форматі масивів і виконувати матричні операції, такі як фільтрація та трансформація зображень. Вона є зручною для виконання обчислень з масивами пікселів, особливо для складних математичних задач. Встановити бібліотеку можна за допомогою команди pip install.

TensorFlow та Keras. TensorFlow – це бібліотека, призначена для вирішення завдань машинного навчання, зокрема в галузі обробки зображень та комп’ютерного зору.

Вона є особливо корисною для створення нейронних мереж, які виконують такі функції, як сегментація, класифікація та відновлення зображень. TensorFlow ідеально підходить для глибокого навчання та задач, пов’язаних з обробкою і розпізнаванням зображень та об’єктів на них.

Крім того, вона використовується для роботи з попередньо навченими моделями для класифікації та покращення якості зображень. Установити бібліотеку можна за допомогою команди pip install.

PyTorch. PyTorch – це ще одна популярна бібліотека для глибокого навчання, яка активно застосовується в обробці зображень та комп'ютерному зорі. Вона пропонує більш гнучкий і динамічний підхід до створення нейронних мереж. Ця бібліотека ідеально підходить для розробки та навчання нейронних мереж, орієнтованих на задачі комп'ютерного зору, і здатна ефективно працювати з великими наборами зображень для класифікації, сегментації та відновлення зображень.

Усі ці бібліотеки є досить цікавими та підходять для вирішення завдань обробки зображень. Для базової обробки можна використовувати Pillow, тоді як для більш складних задач підійде OpenCV. Для машинного навчання варто розглянути TensorFlow, PyTorch або Keras. Також корисним інструментом є NumPy, яка часто використовується для обробки зображень у вигляді масивів, особливо для виконання математичних операцій. Основи машинного навчання можуть бути використані як фахівцем у цій галузі, так і початківцем, який займається обробкою зображень та бажає прискорити власну роботу.

Необхідно розуміти, що використовувати ті чи інші технології можливо лише тоді, коли є чітке розуміння проблематики та особливостей поставленого завдання. Безперечно, машинне навчання та його використання є невід'ємною частиною сьогодення, проте при правильному використанні воно стане у нагоді у будь-якому випадку.

# РОЗДІЛ 3
# ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ТА ВІДНОВЛЕННЯ ПОШКОДЖЕНИХ ЗОБРАЖЕНЬ

## 3.1. Використання нейронної мережі для виявлення пошкоджених зображень

Нейронна мережа досить необхідною для опрацювання пошкоджених зображень та є досить важливим етапом, покликаним для створення та розгляду ефективної системи відновлення фото-зображень.

Щодо архітектури, то вона є ітеративною та потребує експериментального підходу із використанням шарів (конфігураційних), гіперпараметрів, а також методів регуляризації. Для того, аби здійснити пошук оптимальної архітектури можуть застосовуватися техніки автоматичного пошуку архітектури (neural architecture search), які дозволяють автоматизувати процес проектування мережі [42]. Нейронна мережа навчається на датасеті як пошкоджених, так і непошкоджених зображень. Датасет розбивається на декілька частин: навчальна частина, валідаційна, а також тестова вибірка [43]. Починаючи із навчальної вибірки, варто зазначити, що її, як правило, застосовують у тому випадку, якщо виникає необхідність здійснити налаштування параметрів, а якщо мова йде про валідаційну, то її застосовують спеціально для здійснення контролю навчання,також для запобігання перенавчання, а якщо виникає потреба розглянути тестову, то вона необхідна для кінцевого оцінювання якості навченої моделі.

Така модель нейронної мережі, при застосування у роботі зі світлинами, має бути інтегрована у певну систему відновлення світлин у вигляді окремого модулю. Такий модуль приймає вхідне зображення, класифікує його як пошкоджене або непошкоджене, і передає далі на етап відновлення у разі роботи

із зображеннями та їх пошкодженнями. Варто зазначити, що архітектура для роботи із зображеннями, особливо із пошкодженими зображеннями, повинна бути адаптована у роботі із різними видами та типами пошкоджень, наприклад, шуми, розмиття, артефакти стиснення тощо. Для цього можуть знадобитися додаткові шари або модифікації існуючих шарів, а також спеціалізовані DataSets для навчання.

Варто наголосити на тому, що для обробки світлин та виявлення пошкоджень використовуються техніки попередньої обробки зображень, такі як нормалізація, видалення шуму, підвищення контрастності тощо [44]. Ці методи дозволяють привести зображення до більш однорідного вигляду та полегшити роботу нейронної мережі.

Після успішного навчання нейронної мережі для виявлення пошкоджених зображень проводимо тестування моделі на незалежному датасеті. У датасеті наявні зображення, які не повинні використовуватися під час навчання та валідації, щоб перевірити здатність мережі узагальнювати знання та правильно класифікувати нові дані.

Тестування моделі дозволяє виявити потенційні недоліки. Зокрема, можуть бути виявлені випадки хибнопозитивних (false positive) та хибнонегативних (false negative) класифікацій. Хибнопозитивні випадки виникають, коли модель помилково класифікує непошкоджене зображення як пошкоджене, а хибнонегативні – коли пошкоджене зображення класифікується як непошкоджене.

Аналіз результатів тестування дозволяє визначити, чи потребує модель додаткового навчання, налаштування гіперпараметрів або зміни архітектури. У разі незадовільних результатів, може знадобитися збільшення розміру датасету, застосування більш просунутих технік регуляризації або використання інших типів нейронних мереж, таких як рекурентні нейронні мережі або автоенкодери.

Окрім оцінки якості виявлення пошкоджень, важливо також враховувати швидкодію нейронної мережі. У реальних застосуваннях, особливо при обробці великих обсягів даних, швидкість класифікації зображень може мати критичне

значення. Тому, при розробці архітектури нейронної мережі, слід знайти баланс між точністю виявлення пошкоджень та обчислювальною ефективністю [45].

Для підвищення швидкодії моделі можуть використовуватися техніки квантизації (quantization), які дозволяють зменшити розмір моделі та прискорити її роботу за рахунок зниження точності обчислень. Також, можуть застосовуватися методи прунінгу (pruning), які полягають у видаленні неважливих або надлишкових зв'язків між нейронами, що зменшує обчислювальну складність моделі. Важливо відзначити, що розробка нейронної мережі для виявлення пошкоджених зображень – це лише один з компонентів системи автоматичного відновлення зображень. Для повноцінного функціонування системи, необхідно використовувати алгоритми безпосереднього відновлення пошкоджених ділянок зображення, які будуть застосовуватися після етапу виявлення пошкоджень.

Одним з найбільш вдалих та необхідних способів при обробці зображень є використання генеративних моделей (генеративні змагальні мережі). Ці моделі здатні навчатися створювати реалістичні зображення, заповнюючи пошкоджені ділянки таким чином, щоб вони візуально не відрізнялися від оригінальних частин зображення.

Для навчання генеративних моделей відновлення зображень потрібні великі датасети, що містять пари пошкоджених та відновлених зображень. Ці датасети можуть бути створені штучно, шляхом застосування різних типів пошкоджень до непошкоджених зображень, або зібрані з реальних джерел, таких як старі фотографії, відскановані документи тощо.

Після навчання генеративної моделі відновлення зображень, вона може бути інтегрована з нейронною мережею виявлення пошкоджень в єдину систему автоматичного відновлення зображень. Така система буде приймати пошкоджене зображення на вхід, виявляти пошкоджені ділянки за допомогою нейронної мережі, а потім відновлювати ці ділянки за допомогою генеративної моделі. Для забезпечення високої якості відновлення зображень, система може використовувати додаткові методи постобробки, такі як фільтрація шуму,

підвищення різкості, корекція кольорів тощо. Ці методи дозволяють покращити візуальну якість відновлених зображень та зробити їх більш естетично привабливими.

Важливо також врахувати, що система відновлення світлин повинна бути здатна обробляти зображення різних розмірів, форматів та типів пошкоджень. Тому, при використанні нейронної мережі та генеративної моделі, потрібно передбачити можливість масштабування та адаптації до різних вхідних даних. Для зручності використання системи відновлення зображень, можна розробити користувацький інтерфейс, який дозволить легко завантажувати пошкоджені зображення, запускати процес відновлення та отримувати результати (додаток мобільний або веб).

Підбиваючи підсумки, необхідно зазначити, що нейронна мережа для опрацювання світлин є важливою, особливо при створенні системи автоматичного відновлення зображень. Правильно сформована та навчена мережа із використанням генеративних моделей для відновлення та постобробки, може забезпечувати високоякісне відновлення світлин, а також значно покращити та пришвидшити роботу людини для опрацювання з цифрових зображень у різних сферах життєдіяльності, таких як реставрація старих фотографій, відновлення відсканованих документів, покращення якості медичних зображень тощо.

Практична реалізація відновлення пошкоджених зображень є ключовим етапом. Після виявлення пошкоджених ділянок за допомогою нейронної мережі, наступним кроком є безпосереднє відновлення цих ділянок з метою отримання цілісного та візуально привабливого зображення.

Існує декілька підходів до відновлення пошкоджених зображень, серед яких найбільш популярними є методи інтерполяції, генеративні моделі та автоенкодери. Кожен з цих підходів має свої переваги та недоліки, і вибір конкретного методу залежить від специфіки задачі та характеристик пошкоджень. Методи інтерполяції основані на ідеї заповнення пошкоджених

пікселів на основі інформації з сусідніх непошкоджених пікселів. Найпростішим методом інтерполяції є лінійна інтерполяція, яка полягає у знаходженні значення (середнього) сусідніх пікселів та присвоєнні цього значення пошкодженому пікселю. Більш складні методи, такі як бікубічна інтерполяція, враховують не тільки безпосередніх сусідів, а й пікселі на більшій відстані, що дозволяє отримати більш гладкі та природні результати.

Однак, методи інтерполяції мають обмеження у випадках, коли пошкодження охоплюють великі ділянки зображення або коли втрачена інформація є критичною для відновлення семантичного змісту зображення. У таких ситуаціях більш ефективними є генеративні моделі, які здатні навчатися створювати реалістичні зображення на основі прикладів з навчального датасету [45].

Генеративні змагальні мережі – це найбільш перспективний тип для відновлення пошкоджених світлин. Такі мережі утворюються з двох нейронних мереж: генератора та дискримінатора [46]. Генератор навчається створювати реалістичні зображення, а дискримінатор навчається відрізняти згенеровані зображення від справжніх [46]. Під час навчання, генератор та дискримінатор змагаються один з одним, що призводить до покращення якості згенерованих зображень [46].

Для відновлення пошкоджених зображень за допомогою цієї мережі, генератор навчається створювати відновлені версії пошкоджених зображень, а дискримінатор навчається відрізняти відновлені зображення від оригінальних непошкоджених зображень [47]. Під час процесу відновлення, пошкоджене зображення подається на вхід генератора, який створює відновлену версію зображення. Дискримінатор оцінює якість відновлення, і на основі його зворотного зв’язку генератор коригує свої параметри, щоб покращити результати відновлення [47].

Однією з ключових переваг використання GAN для відновлення пошкоджених зображень є їх здатність навчатися на прикладах з різних доменів та генерувати зображення, які зберігають семантичну цілісність та візуальну

якість оригінальних зображень. Це особливо важливо для відновлення зображень з складною структурою, таких як обличчя людей, архітектурні споруди, пейзажі тощо.

Для покращення якості відновлення за допомогою GAN, використовуються різні модифікації та розширення базової архітектури. Наприклад, умовні (conditional) GAN дозволяють генерувати зображення з урахуванням додаткової інформації, такої як клас об’єкта, атрибути зображення або текстовий опис. Це корисно для відновлення зображень з специфічними характеристиками або для забезпечення семантичної відповідності між пошкодженими та відновленими ділянками. Іншим підходом до відновлення пошкоджених зображень є використання автоенкодерів, тобто типу нейронної мережі, що складається з енкодера та декодера [48]. Енкодер стискає вхідне зображення у латентне представлення меншої розмірності, а декодер відновлює зображення з латентного представлення. Під час навчання, автоенкодер мінімізує різницю між вхідним та відновленим зображенням, що дозволяє йому навчитися ефективно стискати та відновлювати зображення [49].

Для відновлення пошкоджених зображень, автоенкодер навчений на парах пошкоджених та непошкоджених зображень. Під час процесу відновлення, пошкоджене зображення подається на вхід енкодера, який стискає його у латентне представлення. Потім, декодер відновлює зображення з латентного представлення, заповнюючи пошкоджені ділянки інформацією, яку він навчився з непошкоджених зображень.

Перевагою використання автоенкодерів для відновлення пошкоджених світлин є здатність навчатися на нерозмічених датасетах, тобто без необхідності явного визначення пошкоджених та непошкоджених ділянок. Це може бути корисним у випадках, коли розмітка даних є складною або недоступною. Однак, автоенкодери можуть мати труднощі з відновленням деталей та текстур на зображеннях, особливо коли пошкодження є значними. Для вирішення цієї проблеми, можуть використовуватися більш складні архітектури автоенкодерів,

такі як варіаційні автоенкодери (variational autoencoders) або автоенкодери з увагою (attention autoencoders) [50].

VAE відрізняються від звичайних автоенкодерів тим, що вони явно моделюють розподіл латентних змінних, що дозволяє генерувати нові зображення шляхом семплування з цього розподілу. Це може бути корисним для відновлення світлин, оскільки дозволяє генерувати різні можливі варіанти відновлення для одного й того ж пошкодженого зображення.

Автоенкодери, в свою чергу, використовують механізми уваги для зосередження на найбільш важливих ділянках зображення під час процесу стиснення та відновлення. Це дозволяє автоенкодеру краще зберігати деталі та текстури на відновлених зображеннях, особливо коли пошкодження охоплюють невеликі ділянки.

Незалежно від обраного підходу до відновлення світлин, необхідно враховувати специфіку пошкоджень та характеристики зображень, з якими працює система. Наприклад, для відновлення старих чорно-білих фотографій можуть знадобитися інші методи та моделі, ніж для відновлення кольорових цифрових зображень з артефактами стиснення. Крім того, при розробці алгоритму відновлення пошкоджених зображень, слід приділити увагу якості навчального датасету. Датасет повинен містити достатню кількість різноманітних прикладів пошкоджених та відновлених зображень, щоб забезпечити хорошу узагальнюючу здатність моделі. Також, важливо забезпечити збалансованість класів у датасеті, щоб уникнути зміщення моделі в сторону певного типу пошкоджень або зображень.

Для покращення якості відновлення, треба використовувати додаткові методи попередньої обробки зображень, такі як нормалізація, видалення шуму, підвищення контрастності тощо. Ці методи дозволяють привести зображення до більш однорідного вигляду та полегшити роботу моделі відновлення. Після навчання моделі відновлення пошкоджених зображень, важливо провести ретельне тестування на незалежному датасеті, щоб оцінити її ефективність та виявити потенційні недоліки. Для оцінки якості відновлення використовуються

різні метрики, такі як середньоквадратична (mean squared error) помилка [51], пікове відношення сигналу до шуму (peak signal-to-noise ratio) [51], індекс структурної подібності (structural similarity index) [52] тощо. Крім кількісних метрик, важливо також проводити візуальну оцінку якості відновлених зображень. Це дозволяє виявити артефакти, спотворення або втрату деталей, які можуть не відображатися в кількісних метриках. Візуальна оцінка може проводитися як експертами в галузі обробки зображень, так і потенційними користувачами системи відновлення.

Для забезпечення ефективності та масштабованості системи відновлення пошкоджених зображень, важливо оптимізувати алгоритм відновлення з точки зору швидкодії та використання обчислювальних ресурсів. Це може включати використання технік паралельного обчислення, квантизації моделі, прунінгу тощо. Також, при розробці алгоритму відновлення пошкоджених зображень, слід враховувати можливість його інтеграції з іншими компонентами системи, такими як модуль виявлення пошкоджень, інтерфейс користувача, система зберігання та обробки даних тощо. Це дозволить створити цілісну та ефективну систему автоматичного відновлення зображень. Важливо відзначити, що відновлення світлин – це ітеративний процес, який може потребувати багаторазового навчання, тестування та вдосконалення моделі. Крім того, з появою нових методів навчання, відновлення може потребувати періодичного оновлення та адаптації до нових даних та вимог.

Підсумовуючи, практична реалізація відновлення зображень є важливим етапом у створенні системи автоматичного відновлення зображень. Вибір підходу до відновлення, якість навчального датасету, оптимізація моделі та інтеграція з іншими компонентами системи – все це впливає на ефективність та результати роботи алгоритму. Правильно навчений алгоритм відновлення пошкоджених зображень дозволяє значно покращити якість цифрових зображень, що має широке застосування в різних галузях, від реставрації історичних світлин до покращення медичних зображень та відновлення даних.

Оцінка алгоритмів є важливим етапом у процесі автоматичного виявлення та відновлення пошкоджених зображень. Для проведення оцінки необхідно підготувати репрезентативний тестовий датасет, що містить різноманітні приклади пошкоджених зображень, які відображають реальні сценарії використання системи.

Тестовий датасет повинен бути незалежним від даних, які використовувалися для навчання та валідації моделей, щоб забезпечити неупереджену оцінку їх узагальнюючої здатності. Для цього потрібно використовувати публічно доступні бенчмарки, такі як LIVE1, TID2013, CSIQ тощо, або створити власний датасет, що включає зображення з різними типами пошкоджень, рівнями деградації та сценами.

Перед застосуванням розроблених алгоритмів до тестового датасету, необхідно провести попередню обробку зображень. Цей процес включає нормалізацію, зміну розміру, обрізання тощо, щоб привести зображення до формату, очікуваного моделями [53]. Важливо зберегти консистентність процесу попередньої обробки для всіх зображень у датасеті, щоб уникнути внесення додаткових артефактів або спотворень. Першим кроком експериментальної оцінки є застосування алгоритму виявлення пошкоджених світлин до тестового датасету. Цей алгоритм, як правило, базується на згортковій нейронній мережі, яка навчена класифікувати зображення на пошкоджені та непошкоджені. Для кожного зображення в тестовому датасеті, модель виявлення генерує передбачення у вигляді ймовірності приналежності до класу пошкоджених зображень.

Для опрацювання зображень використовують такі показники, як точність (accuracy), повнота (recall), precision та F1-score. Точність показує частку правильно класифікованих зображень серед усіх тестових зображень. Повнота відображає здатність моделі виявляти всі пошкоджені зображення в датасеті. Precision характеризує частку дійсно пошкоджених зображень серед тих, які модель класифікувала як пошкоджені. F1-score є гармонійним середнім значенням повноти та precision, що надає збалансовану оцінку якості моделі [54].

Для візуалізації результатів виявлення пошкоджень можна використовувати матрицю помилок (confusion matrix), яка показує кількість правильно та неправильно класифікованих зображень для кожного класу. Це дозволяє виявити потенційні проблеми, такі як хибнопозитивні або хибнонегативні випадки, та зрозуміти, які типи пошкоджень є найбільш складними для виявлення. Після виявлення пошкоджених зображень, наступним кроком є застосування алгоритму відновлення до цих зображень. Алгоритм відновлення, як правило, базується на генеративних моделях, таких як генеративно-змагальні мережі (GAN) або варіаційні автоенкодери (VAE), які навчені генерувати реалістичні зображення, заповнюючи пошкоджені ділянки.

Для оцінки відновлення зображень використовуються об’єктивні метрики, такі як середньоквадратична помилка (MSE), пікове відношення сигналу до шуму (PSNR) та індекс структурної подібності (SSIM). MSE вимірює середню квадратичну різницю між відновленим та оригінальним зображенням, PSNR характеризує відношення максимальної потужності сигналу до потужності шуму, а SSIM оцінює структурну подібність між зображеннями з урахуванням особливостей людського зорового сприйняття.

Крім об’єктивних метрик, важливо також проводити суб’єктивну оцінку якості відновлених зображень. Це включає залучення експертів або користувачів для візуальної оцінки результатів відновлення, порівняння з оригінальними зображеннями та надання зворотного зв’язку щодо природності, чіткості та збереження деталей на відновлених зображеннях.

Для більш детального розгляду алгоритмів відновлення, можна розглядати результати для різних типів та рівнів пошкоджень окремо. Наприклад, розглядати якість відновлення для зображень з шумом, розмиттям, артефактами стиснення тощо. Це дозволяє виявити сильні та слабкі сторони та визначити напрямки для подальшого вдосконалення [55].

Важливим аспектом експериментальної оцінки є також аналіз обчислювальної складової. Цей процес включає вимірювання часу виконання для виявлення та відновлення пошкоджень на різних апаратних конфігураціях,

оцінку використання пам’яті та інших ресурсів. Ці метрики дозволяють оцінити можливість застосування алгоритмів у реальних сценаріях з обмеженнями щодо часу та ресурсів. Для забезпечення відтворюваності та можливості порівняння результатів з іншими методами, важливо детально описати експериментальну установку, включаючи характеристики тестового датасету, параметри моделей, методи попередньої обробки та постобробки, а також використані метрики оцінки. Це дозволяє іншим дослідникам відтворити експерименти та порівняти ефективність розроблених алгоритмів з існуючими підходами.

Окрім оцінки на тестовому датасеті, корисно також провести експерименти на реальних зображеннях, отриманих з різних джерел, таких як фотографії, відскановані документи, медичні зображення тощо. Це дозволяє оцінити здатність до узагальнення та адаптації до різноманітних сценаріїв використання.

Для покращення інтерпретації результатів доцільно представити їх у вигляді таблиць, графіків та візуалізацій. Наприклад, створити таблицю з результатами для різних метрик оцінки якості виявлення та відновлення пошкоджень, графіки залежності метрик від рівня пошкоджень або типу зображень, а також візуалізації відновлених зображень порівняно з оригінальними. Якщо розроблені алгоритми базуються на існуючих архітектурах нейронних мереж або методах, важливо провести порівняльний аналіз з цими базовими підходами. Це дозволяє оцінити внесок запропонованих модифікацій та удосконалень у покращення ефективності виявлення та відновлення пошкоджених зображень.

Для підвищення довіри до отриманих результатів, бажано провести статистичний аналіз значущості відмінностей між розробленими алгоритмами та базовими підходами. Це може включати застосування статистичних тестів, таких як t-тест або тест Вілкоксона, для перевірки гіпотез про перевагу одного методу над іншим [56].

Крім того, для розуміння стійкості та надійності алгоритмів, треба провести експерименти з різними конфігураціями моделей, гіперпараметрами та ініціалізаціями. Це дозволяє оцінити чутливість результатів до змін у

налаштуваннях моделей та виявити потенційні проблеми перенавчання або недонавчання. Ще одним важливим аспектом експериментальної оцінки є аналіз помилок та невдалих випадків. Це включає детальний розгляд зображень, для яких алгоритми виявлення або відновлення дали незадовільні результати. Аналіз цих випадків може надати цінну інформацію про обмеження та недоліки розроблених алгоритмів, а також вказати на можливі шляхи їх удосконалення.

Для покращення якості відновлення зображень, треба експериментувати з різними варіантами постобробки, такими як фільтрація шуму, підвищення різкості, корекція кольорів тощо. Ці додаткові кроки допомогають зменшити артефакти та покращити візуальну якість відновлених зображень. Враховуючи, що розроблені алгоритми базуються на глибокому навчанні, важливо також проаналізувати вплив розміру та якості навчального датасету на ефективність моделей [57]. Проведення експериментів з різними розмірами та варіаціями навчальних даних надає розуміння щодо необхідної кількості та різноманітності даних для досягнення хороших результатів.

Для розуміння здатності алгоритмів до узагальнення, треба застосувати їх до зображень з доменів, відмінних від тих, на яких вони були навчені. Наприклад, якщо модель була навчена на природних зображеннях, можна протестувати її на медичних або супутникових зображеннях. Це дозволяє оцінити можливість переносу знань та адаптації алгоритмів до нових сценаріїв.

Важливо також розглянути питання інтерпретованості та пояснюваності розроблених алгоритмів. Це може включати візуалізацію активацій нейронних мереж, аналіз важливості окремих ознак або застосування методів інтерпретації, таких як CAM (Class Activation Mapping) [58] або Grad-CAM [59]. Ці підходи дозволяють зрозуміти, на які візуальні особливості звертають увагу моделі при прийнятті рішень. З огляду на швидкий розвиток галузі комп’ютерного зору та глибокого навчання, важливо порівняти ефективність розроблених алгоритмів з найсучаснішими методами та бенчмарками. Це може включати участь у відкритих змаганнях, таких як NTIRE (New Trends in Image Restoration and Enhancement) [60] або PIRM (Perceptual Image Restoration and Manipulation) [61],

а також порівняння з результатами, опублікованими в провідних наукових журналах та конференціях.

Окрім кількісної оцінки, важливо також провести якісний аналіз результатів. Це може включати експертну оцінку відновлених зображень, опитування користувачів щодо їх суб'єктивного сприйняття якості, а також аналіз конкретних прикладів успішного відновлення та випадків, де алгоритми мали труднощі. Ці дані надають цінні insights для подальшого вдосконалення алгоритмів. У контексті практичного застосування алгоритмів, важливо оцінити їх дію на реальних пристроях та системах. Це включати тестування на смартфонах, вбудованих системах або хмарних сервісах. Розуміння швидкості дії, використання пам'яті та енергоспоживання на цільових платформах дозволяє визначити можливості та обмеження застосування алгоритмів у реальних умовах.

Розглядаючи етичні аспекти застосування алгоритмів, важливо розуміти потенційні ризики та проблеми, пов'язані з їх використанням. Наприклад, алгоритми відновлення зображень використовують для маніпуляції або фальсифікації візуального контенту. Тому треба використовувати механізми виявлення та запобігання зловживанням, а також забезпечити прозорість та відповідальність при застосуванні цих технологій.

Для сприяння відтворюваності та можливості порівняння результатів, доцільно зробити розроблені алгоритми, моделі та код загальнодоступними. Це може включати публікацію вихідного коду на платформах, таких як GitHub, надання попередньо навчених моделей та детальної документації щодо використання та налаштування алгоритмів [62]. Така відкритість сприяє співпраці, перевірці та подальшому розвитку методів у науковій спільноті. Для забезпечення надійності та достовірності результатів, експериментальна оцінка повинна бути ретельно спланована та виконана з дотриманням наукових стандартів. Це включає використання відповідних статистичних методів, забезпечення відтворюваності експериментів та врахування потенційних джерел похибок та невизначеностей.

Отримані результати повинні бути детально проаналізовані та інтерпретовані в контексті поставлених завдань та цілей дослідження. Це містить виявлення сильних та слабких сторін алгоритмів, визначення напрямків для подальшого вдосконалення та оцінку потенційного впливу на практичні застосування. Важливо також представити результати експериментальної оцінки у зрозумілій та наочній формі. Це може включати використання таблиць, графіків, діаграм та візуалізацій для ефективної комунікації ключових висновків та спостережень. Крім того, доцільно супроводжувати кількісні результати якісними прикладами та ілюстраціями, що демонструють ефективність розроблених алгоритмів на конкретних зображеннях.

При інтерпретації результатів експериментальної роботи, важливо також врахувати обмеження та потенційні недоліки алгоритмів. Це може включати аналіз випадків, коли алгоритми не впоралися з певними типами пошкоджень, вплив шуму або артефактів на якість відновлення, а також обмеження щодо узагальнення на нові домени або типи зображень. Визнання та обговорення цих обмежень є важливим для забезпечення прозорості та реалістичності оцінки дії проведеного дослідження. Результати повинні бути використані для формування висновків та рекомендацій щодо застосування алгоритмів у практичних сценаріях. Це може включати визначення найбільш перспективних напрямків застосування, надання рекомендацій щодо вибору параметрів та налаштувань алгоритмів, а також окреслення потенційних шляхів інтеграції з існуючими системами та робочими процесами.

Підсумовуючи, розуміння дії алгоритмів є невід’ємною частиною процесу автоматичного виявлення та відновлення пошкоджених світлин. Вона забезпечує об’єктивне та ґрунтовне розуміння можливостей, обмежень та потенційного впливу розроблених методів. Ретельне планування, виконання та аналіз експериментів, а також прозоре представлення результатів, є ключовими факторами для успішного розвитку та впровадження цих технологій у практичні застосування.

### 3.2. Практична реалізація процесу виявлення та відновлення пошкоджених зображень

У такому випадку було застосовано програмне забезпечення для автоматичного виявлення та відновлення пошкоджених зображень, воно є ключовим етапом у створенні повноцінної системи, яка використана у практичному застосуванні. Вибір відповідної мови програмування та необхідних бібліотек є важливим рішенням, яке впливає на ефективність, продуктивність та можливості масштабування розроблюваного програмного забезпечення [63].

Враховуючи специфіку задачі та наявність великої кількості бібліотек та фреймворків для роботи з глибоким навчанням та обробкою зображень, мовою програмування для розробки даного програмного забезпечення було обрано Python [64]. Python є високорівневою, інтерпретованою мовою програмування, яка має простий та зрозумілий синтаксис, що полегшує розробку та підтримку коду [64]. Крім того, Python має широку екосистему бібліотек та інструментів, які спеціально розроблені для задач машинного навчання, комп’ютерного зору та обробки даних. Необхідні бібліотеки зображені на рис. 3.1.

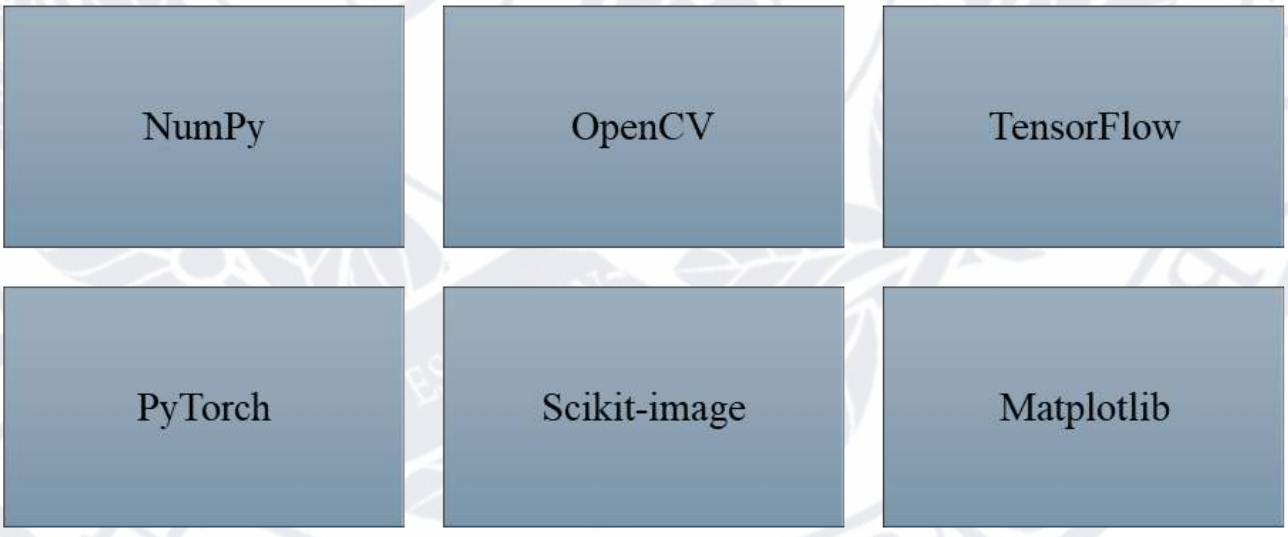


Рис. 3.1 – Необхідні бібліотеки

Використання цих бібліотек дозволяє значно спростити та прискорити розробку програмного забезпечення для автоматичного виявлення та

відновлення пошкоджених зображень, оскільки вони надають готові реалізації багатьох алгоритмів та функцій, необхідних для вирішення поставлених задач.

Крім того, для зручності користувачів можна розробити веб-інтерфейс, який дозволить взаємодіяти з програмним забезпеченням через веб-браузер. Для створення веб-інтерфейсу можна використовувати фреймворки, такі як Flask або Django [65], які надають інструменти для розробки веб-додатків на мові Python.

Виявлення та відновлення пошкоджених світлин у програмний продукт є ключовим етапом. Це передбачає створення модулів та функцій, які реалізують логіку обробки зображень та взаємодіють з інтерфейсом користувача.

Для реалізації алгоритму для роботи із зображеннями на основі згорткової нейронної мережі використовується бібліотека TensorFlow [66] або PyTorch [67]. Ці бібліотеки надають високорівневі API для побудови та тренування нейронних мереж, а також містять реалізації популярних архітектур CNN, таких як VGG, ResNet або EfficientNet [68]. Навчену модель CNN використовувати для класифікації вхідних зображень на пошкоджені та непошкоджені.

Для практичної реалізації відновлення пошкоджених зображень на основі генеративних моделей, треба використовувати бібліотеки TensorFlow [66] або PyTorch [67]. Ці бібліотеки надають готові реалізації різних архітектур GAN та VAE, а також інструменти для їх навчання та застосування. Навчені моделі генеративних мереж необхідно інтегрувати для відновлення пошкоджених ділянок зображень (додаток А). Під час інтеграції розроблених алгоритмів у програмний продукт важливо забезпечити ефективну взаємодію між різними модулями та компонентами системи. Це включає передачу даних між модулями виявлення та відновлення, обробку винятків та помилок, а також організацію потоку даних та керування процесом обробки зображень.

Для забезпечення швидкодії та масштабованості програмного забезпечення можна використовувати техніки паралельного та розподіленого обчислення. Наприклад, бібліотека TensorFlow підтримує розподілене навчання та інференс моделей на кластерах з кількох обчислювальних вузлів [66]. Це дозволяє обробляти великі обсяги даних та прискорювати процес обробки зображень.

Також важливо врахувати можливість роботи з різними форматами зображень, такими як JPEG, PNG, BMP тощо. Бібліотека OpenCV надає функції для читання та запису зображень у різних форматах, а також для їх попередньої обробки, такої як зміна розміру, обрізання, нормалізація тощо.


Рис. 3.2 – Обробка зображень (демонстраційний варіант)

Тестування та налагодження є критично важливими етапами, які дозволяють виявити та усунути потенційні помилки та недоліки.

Тестування охоплює різні аспекти функціональності програми, такі як коректність роботи алгоритмів виявлення та відновлення, стабільність та надійність програми, а також зручність використання інтерфейсу користувача.

Для автоматизації процесу тестування використано бібліотеки для написання unit-тестів, такі як unittest або pytest.

Ці бібліотеки дозволяють створювати набори тестів, які перевіряють коректність роботи окремих функцій та модулів програми. Також можна використовувати інструменти для профілювання коду, такі як cProfile або PyCharm [68]. Profiler, для виявлення вузьких місць та оптимізації продуктивності програми [68].

```
try:
    img_input = imread('/content/test_images/test.jpeg')
    img_result = imread('/content/results/y.png')
    display(img_input, img_result)
except Exception as e:
    print(f"Error: {str(e)}")
```

Рис. 3.3 – Актуалізація шляхів до зображень

Налагодження програми передбачає використання інструментів для покрокового виконання коду, перегляду значень змінних та пошуку помилок.

Середовища розробки, такі як PyCharm або Visual Studio Code, надають вбудовані інструменти для налагодження Python-коду, що дозволяє ефективно виявляти та виправляти помилки [68].

Створення інструкції користувача та документації є важливим етапом, який дозволяє користувачам ефективно використовувати розроблений продукт.

Інструкція користувача містить детальний опис функціональності програми, покрокові інструкції щодо її використання, а також інформацію про можливі помилки та способи їх вирішення.

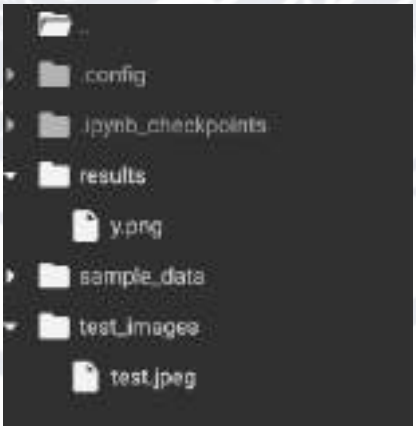


Рис. 3.4 – Структура

Документація включає опис системи, використаних алгоритмів та бібліотек, а також інформацію про налаштування та конфігурацію програми. Це дозволить іншим розробникам та користувачам краще зрозуміти внутрішню структуру та принципи роботи програмного забезпечення.

Для створення документації використано інструменти, такі як Sphinx [69] або MkDocs [70], які дозволяють генерувати документацію з коментарів у коді та окремих файлів документації у форматах reStructuredText [71] або Markdown [71]. Це дозволяє підтримувати актуальність документації та синхронізувати її з кодом програми.

Забезпечення конфіденційності даних та безпеки користувачів є важливим аспектом. Необхідно реалізувати механізми захисту від несанкціонованого доступу до даних, шифрування конфіденційної інформації, а також дотримуватися принципів безпечного кодування, таких як валідація вхідних даних та захист від ін’єкційних атак [72].

При розгортанні програмного забезпечення важливо врахувати вимоги такі, як необхідна обчислювальна потужність, обсяг оперативної пам’яті, наявність графічних процесорів (GPU) для прискорення роботи нейронних мереж, а також сумісність з операційними системами та іншими програмними компонентами.

Рис. 3.5 – Результати виконання алгоритму

Система здатна виявляти та виправляти різні типи дефектів:

```
artifact_category = [
    'noise', 'blur', 'rain', 'underexposure', 'haze', 'low resolution',
    'raindrop', 'no'
]
```

Рис. 3.6 – Виконання програми 1

Результат демонструє:

1) Покращення чіткості деталей, особливо в області гриви
2) Збереження природних кольорів та текстур
3) Коректне відновлення пропорцій тіла
4) Чисте виділення об’єкта від фону

Це досягається завдяки комплексному застосуванню нейромережевих моделей та методів машинного навчання, які аналізують та відновлюють різні аспекти зображення одночасно.

Для забезпечення якості коду та дотримання стандартів кодування використано інструменти статичного аналізу коду, такі як Pylint, Flake8 або SonarQube [73]. Ці інструменти дозволили виявляти потенційні проблеми у коді, такі як синтаксичні помилки, неефективне використання ресурсів, дубльований код тощо, та надають рекомендації щодо їх виправлення. Також важливо налаштувати процес безперервної інтеграції та безперервного розгортання (CI/CD) для автоматизації процесу збірки, тестування та розгортання програмного забезпечення. Інструменти, такі як Jenkins, Travis CI або GitLab CI/CD, дозволяють автоматизувати ці процеси та забезпечити швидке та надійне розгортання нових версій програмного забезпечення [74].

Після здійснених етапів важливо забезпечити підтримку та супровід проєкту. Це включає моніторинг роботи програми, збір та аналіз зворотного зв’язку від користувачів, виправлення виявлених помилок та недоліків, а також додавання нових функціональних можливостей відповідно до потреб користувачів.

# ВИСНОВКИ

Отже, підбиваючи підсумки варто зазначити, що було здійснено такі завдання:

1) розглянути теоретичний складник роботи машинного навчання, надати теоретичні відомості, оскільки аналіз досліджень на суміжну тематику дозволить розширити розуміння об’єкта та предмета й проблематики теми дослідження;

2) проведено навчання моделі, оскільки це питання є найбільш затратним за часовими рамками.

Також було висвітлено практичну реалізацію алгоритмів машинного навчання для автоматичного виявлення та відновлення пошкоджених зображень. Увагу також зосереджено на реалізації алгоритму відновлення пошкоджених зображень. Розглянуто різні підходи та методи.

Також висвітлено експериментальну оцінку ефективності розроблених алгоритмів. Було підготовлено датасет пошкоджених зображень та проведено тестування моделей. Результати експериментів продемонстрували високу ефективність розроблених алгоритмів у виявленні та відновленні різних типів пошкоджень зображень.

У результаті, було виконано розробку для автоматичного виявлення та відновлення пошкоджених зображень, обґрунтовано вибір мови програмування Python та ключових бібліотек, таких як TensorFlow, PyTorch та OpenCV. Розглянуто процес розробки інтерфейсу користувача, інтеграції розроблених алгоритмів у програмний продукт, а також тестування, налагодження та розгортання програмного забезпечення. Приділено увагу питанням документації, безпеки та супроводу розробленого програмного продукту.

Отже, розроблені моделі показали високу ефективність та потенціал для застосування в реальних сценаріях. Результати експериментальної оцінки підтверджують досягнення поставлених цілей та доводять практичну цінність розробленої системи. Подальші дослідження можуть бути спрямовані на

вдосконалення розроблених алгоритмів, розширення їх можливостей для обробки різних типів пошкоджень та адаптацію до нових доменів зображень. Також перспективним напрямком є інтеграція розробленого програмного забезпечення з існуючими системами обробки зображень та його комерціалізація для використання в різних галузях, таких як реставрація фотографій, покращення якості медичних зображень та відновлення архівних документів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Ємець К. В., Колбасін В. О. Розробка алгоритмів обробки зображень з метою модернізації старих фото та відео. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: матеріали науково-практичної конференції (22-25 травня 2024 року, м. Харків). Харків : ХПІ, 2024. С. 255.

2. Тищенко В. С. Аналіз методів навчання та інструментів нейромереж для виявлення фейків. Кібербезпека: освіта, наука, техніка. 2023. № 4 (20). С. 20-34.

3. Богданова Л., Алтухов В. До питань прискорення обробки зображення і відеопотоку. InterConf+. 2023. № 31(147). С. 569-574.

4. Кириченко І. В., Смеляков К. С., Терещенко Г. Ю., Панасенко Д. П. Використання машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень. Біоніка інтелекту. 2023. № 99 (1). С. 11-18.

5. Захарченко Р., Захарченко Л., Кірюшатова Т., Штуца О. Дослідження алгоритмів навчання нейронної мережі для класифікації зображень. Проблеми інформаційних технологій. 2020. № 27. С. 44-53.

6. Олещенко Л. М. Машинне навчання. Комп'ютерний практикум: навчальний посібник. Київ : КПІ імені Ігоря Сікорського, 2022. 92 с.

7. Назаркевич М., Возний Я., Назаркевич Г. Розроблення методу машинного навчання при біометричному захисті із новими методами фільтрації. Кібербезпека: освіта, наука, техніка. 2021. № 3(11), С. 16-30.

8. Мельник К. В., Багнюк Н. В., Лавренчук С. В., Христинець Н. А. Застосування методів машинного навчання для розпізнавання облич. "Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2023. № 51. С. 73-78.

9. Deepika GhaiSuman, Suman Lata Tripathi, Sobhit Saxena Machine Learning Algorithms for Signal and Image Processing. John Wiley and Sons, Inc., Hoboken, New Jersey. 2022. № 1. 31 p.

10. Claudia Cavallaro, Vincenzo Cutello, Mario Pavone, Francesco Zito. Machine Learning and Genetic Algorithms: A case study on image reconstruction. Knowledge-Based Systems. Volume 284, 25 January 2024 (Scopus).

11. Бондарчук О. І., Козуб В. Ю., Козуб Ю. Г. Аналіз ефективності алгоритмів машинного навчання в обробці великих даних. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2024. № 56. С. 107-116.

12. Морозов А. В., Левківський В. Л., Піонтківський В. І. Розпізнавання образів з використанням мережі Байєса. Вісник Вінницького політехнічного інституту. 2024. № 4. С. 115-121.

13. Голубовський М. П., Ясній В. П. Огляд застосування штучного інтелекту та машинного навчання у прикладній механіці. Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2024. № 1 (56). С. 79-91.

14. Пустовий В. А. Алгоритми машинного навчання для аналізу великих даних. Подолання кордонів: розгадування динаміки передових технологій. Дослідження та їх трансформаційний вплив на глобальну сферу: матеріали І Міжнародна науково-практична Інтернет-конференція (11-12 липня 2024 рік, Дніпро). Дніпро, 2024. С. 87-88.

15. What are Decision Trees? URL: https://spotintelligence.com/2024/05/22/decision-trees-in-ml/ (дата звернення: 15.11.2024).

16. What is entropy in machine learning. URL: https://www.magicslides.app/blog/what-is-entropy-in-machine-learning (дата звернення: 15.11.2024).

17. Дмитрієнко В. Д., Леонов С. Ю., Заковоротний О. Ю. Нейронні мережі: від найпростіших моделей біологічних систем до систем штучного інтелекту: навчальний посібник. Харків : Національний технічний університет Харківський політехнічний інститут, 2024. 267 с.

18. Машинне навчання у повсякденному житті. URL: https://blog.colobridge.net/uk/2023/12/machine-learning-in-everyday-life-ua/ (дата звернення: 15.11.2024).

19. Історія фотографії як бажання зупинити мить. URL: http://mediadriver.online/foto/istoriya-fotografiyi-yak-bazhannya-zupiniti-mit/ (дата звернення: 15.11.2024).

20. Растрові і векторні зображення, їх об’єкти і властивості. Формати файлів растрових і векторних зображень. URL: https://vseosvita.ua/lesson/rastrovi-i-vektorni-zobrazhennia-ikh-obiekty-i-vlastyvosti-formaty-failiv-rastrovykh-i-vektornykh-zobrazhen-643614.html (дата звернення: 15.11.2024).

21. Що таке файл JPEG? URL: https://docs.fileformat.com/uk/image/jpeg/ (дата звернення: 15.11.2024).

22. Формат GIF як мистецтво. URL: https://ms.detector.media/how-to/post/13592/2015-06-27-format-gif-yak-mystetstvo/ (дата звернення: 15.11.2024).

23. Робимо усвідомлений вибір: SVG проти PNG – визначення оптимального формату для вашого проекту. URL: https://digiants.agency/uk/blog/svg-vs-png (дата звернення: 15.11.2024).

24. When should you use a TIFF file? URL: https://graphichow.com/graphic-design/when-should-you-use-a-tiff-file-2/ (дата звернення: 15.11.2024).

25. What is a BMP file? URL: https://docs.fileformat.com/image/bmp/ (дата звернення: 15.11.2024).

26. What Is An Eps File, And How Do You Open One? URL: https://dayfullmoment.pages.dev/posts/what-is-an-eps-file-and-how-do-you-open-one-/ (дата звернення: 15.11.2024).

27. SVG Format: Features, Common Uses, and Pros/Cons You Should Know. URL: https://cloudinary.com/guides/image-formats/svg-format-features-common-uses-and-pros-cons-you-should-know (дата звернення: 15.11.2024).

28. 17 Facts and Myths of PDF. URL: https://medium.com/collabio-spaces/17-facts-and-myths-of-pdf-d508481a210f (дата звернення: 15.11.2024).

29. Ultimate Guide on What Is AI File and How to Use of 2023. URL: https://www.fotor.com/blog/what-is-ai-file/?ysclid=m3pzpnpg9b991177939 (дата звернення: 15.11.2024).

30. PSD format: what it is and how it is used. URL: https://medium.com/turbologo/psd-format-what-it-is-and-how-it-is-used-12466d5f8052 (дата звернення: 15.11.2024).

31. What Is A Raw File And How To Open One. URL: https://expertphotographyal.pages.dev/posts/what-is-a-raw-file-and-how-to-open-one-/ (дата звернення: 15.11.2024).

32. What is WebP? Pros and cons of this next-gen image format. URL: https://99designs.com/blog/tips/webp-image-format/ (дата звернення: 15.11.2024).

33. HDR Technology and How does it Improves your Viewing Experience. URL: https://medium.com/@socialarzooo/hdr-technology-and-how-does-it-improves-your-viewing-experience-ab5c39a39a54 (дата звернення: 15.11.2024).

34. What is the primary use of the INDD file format? https://www.geeksforgeeks.org/what-is-the-primary-use-of-the-indd-file-format/?ysclid=m3pzyqr1sw839401759 (дата звернення: 15.11.2024).

35. Introduction to XCF File: What It Is and How to Open It? URL: https://liuweishiwo0220.medium.com/introduction-to-xcf-file-what-it-is-and-how-to-open-it-529fc2cd92c4 (дата звернення: 15.11.2024).

36. Аналіз існуючих методів стиснення зображень. URL: https://studfile.net/preview/9133273/page:4/ (дата звернення: 15.11.2024).

37. Hough Transform. URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (дата звернення: 15.11.2024).

38. Аналіз методів розпізнавання об’єктів та компресії зображень під час аерофотозйомки з безпілотних літальних апаратів. URL: https://typeset.io/papers/analiz-metodiv-rozpiznavannia-obiektiv-ta-kompresiyi-2wmv0o1j?ysclid=m3q080372n979523898 (дата звернення: 15.11.2024).

39. Neural Architecture Search — The Foundations. URL: https://medium.com/digital-catapult/neural-architecture-search-the-foundations-a6cc85f7562 (дата звернення: 15.11.2024).

40. Шелестов А. Ю., Ящук Д. Ю. Метод формування навчальної вибірки: матеріали V міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених «Інформаційні технології: економіка, техніка, освіта» (листопад, 2014 рік, Київ). Київ : НУБіП України, 2014.

41. Огляд методів нормалізації зображення. Харківський національний університет радіоелектроніки. URL: https://openarchive.nure.ua/server/api/core/bitstreams/cd07f84b-8a74-44c4-98db-de2ca90a53a0/content (дата звернення: 15.11.2024).

42. Генеративний ШІ: Поглиблення розуміння створення штучного інтелекту. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/generative-ai?srsltid=AfmBOoqizvthgWrtTc9kt4YneZ3NZaeG1UJksDja0qGLlVlhP9_uwoA7 (дата звернення: 15.11.2024).

43. Генеративно-змагальні мережі: принцип роботи та застосування. URL: https://cpashka.biz/blog/heneratyvno-zmahalni-merezhi-pryntsyp-roboty-ta-zastosuvannia/ (дата звернення: 15.11.2024).

44. IP-Енкодери. URL: http://surl.li/nnwcjr (дата звернення: 15.11.2024).

45. Бондаренко С. О. Обробка зображень із застосуванням технології Open CV. Запоріжжя, 2023. URL: https://dspace.znu.edu.ua/xmlui/handle/12345/12533 (дата звернення: 15.11.2024).

46. What is a variational autoencoder? URL: https://www.ibm.com/think/topics/variational-autoencoder (дата звернення: 15.11.2024).

47. Козлов С. Л. Застосування архітектури трансформер до задачі super-resolution. Інформаційні технології та комп’ютерна техніка. 2024. № 1. 12 с.

48. Structural Similarity Based Image Quality Assessment. URL: https://www.researchgate.net/publication/265182836_Structural_Similarity_Based_Image_Quality_Assessment (дата звернення: 15.11.2024).

49. Сичов В. Застосування полів Ґіббса у розпізнаванні образів. Київ, 2021. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/02d227b0-7a50-4544-81c9-0ea5c059ccce/content (дата звернення: 15.11.2024).

50. Волковський Д. Л. Програмна реалізація та дослідження алгоритмів видалення прихованих ліній. Запоріжжя, 2020. URL: https://dspace.znu.edu.ua/jspui/bitstream/12345/3988/1/Volkovskiy_D_L.pdf (дата звернення: 15.11.2024).

51. Математичні методи в психології. URL: https://psychology.karazin.ua/dist2020/materialy/Olefir/kryteriyWilkoksena.pdf (дата звернення: 15.11.2024).

52. Островський А. Я. Методи та засоби створення і розгортання інфраструктури комп'ютерних систем за допомогою генеративних алгоритмів машинного навчання : магістерська робота / Тернопільський національний технічний університет імені Івана Пулюя. Тернопіль, 2023. URL: https://elartu.tntu.edu.ua/bitstream/lib/43332/2/Andrii_Ostrivskyi.pdf (дата звернення: 15.11.2024).

53. Class Activation Mapping (CAM): Better Interpretability in Deep Learning Models. URL: https://zilliz.com/learn/class-activation-mapping-CAM (дата звернення: 15.11.2024).

54. What is the Grad CAM method? URL: https://datascientest.com/en/what-is-the-grad-cam-method (дата звернення: 15.11.2024).

55. 9th New Trends in Image Restoration and Enhancement Workshop and Challenges. URL: https://openaccess.thecvf.com/CVPR2024_workshops/NTIRE (дата звернення: 15.11.2024).

56. The 2018 PIRM Challenge on Perceptual Image Super-resolution. URL: https://arxiv.org/abs/1809.07517 (дата звернення: 15.11.2024).

57. Let’s build from here, together. URL: https://www.github.careers/careers-home (дата звернення: 15.11.2024).

58. Дерій М. В. Алгоритми зменшення простору ознак при класифікації біомедичних зображень. Тернопіль, 2018. URL:

http://dspace.wunu.edu.ua/bitstream/316497/28030/1/Дерій.pdf (дата звернення: 15.11.2024).

59. Python is a programming language that lets you work quickly and integrate systems more effectively. URL: https://www.python.org/ (дата звернення: 15.11.2024).

60. Django makes it easier to build better web apps more quickly and with less code. URL: https://www.djangoproject.com/ (дата звернення: 15.11.2024).

61. An end-to-end platform for machine learning. URL: https://www.tensorflow.org/ (дата звернення: 15.11.2024).

62. PyTorch. URL: https://pytorch.org/ (дата звернення: 15.11.2024).

63. PyCharm Official. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 15.11.2024).

64. Sphinx. URL: https://www.sphinx-doc.org/en/master/ (дата звернення: 15.11.2024).

65. MkDocs. URL: https://www.mkdocs.org/ (дата звернення: 15.11.2024).

66. reStructuredText Primer. URL: https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html (дата звернення: 15.11.2024).

67. What is EfficientNet? The Ultimate Guide. URL: https://blog.roboflow.com/what-is-efficientnet/ (дата звернення: 15.11.2024).

68. Шемаєв Д. І. Розроблення застосунку для генерування тестових наборів даних за допомогою нейромереж GAN. Харків, 2023. URL: https://openarchive.nure.ua/entities/publication/9345e790-51e7-41c5-a546-ab00bdceed46 (дата звернення: 15.11.2024)

# ДОДАТКИ

## ДОДАТОК А

```
import numpy as np
import torch
import torch.nn.functional as F
from einops import rearrange, repeat
from ldm.models.autoencoder import (AutoencoderKL, IdentityFirstStage,
                                    VQModelInterface)
from ldm.modules.diffusionmodules.util import (extract_into_tensor,
                                               make_beta_schedule, noise_like)
from ldm.modules.encoders.modules import AbstractEncoder
from ldm.util import (count_params, default, exists, instantiate_from_config,
                      isimage, ismap, log_txt_as_img, mean_flat)
from torch.optim.lr_scheduler import CosineAnnealingLR
from torchvision.utils import make_grid
from transformers import CLIPFeatureExtractor, CLIPModel,
CLIPTokenizer

from stable_diffusion.ldm.models.diffusion.ddpm_edit import
LatentDiffusion
from stable_diffusion.ldm.util import default

artifact_category = [
  'noise', 'blur', 'rain', 'underexposure', 'haze', 'low resolution',
  'raindrop', 'no'
]

def cliploss(p, g):
  loss = 0
  for i in range(p.size(1)):
    p_i = p[:, i]
```

Рисунок А.1 – Реалізація коду 1

```
        g_i = g[:, i]
        g_i = g_i.view(-1, 1)
        p_i = p_i.view(-1, 1)
        loss_i = torch.sqrt(p_i * g_i + 1e-8)
        loss = loss + loss_i
    loss = 1 - loss
    loss = loss / p.size(1)
    return torch.mean(loss)


def predict_logit(x, text, gt_logits, model):
    batch_size = x.size(0)
    num_patch = 1

    logits_per_image, full_x = model.forward(x, text)

    logits_per_image = logits_per_image.t()

    logits_per_image = logits_per_image.view(batch_size, num_patch, -1)

    logits_per_image = logits_per_image.mean(1)

    logits_distortion_gumbel = F.softmax(logits_per_image, dim=1)

    text_distortion = torch.mm(gt_logits, full_x.reshape(8, -1))

    text_arg = text_distortion.reshape(batch_size, 77, 768)

    return logits_distortion_gumbel, text_arg
```

Рисунок А.2 – Реалізація коду 2

```
class CLIPEmbedder(AbstractEncoder):
    """Uses the CLIP transformer encoder for text (from Hugging Face)"""

    def __init__(self,
                 version="openai/clip-vit-large-patch14",
                 device="cuda",
                 max_length=77):
        super().__init__()
        self.tokenizer = CLIPTokenizer.from_pretrained(version)
        self.clipmodel = CLIPModel.from_pretrained(version)
        self.device = device
        self.max_length = max_length
        self.processor = CLIPFeatureExtractor.from_pretrained(version)
        self.clipmodel.eval()

    def forward(self, image, text):
        tokens = self.tokenizer(text,
                                truncation=True,
                                max_length=self.max_length,
                                return_length=False,
                                return_overflowing_tokens=False,
                                padding="max_length",
                                return_tensors="pt")

        image = (image + 1) * 255. / 2
        images = self.processor(images=image,
                                do_resize=False,
                                return_tensors="pt")
        tokens.data['pixel_values'] = images.data['pixel_values']
```

Рисунок А.3 – Реалізація коду 3

```
        for k in tokens:
            if hasattr(tokens[k], 'device'):
                tokens[k] = tokens[k].to(self.device)
        outputs = self.clipmodel(**tokens)

        logits_per_text = outputs.logits_per_text
        text_outputs = outputs.text_model_output.last_hidden_state
        return logits_per_text.float(), text_outputs


def disabled_train(self, mode=True):
    """Overwrite model.train with this function to make sure train/eval mode
    does not change anymore."""
    return self


from NAFNet.basicsr.models.archs.NAFNet_arch import NAFNet

class NAFNet_Combine(NAFNet):
    def forward(self, inp):
        B, C, H, W = inp.shape
        inp = self.check_image_size(inp)

        x = self.intro(inp)

        encs = []

        for encoder, down in zip(self.encoders, self.downs):
            x = encoder(x)
            encs.append(x)
```

Рисунок А.4 – Реалізація коду 4

```
            x = down(x)

        x = self.middle_blks(x)

        for decoder, up, enc_skip in zip(self.decoders, self.ups, encs[::-1]):
            x = up(x)
            x = x + enc_skip
            x = decoder(x)

        x = self.ending(x)

        # weight of SCM
        weight = 1
        x = x * weight + inp[:, 3:6, :, :]

        return x[:, :, :H, :W]


class CycleLatentDiffusion(LatentDiffusion):

    def __init__(self,  *args, **kwargs):
        super().__init__(if_init_cond=False, *args, **kwargs)
        self.model_assessment = CLIPEmbedder(device=self.device)
        self.NAFNet = NAFNet_Combine(img_channel=6,
                                     width=64,
                                     enc_blk_nums=[2, 2, 4, 8],
                                     middle_blk_num=12,
                                     dec_blk_nums=[2, 2, 2, 2])

        for name, para in self.model_assessment.named_parameters():
```

Рисунок А.5 – Реалізація коду 5

```
        ])
        if self.use_scheduler:
            assert 'target' in self.scheduler_config

            print("Setting up LambdaLR scheduler...")
            scheduler = [{
                'scheduler':
                CosineAnnealingLR(opt, T_max=4000, eta_min=1e-7),
                'interval':
                'step',
                'frequency':
                1
            }]
            return [opt], scheduler
        return opt

    def training_step(self, batch, batch_idx):

        loss, loss_dict = self.shared_step(batch)

        self.log_dict(loss_dict,
                      prog_bar=True,
                      logger=True,
                      on_step=True,
                      on_epoch=True)

        self.log("global_step",
                 self.global_step,
                 prog_bar=True,
                 logger=True,
```

Рисунок А.6 – Реалізація коду 6

```
                on_step=True,
                on_epoch=False)

        if self.use_scheduler:
            lr = self.optimizers().param_groups[0]['lr']
            self.log('lr_abs',
                     lr,
                     prog_bar=True,
                     logger=True,
                     on_step=True,
                     on_epoch=False)

        return loss

    def shared_step(self, batch):
        x, c, clip_loss, xc, x_edited = self.get_input(
            batch,
            self.first_stage_key,
            return_original_edited=True,
            return_original_cond=True,
            if_cliploss=True)
        loss = self(x, c, clip_loss, x_edited, xc)
        return loss

    def forward(self, x, c, clip_loss, x_edited, xc, *args, **kwargs):
        # b, c, h, w, device, img_size, = *x.shape, x.device, self.image_size
        # assert h == img_size and w == img_size, f'height and width of image
must be {img_size}'
        t = torch.randint(0,
                          self.num_timesteps, (x.shape[0], ),
```

Рисунок А.7 – Реалізація коду 7

```
                    device=self.device).long()
        if self.model.conditioning_key is not None:
            assert c is not None
            if self.cond_stage_trainable:
                c = self.get_learned_conditioning(c)
            if self.shorten_cond_schedule:  # TODO: drop this option
                tc = self.cond_ids[t].to(self.device)
                c = self.q_sample(x_start=c,
                                  t=tc,
                                  noise=torch.randn_like(c.float()))

        return self.p_losses(x, c, x_edited, t, clip_loss, xc, *args, **kwargs)

    def get_input(self,
                  batch,
                  k,
                  return_first_stage_outputs=False,
                  force_c_encode=False,
                  cond_key=None,
                  return_original_cond=False,
                  bs=None,
                  return_original_edited=False,
                  uncond=0.05,
                  if_cliploss=False):
        self.model_assessment.device = self.device
        with torch.no_grad():
            x = batch[k]
            if bs is not None:
                x = x[:bs]
            x = x.to(self.device)
```

Рисунок А.8 – Реалізація коду 8

```
        encoder_posterior = self.encode_first_stage(x)
        z = self.get_first_stage_encoding(encoder_posterior).detach()
        cond_key = cond_key or self.cond_stage_key
        xc = batch[cond_key]
        if bs is not None:
            # xc["c_crossattn"] = xc["c_crossattn"][:bs]
            xc["c_concat"] = xc["c_concat"][:bs]
        cond = {}

        # To support classifier-free guidance, randomly drop out only text conditioning 5%, only image conditioning 5%, and both 5%.
        random = torch.rand(x.size(0), device=x.device)
        input_mask = 1 - rearrange(
            (random >= uncond).float() *
            (random < 3 * uncond).float(), "n -> n 1 1 1")
        cond["c_concat"] = [
            input_mask * self.encode_first_stage(
                (xc["c_concat"].to(self.device))).mode().detach()
        ]
    joint_texts = [
        f"A photo needs {d} artifact reduction" for d in dists_map
    ]
    edit_x = batch['edit']['c_concat']
    gt_logits = []
    for bs in range(len(batch['edited'])):
        if 'noise' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([1., 0., 0., 0., 0., 0., 0., 0.])
        elif 'blur' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([0., 1., 0., 0., 0., 0., 0., 0.])
        elif 'rain' in batch['edit']['c_crossattn'][bs].split(' '):
```

Рисунок А.9 – Реалізація коду 9

```
            gt_logits.append([0., 0., 1., 0., 0., 0., 0., 0.])
        elif 'underexposure' in batch['edit']['c_crossattn'][bs].split(
                ' '):
            gt_logits.append([0., 0., 0., 1., 0., 0., 0., 0.])
        elif 'haze' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([0., 0., 0., 0., 1., 0., 0., 0.])
        elif 'resolution' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([0., 0., 0., 0., 0., 1., 0., 0.])
        elif 'raindrop' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([0., 0., 0., 0., 0., 0., 1., 0.])
        elif 'no' in batch['edit']['c_crossattn'][bs].split(' '):
            gt_logits.append([0., 0., 0., 0., 0., 0., 0., 1.])
    gt_logits = torch.FloatTensor(gt_logits).to(self.device)
    logits_per_image, text_arg_feature = predict_logit(
        edit_x, joint_texts, gt_logits, self.model_assessment)
    clip_loss = cliploss(logits_per_image, gt_logits)
    random = torch.rand(x.size(0), device=x.device)
    prompt_mask = rearrange(random < 2 * 0.05, "n -> n 1 1")
    null_prompt_tokens = self.model_assessment.tokenizer(
        "",
        truncation=True,
        max_length=77,
        return_length=False,
        return_overflowing_tokens=False,
        padding="max_length",
        return_tensors="pt").to(self.device)
    null_prompt_feature = self.model_assessment.clipmodel.text_model(
        **null_prompt_tokens).last_hidden_state
    cond["c_crossattn"] = [
        torch.where(prompt_mask, null_prompt_feature,
```

Рисунок А.10 – Реалізація коду 10

```
                text_arg_feature.float())
        ]
        if if_cliploss:
            out = [z, cond, clip_loss]
        else:
            out = [z, cond]
        if return_first_stage_outputs:
            with torch.no_grad():
                xrec = self.decode_first_stage(z)
                out.extend([x, xrec])
        if return_original_cond:
            out.append(xc)
        if return_original_edited:
            out.append(batch[k])
        return out

    def q_sample(self, x_start, t, noise=None):
        noise = default(noise, lambda: torch.randn_like(x_start))
        x_start_alpha = extract_into_tensor(self.sqrt_alphas_cumprod, t,
                                            x_start.shape)
        noise_beta = extract_into_tensor(self.sqrt_one_minus_alphas_cumprod,
t,
                                         x_start.shape)
        return (x_start_alpha * x_start +
                noise_beta * noise), x_start_alpha, noise_beta

    def p_losses(self, x_start, cond, x_edited, t, clip_loss, xc, noise=None):
        noise = default(noise, lambda: torch.randn_like(x_start))
        x_noisy, x_start_alpha, noise_beta = self.q_sample(x_start=x_start,
                                                           t=t,
```

Рисунок А.11 – Реалізація коду 11

```
                                    noise=noise)

model_output = self.apply_model(x_noisy, t, cond)
model_recon = self.decode_first_stage(
   (x_noisy - noise_beta * model_output) / x_start_alpha)
recon = x_edited

loss_dict = {}
prefix = 'train' if self.training else 'val'

recon_stack = torch.cat((xc['c_concat'], model_recon), dim=1)
result = self.NAFNet(recon_stack)
if self.parameterization == "x0":
   target = x_start
elif self.parameterization == "eps":
   target = noise
else:
   raise NotImplementedError()

loss_dict.update({f'{prefix}/clip_loss': clip_loss.mean()})

loss_simple = self.get_loss(model_output, target,
                            mean=False).mean([1, 2, 3])
loss_dict.update({f'{prefix}/loss_simple': loss_simple.mean()})

loss_final = self.get_loss(result, recon, mean=False).mean([1, 2, 3])
loss_dict.update({f'{prefix}/loss_final': loss_final.mean()})

logvar_t = self.logvar[t].to(self.device)
loss = loss_simple / torch.exp(logvar_t) + logvar_t
```

Рисунок А.12 – Реалізація коду 12

```
        if self.learn_logvar:
            loss_dict.update({f'{prefix}/loss_gamma': loss.mean()})
            loss_dict.update({'logvar': self.logvar.data.mean()})

        loss = self.l_simple_weight * loss.mean()

        loss_vlb = self.get_loss(model_output, target,
                                 mean=False).mean(dim=(1, 2, 3))
        loss_vlb = (self.lvlb_weights[t] * loss_vlb).mean()
        loss_dict.update({f'{prefix}/loss_vlb': loss_vlb})
        loss += (self.original_elbo_weight * loss_vlb)
        loss += clip_loss
        # loss += loss_NafNet.mean()
        loss += loss_final.mean()
        loss_dict.update({f'{prefix}/loss': loss})

        return loss, loss_dict

    def log_images(self,
                   batch,
                   N=4,
                   n_row=4,
                   sample=True,
                   ddim_steps=200,
                   ddim_eta=1.,
                   return_keys=None,
                   quantize_denoised=True,
                   inpaint=False,
                   plot_denoise_rows=False,
                   plot_progressive_rows=False,
```

Рисунок А.13 – Реалізація коду 13

```
        plot_diffusion_rows=False,
        **kwargs):

use_ddim = False

log = dict()
z, c, x, xrec, xc, edited = self.get_input(
    batch,
    self.first_stage_key,
    return_first_stage_outputs=True,
    force_c_encode=True,
    return_original_cond=True,
    return_original_edited=True,
    bs=N,
    uncond=0)
N = min(x.shape[0], N)
n_row = min(x.shape[0], n_row)
log["inputs"] = x
log["reals"] = xc["c_concat"]
log["reconstruction"] = xrec
log["gt"] = edited
if self.model.conditioning_key is not None:
    if hasattr(self.cond_stage_model, "decode"):
        xc = self.cond_stage_model.decode(c)
        log["conditioning"] = xc
    elif self.cond_stage_key in ["caption"]:
        xc = log_txt_as_img((x.shape[2], x.shape[3]), batch["caption"])
        log["conditioning"] = xc
    elif self.cond_stage_key == 'class_label':
        xc = log_txt_as_img((x.shape[2], x.shape[3]),
```

Рисунок А.14 – Реалізація коду 14

```
                    batch["human_label"])
            log['conditioning'] = xc
        elif isimage(xc):
            log["conditioning"] = xc
        if ismap(xc):
            log["original_conditioning"] = self.to_rgb(xc)

    if plot_diffusion_rows:
        # get diffusion row
        diffusion_row = list()
        z_start = z[:n_row]
        for t in range(self.num_timesteps):
            if t % self.log_every_t == 0 or t == self.num_timesteps - 1:
                t = repeat(torch.tensor([t]), '1 -> b', b=n_row)
                t = t.to(self.device).long()
                noise = torch.randn_like(z_start)
                z_noisy = self.q_sample(x_start=z_start, t=t, noise=noise)
                diffusion_row.append(self.decode_first_stage(z_noisy))

        diffusion_row = torch.stack(
            diffusion_row)  # n_log_step, n_row, C, H, W
        diffusion_grid = rearrange(diffusion_row, 'n b c h w -> b n c h w')
        diffusion_grid = rearrange(diffusion_grid,
                                   'b n c h w -> (b n) c h w')
        diffusion_grid = make_grid(diffusion_grid,
                                   nrow=diffusion_row.shape[0])
        log["diffusion_row"] = diffusion_grid

    if sample:
        # get denoise row
```

Рисунок А.15 – Реалізація коду 15

```
with self.ema_scope("Plotting"):
    samples, z_denoise_row = self.sample_log(cond=c,
                                             batch_size=N,
                                             ddim=use_ddim,
                                             ddim_steps=ddim_steps,
                                             eta=ddim_eta)
x_samples = self.decode_first_stage(samples)
log["samples"] = x_samples

with torch.no_grad():
    stack = torch.cat((xc["c_concat"], x_samples), dim=1)
    recon_final = self.NAFNet(stack)
log["recon_final"] = recon_final

if plot_denoise_rows:
    denoise_grid = self._get_denoise_row_from_list(z_denoise_row)
    log["denoise_row"] = denoise_grid

if quantize_denoised and not isinstance(
        self.first_stage_model, AutoencoderKL) and not isinstance(
            self.first_stage_model, IdentityFirstStage):
    # also display when quantizing x0 while sampling
    with self.ema_scope("Plotting Quantized Denoised"):
        samples, z_denoise_row = self.sample_log(
            cond=c,
            batch_size=N,
            ddim=use_ddim,
            ddim_steps=ddim_steps,
            eta=ddim_eta,
            quantize_denoised=True)
```

Рисунок А.16 – Реалізація коду 16

```
        x_samples = self.decode_first_stage(samples.to(self.device))
        log["samples_x0_quantized"] = x_samples

    if inpaint:
        # make a simple center square
        b, h, w = z.shape[0], z.shape[2], z.shape[3]
        mask = torch.ones(N, h, w).to(self.device)
        # zeros will be filled in
        mask[:, h // 4:3 * h // 4, w // 4:3 * w // 4] = 0.
        mask = mask[:, None, ...]
        with self.ema_scope("Plotting Inpaint"):
            samples, _ = self.sample_log(cond=c,
                                         batch_size=N,
                                         ddim=use_ddim,
                                         eta=ddim_eta,
                                         ddim_steps=ddim_steps,
                                         x0=z[:N],
                                         mask=mask)
        x_samples = self.decode_first_stage(samples.to(self.device))
        log["samples_inpainting"] = x_samples
        log["mask"] = mask

        # outpaint
        with self.ema_scope("Plotting Outpaint"):
            samples, _ = self.sample_log(cond=c,
                                         batch_size=N,
                                         ddim=use_ddim,
                                         eta=ddim_eta,
                                         ddim_steps=ddim_steps,
                                         x0=z[:N],
```

Рисунок А.17 – Реалізація коду 17

```
                                    mask=mask)
            x_samples = self.decode_first_stage(samples.to(self.device))
            log["samples_outpainting"] = x_samples

        if plot_progressive_rows:
            with self.ema_scope("Plotting Progressives"):
                img, progressives = self.progressive_denoising(
                    c,
                    shape=(self.channels, self.image_size, self.image_size),
                    batch_size=N)
            prog_row = self._get_denoise_row_from_list(
                progressives, desc="Progressive Generation")
            log["progressive_row"] = prog_row

        if return_keys:
            if np.intersect1d(list(log.keys()), return_keys).shape[0] == 0:
                return log
            else:
                return {key: log[key] for key in return_keys}
        return log

    @torch.no_grad()
    def validation_step(self, batch, batch_idx):
        _, loss_dict_no_ema = self.shared_step(batch)
        with self.ema_scope():
            _, loss_dict_ema = self.shared_step(batch)
            loss_dict_ema = {
                key + '_ema': loss_dict_ema[key]
                for key in loss_dict_ema
            }
```

Рисунок А.18 – Реалізація коду 18

**ДЕКЛАРАЦІЯ**

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

_____

*Повністю вказується ПІБ та статус ( освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)*

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

**ЗОБОВ'ЯЗУЮСЬ:**

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

**ПІДТВЕРДЖУЮ:**

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

**УСВІДОМЛЮЮ:**

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)