МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Зварич Роман Олексійович

«Алгоритми попереднього опрацювання зображень з використанням хмарних обчислень / Image preprocessing algorithms using cloud computing»

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІм-21
Р.О. Зварич
_____

Науковий керівник:
к.т.н., доц. О. Й. Піцун
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____Л. О. Дубчак

ТЕРНОПІЛЬ – 2024

АНОТАЦІЯ

Зварич Р. О. Алгоритми попереднього опрацювання зображень з використанням хмарних обчислень. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2024.

Робота написана обсягом 71 сторінка і містить 28 ілюстрацій, , 3 додатки та 42 джерела за переліком посилань. Метою кваліфікаційної роботи є розробка підходів до розгортання програмного забезпечення для опрацювання зображень на хмарних сервісах.

Методи досліджень. Для розв'язання поставлених задач у кваліфікаційній роботі використано: методи: лінійної алгебри та аналітичної геометрії (для створення алгоритмів адаптивного опрацювання зображень); об'єктно-орієнтованого програмування (для проектування програмних засобів).

Результати дослідження: алгоритми адаптивної обробки зображень на низькому рівні комп'ютерного зору, алгоритми автоматичної сегментації зображень, алгоритм розгортання програмного забезпечення в хмарному середовищі для підвищення ефективності опрацювання та спрощення процесу обробки. Адаптивність розроблених алгоритмів грунтується на використанні вхідних параметрів зображень та сформованій в нечіткій формі правил їх опрацювання.

Результати цього дослідження можуть стати основою для покращення якості опрацювання зображень з використання хмарних сервісів не залежно від типу провайдера.

Орієнтовні напрямки розвитку досліджень: розробка спеціалізованих засобів у вигляді веб-сервісів з використанням хмарних технологій для опрацювання зображень.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, АНАЛІЗ, ЗОБРАЖЕННЯ, ПОПЕРЕДНЯ ОБРОБКА, ХМАРНІ СЕРВІСИ.

ANNOTATION

Zvarych R. O. Algorithms of pre-processing of images using cloud computing. – Manuscript.

Qualification work for obtaining the degree of "Master" in specialty 123 "Computer Engineering", educational and professional program. Western Ukrainian National University, Ternopil, 2024.

The work is written in 71 pages and contains 28 illustrations, 7 appendices and 42 sources according to the list of references. The purpose of the qualification work is to develop approaches to deploying software for image processing on cloud services.

Research methods. To solve the tasks set in the qualification work, the following methods were used: linear algebra and analytical geometry (for creating adaptive image processing algorithms); object-oriented and functional programming (for designing software tools).

Research results: adaptive image processing algorithms at a low level of computer vision, automatic image segmentation algorithms, software deployment algorithm in a cloud environment to increase processing efficiency and simplify the processing process. The adaptability of the developed algorithms is based on the use of input image parameters and fuzzy processing rules.

The results of this study can become the basis for improving the quality of image processing using cloud services, regardless of the type of provider.

Approximate research development directions: development of specialized tools in the form of web services using cloud technologies for image processing.

KEYWORDS: ALGORITHM, ANALYSIS, IMAGE, PRE-PROCESSING, CLOUD SERVICES.

ЗМІСТ

ВСТУП 9
1. Аналіз алгоритмів опрацювання зображень 12
1.1 Алгоритми попередньої обробки зображень 12
1.2 Алгоритми сегментації зображень 16
1.3 Аналіз засобів інтеграції програмного коду на хмарних сервісах 22
1.4 Аналіз завдання магістерської роботи та постановка задач дослідження 26
2. Алгоритми опрацювання зображень на хмарних сервісах 27
2.1 Адаптивний алгоритм попереднього оброблення зображень 27
2.2 Адаптивний алгоритм сегментації зображень 33
2.3 Алгоритм розгортання ПЗ на хмарному сервісі 38
2.4 Висновки до розділу 2 40
3. Програмна реалізація на хмарному сервісі 41
3.1 Структура програмного модуля опрацювання зображень 41
3.2 Реалізація CI/CD 47
3.3 Порівняльний аналіз результатів 51
3.4 Висновки до розділу 55
Висновки 56
Список використаних джерел 57
Додаток А Лістинг коду програми для обробки зображень ..... **Помилка! Закладку не визначено.**
Додаток Б Лістинг коду програми для розгортання ПЗ .... **Помилка! Закладку не визначено.**
Додаток В Світлокопії виданих публікацій ..... **Помилка! Закладку не визначено.**

ВСТУП

У сучасному світі інформаційних технологій обробка зображень відіграє важливу роль у багатьох сферах, включаючи медицину, промисловість, автоматизацію, а також у повсякденному житті. Зображення є основним джерелом даних у багатьох застосуваннях, тому їхнє попереднє опрацювання стає невід'ємною частиною аналізу та візуалізації[1-2]. Однією з ключових задач попереднього опрацювання є підвищення якості зображень, усунення шумів, корекція кольорів і контрасту, що сприяє покращенню точності подальшого аналізу.

В умовах швидкого зростання обсягу даних та потреби в потужних обчислювальних ресурсах, традиційні підходи до обробки зображень часто виявляються недостатніми. Хмарні технології пропонують нові можливості для вирішення цих проблем, надаючи доступ до потужних обчислювальних ресурсів, які дозволяють виконувати складні алгоритми обробки зображень на великих обсягах даних[3]. Використання хмари для обробки зображень забезпечує гнучкість, масштабованість та економічність, що особливо важливо для організацій, які прагнуть зменшити витрати на інфраструктуру.
Мета даного дипломного проекту полягає у дослідженні та розробці алгоритмів попереднього опрацювання зображень, що використовують хмарні технології. У процесі виконання проекту буде проведено аналіз існуючих методів обробки зображень, розглянуто переваги та недоліки хмарних рішень, а також реалізовано практичний приклад застосування розроблених алгоритмів у середовищі хмари.

Результати цього дослідження можуть мати важливе значення для подальшого розвитку технологій обробки зображень і їх інтеграції у

різноманітні галузі, що, в свою чергу, сприятиме підвищенню ефективності роботи з даними та їх аналізу.

Хмарні технології пропонують нові можливості для ефективного опрацювання зображень, дозволяючи користувачам отримати доступ до потужних обчислювальних ресурсів без необхідності в значних інвестиціях в апаратне забезпечення[4-5]. Завдяки хмарам, обробка зображень може відбуватися в реальному часі, що забезпечує високу продуктивність та масштабованість. Крім того, хмарні рішення дозволяють зберігати великі обсяги даних та забезпечують гнучкість у виборі інструментів для їх обробки.

Метою кваліфікаційної роботи є розробка підходів до розгортання програмного забезпечення для опрацювання зображень на хмарних сервісах.

Для виконання мети потрібно вирішити наступні задачі:

- проаналізувати алгоритми попереднього оброблення зображень;
- проаналізувати алгоритми сегментації зображень;
- розробити адаптивний алгоритм попереднього оброблення зображень;
- розробити адаптивний алгоритм сегментації зображень;
- програмно реалізувати механізм безперервної доставки коду на хмарне середовище для опрацювання зображень;
- провести порівняльний аналіз створеної системи з іншими онлайн-платформами;

Об’єкт дослідження – процеси розгортання програмного забезпечення в хмарному середовищі.

Предмет дослідження – життєвий шлях розгортання програмного забезпечення з елементами безперервної доставки та безперервної інтеграції програмного коду.

Наукова новизна полягає у розробці алгоритму адаптивного опрацювання зображень та його універсального розгортання на хмарних сервісах для підвищення продуктивності.

Результати кваліфікаційної роботи призначені для використання в навчальному процесі та для проведення наукових досліджень в галузі опрацювання зображень.

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків.

У першому розділі було проаналізовано існуючі алгоритми попереднього оброблення зображень та сегментації, а також досліджено інструменти для розгортання програмного забезпечення у хмарних сервісах.

У другому розділі були наведені алгоритмічні розробки для опрацювання зображень на низькому та середньому рівнях комп'ютерного зору.

У третьому розділі були програмно реалізований модуль для розгортання комплексу для опрацювання зображень у хмарних сервісах з використанням технологій CI/CD.

Результати роботи знайшли своє відображення у роботі Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів «Інтелектуальні комп'ютерні системи та мережі» (Тернопіль, Україна, 5 листопада 2024 року).

1. АНАЛІЗ АЛГОРИТМІВ ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ

1.1 Алгоритми попередньої обробки зображень

У цифровій формі зображення розглядають як набір даних, які можна математично аналізувати та обробляти. У контексті комп'ютерного зору зображення є основним джерелом вхідної інформації для алгоритмів, що дозволяють машині "бачити" та "розуміти" сцени або об'єкти на ньому. Зображення представляється як матриця пікселів, де кожен піксель має певні значення кольору або яскравості. Кожен піксель може бути представлений значенням інтенсивності (для градацій сірого) або значеннями для кожного каналу кольору (для кольорових зображень, наприклад, RGB). Градації сірого - одноканальне зображення, де кожен піксель представлений значенням яскравості (зазвичай від 0 до 255)[6-7]. Кольорове зображення (RGB) - трьохканальне зображення, де кожен піксель має три значення, що відповідають інтенсивності червоного, зеленого та синього кольорів[8-10].

Зображення можуть бути представлені не лише в RGB, але й у колірних просторах, які краще підходять для обробки та аналізу (наприклад, HSV, Lab). HSV виділяє колір, насиченість та яскравість, що зручно для обробки зображень при зміні рівня світла.

Піксельний рівень використовується при базових операціях, таких як фільтрування або обчислення яскравості[11].

Рівень ознак виділяються ключові точки, крапки або кути, які описують важливі деталі зображення. Наприклад, використовується для розпізнавання об’єктів або шаблонів[12].

Рівень сегментації зображення поділяється на області, що представляють окремі об'єкти або фон[13].

Для машинного навчання зображення часто трансформують у вектори ознак, які представляють важливі деталі зображення, придатні для

класифікації чи розпізнавання. Для цього використовують дескриптори, такі як HOG (Histogram of Oriented Gradients) або методи глибинного навчання, які автоматично витягують ознаки.

Приклад виконання медіанної фільтрації наведено на рисунку 1.1.

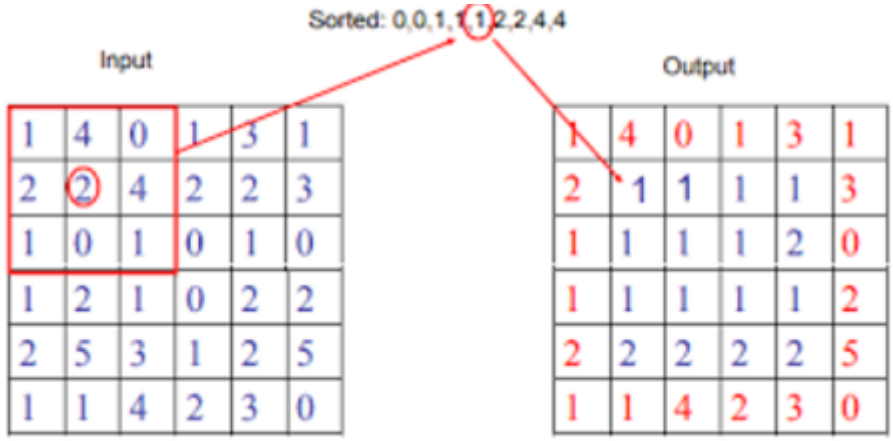


Рисунок 1.1 – Приклад виконання медіанної фільтрації

Сучасні методи комп'ютерного зору часто використовують нейронні мережі, зокрема згорткові нейронні мережі (CNN), що витягують високорівневі ознаки[14-15]. Тут зображення перетворюється у набір багатовимірних тензорів, де кожен рівень мережі представляє зображення з різним рівнем абстракції. Порівняльний аналіз алгоритм попереднього оброблення зображень наведено у таблиці 1.

Таблиця 1.1 – Порівняльний аналіз алгоритм попереднього оброблення зображень

Алгоритм	Призначення	Основні особливості	Переваги	Недоліки
Unsharp Masking	Збільшення різкості шляхом виділення країв	Використовує розмиту версію зображення для виділення контурів, що додаються до оригіналу	Простий, швидкий, ефективний для підвищення різкості	Може створювати шум і артефакти на контрастних краях
Median Filter	Зменшення шуму без втрати чіткості	Заміщує значення кожного пікселя на медіану його сусідів	Ефективний для видалення імпульсного шуму	Зменшує різкість зображення, не підходить для

			(соляно-перцевого)	дрібних деталей
Gaussian Blur	Зменшення шуму	Використовує гауссове розмиття для згладжування зображення, що зменшує рівень шуму	Добре згладжує шум, контроль рівня розмиття	Знижує чіткість країв і дрібні деталі
Bilateral Filter	Зменшення шуму зі збереженням країв	Розмиття відбувається з урахуванням яскравості пікселів, що дозволяє зберігати контури	Ефективне шумозаглушення з збереженням країв	Високі обчислювальні витрати
Histogram Equalization	Покращення контрасту зображення	Розтягує або перерозподіляє значення інтенсивності, збільшуючи контраст, особливо в низькоконтрастних областях	Підвищує видимість деталей у темних або світлих областях	Може створювати шум і втрату деталей при надмірному покращенні
Adaptive Histogram Equalization (CLAHE)	Локальне підвищення контрасту	Поділяє зображення на області й виконує рівномірне підвищення контрасту в кожній області	Підходить для зображень з нерівномірним освітленням	Може створювати артефакти при неправильному налаштуванні
Deconvolution	Видалення розмиття	Використовує інверсні або неруйнівні методи для зменшення розмиття шляхом відновлення початкової структури	Підходить для усунення розмиття, зберігає деталі	Вимагає точних параметрів, чутливий до шуму

Фільтр Гауса зазвичай використовують для аналізу різного роду зображень. Також даний фільтр використовують для обробки двовимірних сигналів. Додатково даний фільтр застосовують для отримання Гаусової модуляції[16]. Даний тип модуляції також застосовують у системах стільникового зв’язку. Приклад 2D згортки наведено на рисунку 1.2.

1	2	1

*

2	3	3
3	5	5
4	4	6

=

	11	
	18	
	18	

Рисунок 1.2 - Приклад 2D згортки

Приклад використання Гаусового фільтру наведено на рисунку 1.3.

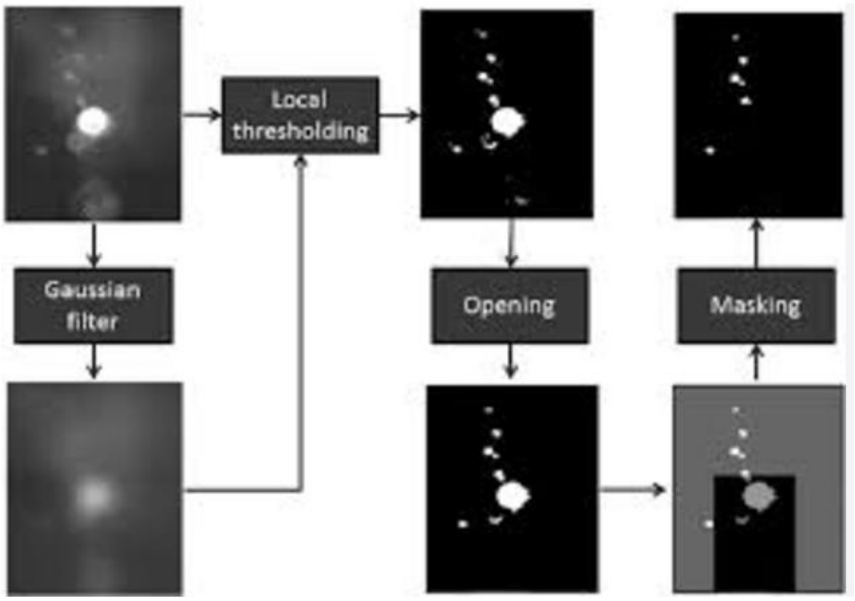


Рисунок 1.3 – Приклад використання Гаусового фільтру

У комп'ютерному зорі коригування яскравості та контрастності зображень є ключовими етапами обробки, які покращують видимість деталей і забезпечують якість для подальшого аналізу[17]. Основні підходи до коригування яскравості та контрастності включають як прості лінійні операції, так і більш складні методи, що враховують структуру зображення.

Приклад застосування різних параметрів для фільтру наведено на рисунку 1.4.

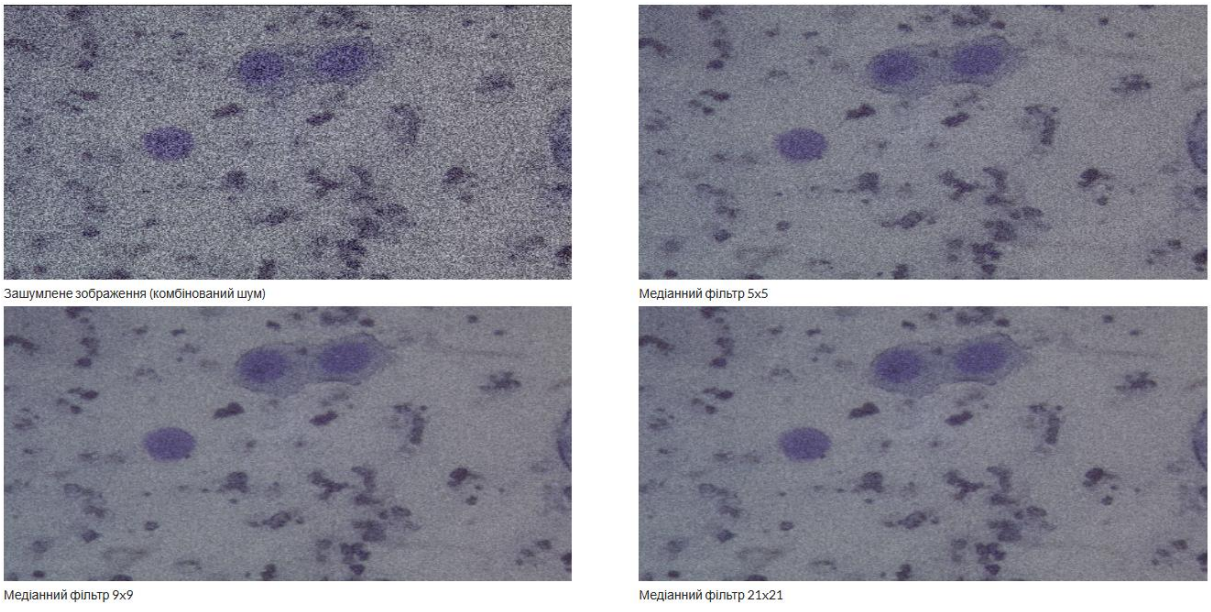


Рисунок 1.4 - Приклад застосування різних параметрів для фільтру

Глобальне вирівнювання гістограми забезпечує рівномірний розподіл яскравості пікселів, підкреслюючи темні та світлі області зображення. Цей підхід підвищує контрастність у зображеннях з низьким контрастом, розраховуючи кумулятивний розподіл гістограми та застосовуючи його для покращення видимості деталей[18-20].

Метод нормалізації за стандартним відхиленням (Standard Deviation Normalization) коригує інтенсивність зображення на основі значення стандартного відхилення. Контрастність можна регулювати шляхом зміни середнього значення і стандартного відхилення інтенсивностей[20-25].

Адаптивне вирівнювання гістограми розбиває зображення на невеликі області та вирівнює інтенсивності в кожній з них окремо. Цей метод обмежує надмірне підсилення контрасту, що допомагає зменшити шум і спотворення, характерні для звичайного вирівнювання гістограми. Його зазвичай застосовують до зображень з неоднорідним освітленням або до медичних знімків.

1.2 Алгоритми сегментації зображень

Сегментація зображень – це процес поділу цифрового зображення на декілька сегментів (областей), кожен з яких відповідає окремому об'єкту або частині об'єкта. Іншими словами, це як розрізати картинку на шматочки таким чином, щоб кожен шматочок представляв певну частину зображення.

Основні галузі застосування сегментації:

- медицина;
- автономні транспортні засоби;
- робототехніка;
- системи відеоспостереження;

- індустріальна автоматизація;
- доповнена реальність.

У медицині сегментація використовується для виділення мікрооб’єктів на зображенні для подальшого їхнього аналізу та оцінки. Використовується для аналізу МРТ, рентгенівських, цитологічних, гістологічних, імуногістохімічних та інших зображень[26].

У автономних транспортних засобах сегментація зображень використовується для розпізнавання об’єктів на дорозі, таких як транспорт, дорожні знати, люди тощо.

У робототехніці використовують для виділення об'єктів для маніпуляцій, наприклад, захоплення предметів роботом, маніпуляції та інші дії тощо[27].

У системах відеоспостереження використовують для відстеження рухомих об'єктів, розпізнавання осіб, ідентифікації. У галузі індустріальної автоматизації сегментацію зображень використовують для контролю якості продукції, дефектоскопія[28-29]. Приклади сегментації зображень наведено на рисунку 1.5.

Рисунок 1.5 – Приклади сегментації зображень

У галузі доповненої реальності сегментацію використовують для створення інтерактивних елементів у реальному світі. На рисунку 1.6 наведено приклади сегментації в застосунках доповненої реальності.

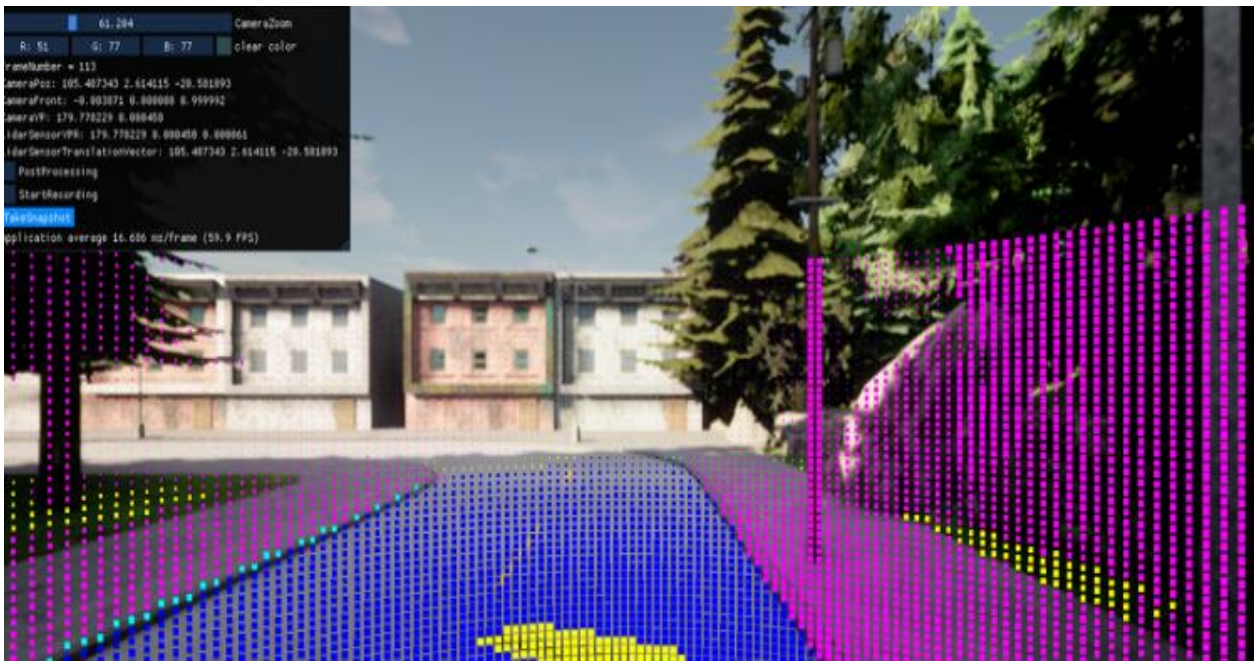


Рисунок 1.6 – Приклади сегментації в застосунках доповненої реальності

До основних завдань, які вирішує сегментація відносять виділення об’єктів, класифікацію пікселів, спрощення зображень, підготовка даних для машинного навчання. До виділення об’єктів відносять визначення меж і контурів об'єктів на зображенні, розпізнавання окремих елементів на зображенні (наприклад, особи на фото, клітини на мікроскопічному знімку)[30-33]. До класифікації пікселів відносять такі пункти як призначення кожному пікселю зображення певного класу (наприклад, "небо", "трава", "будинок"), створення семантичних масок, які показують, до якого класу належать різні частини зображення. Задача спрощення зображення за рахунок сегментації забезпечує зменшення кількості деталей для подальшої обробки, виділення найважливіших характеристик зображення[33-36]. Для підготовки даних до машинного навчання використовують сегментацію для створення масок для навчання моделей глибокого навчання та генерації навчальних даних для задач розпізнавання образів.

До основних методів сегментації відносять такі:

- засновані на порогах;
- засновані на регіонах;

- засновані на контурах;
- засновані на кластеризації;
- глибоке навчання.

Результати сегментації можуть бути використані для визначення типу об'єкта, спостереження за рухом об'єктів у часі, відновлення тривимірної моделі об'єкта, визначення текстурних характеристик об'єктів.

Сегментація заснована на виділенні країв – це один з найпоширеніших методів сегментації зображень. Його основна ідея полягає в тому, що межі між об'єктами на зображенні часто супроводжуються різкими змінами яскравості. Виявляючи ці зміни (краї), ми можемо визначити контури об'єктів і, таким чином, розділити зображення на окремі сегменти. Застосовують різні оператори для обчислення градієнтів зображення (наприклад, оператор Собеля, Превітта, Лапласа)[37]. Градієнт показує напрямок і величину найшвидшого зміни яскравості в точці зображення. Вибирають поріг, який розділяє пікселі на ті, що належать до краю, і ті, що не належать. Пікселі з високим значенням градієнта вважаються частиною краю. Переваги методу виділення країв є:

- простота;
- ефективність;
- чіткі контури;

Недоліки методу виділення країв є:

- чутливість до шуму, оскільки шум може призвести до появи помилкових країв;
- проблема розірваних контурів, оскільки часто виникають розриви в контурах, особливо в областях з низьким контрастом або складними формами об'єктів.
- нездатність розпізнати однорідні області, особливо якщо об'єкти мають однорідний колір або текстуру, то їх межі можуть бути не виявлені.

- Підвищенню точності в ідентифікації меж і областей: видалення шуму та покращення контрасту допомагають виділити об’єкти з мінімальними помилками.

Порівняльний аналіз алгоритмів сегментації наведено у таблиці 1.2.

Таблиця 1.2 - Порівняльний аналіз алгоритмів сегментації

Алгоритм	Метод	Призначення
Порогова сегментація (Thresholding)	Порогове значення для інтенсивності пікселів	Поділ зображення на фони і об’єкти на основі інтенсивності
Сегментація на основі розподілу (Region-based Segmentation)	Виділення областей з подібними характеристиками	Розширює області на основі подібності пікселів
Кластеризація K-середніх (K-means)	Кластеризація за яскравістю або кольором	Поділ зображення на K K кластерів на основі кольорових або текстурних характеристик
Watershed	Гідродинамічна модель	Визначення меж об’єктів на основі розподілу яскравості, схоже на рельєф або вододіл
Модель активних контурів (Active Contours, Snake)	Деформаційні криві, що змінюють форму	Виділення контурів об'єктів зображення через енергетичну мінімізацію
Сегментація на основі нейронних мереж (U-Net, Mask R-CNN)	Глибинне навчання	Сегментація складних сцен, таких як об'єкти в зображеннях з високою роздільною здатністю
Сегментація на основі суперпікселів	Сегментація на однорідні області	Групує пікселі в більш «грубі» структури з урахуванням схожості (наприклад, методом SLIC)

U-Net – це архітектура згорткової нейронної мережі, спеціально розроблена для задач сегментації зображень, особливо у біомедичній області.

Вона була представлена у 2015 році і з того часу стала стандартом де-факто для багатьох завдань сегментації. Архітектуру U-Net наведено на рисунку 1.7.

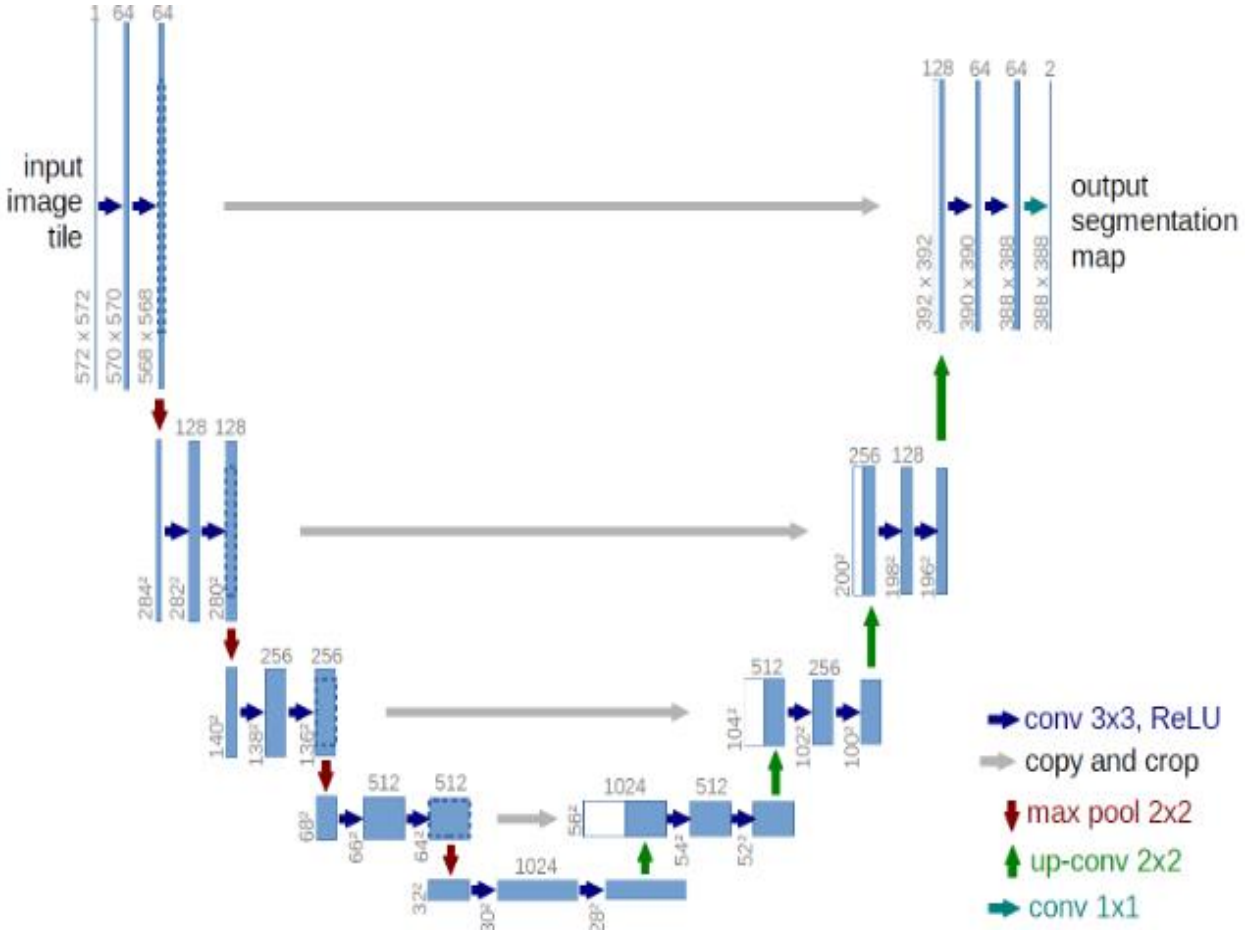


Рисунок 1.7 - Архітектура U-Net

Архітектура U-Net складається з двох основних частин: кодувальника та декодера. Кодувальник виконує стиснення зображення, поступово зменшуючи його просторову роздільну здатність, але збільшуючи кількість каналів (тобто витягуючи більш абстрактні ознаки)[38]. Декодер розширює зображення назад до вихідного розміру, відновлюючи просторові деталі та передбачаючи піксельну маску сегментації. З'єднання зі стрибками є ключовою особливістю U-Net, яка з'єднують відповідні рівні кодувальника та декодера[39]. Ці з'єднання дозволяють декодеру використовувати інформацію з нижчих рівнів, що покращує точність сегментації.

U-Net може ефективно навчатися на обмежених даних, що особливо важливо у медичній галузі. U-Net добре справляється із сегментацією об'єктів різних розмірів та форм. Завдяки з'єднанням зі стрибками U-Net точно визначає межі об'єктів.

1.3 Аналіз засобів інтеграції програмного коду на хмарних сервісах

Останніми роками хмарні сервіси набули значної популярності завдяки своїй здатності забезпечувати масштабованість, гнучкість і зниження витрат на інфраструктуру для бізнесу і розробників. Хмарні сервіси дозволяють організаціям зберігати та обробляти великі обсяги даних, надавати програмне забезпечення як послугу (SaaS) та створювати ефективні середовища для розробки і тестування.

Хмарні технології охоплюють різні моделі надання сервісів, такі як IaaS (Infrastructure as a Service), PaaS (Platform as a Service) і SaaS (Software as a Service), які задовольняють різноманітні потреби користувачів — від простого хостингу даних до комплексних платформ для розробки. Універсальність і адаптивність хмарних сервісів дозволяють підприємствам зосередитися на розвитку основної діяльності, а не на підтримці складної інфраструктури.

Послуги хмарних обчислень можна розділити на три основні категорії. Постачальники IaaS пропонують клієнтам можливість орендувати IT-інфраструктуру за потреби. IaaS включає всі основні будівельні блоки хмарних обчислень, такі як сховище, мережа та сервери. До популярних постачальників IaaS належать Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP).

Постачальники PaaS пропонують клієнтам можливість розробляти, запускати та керувати програмами на хмарній платформі. PaaS включає все необхідне для створення та запуску програми, наприклад веб-сервер, базу

даних та інструменти розробки. До популярних постачальників PaaS належать Heroku, AWS Elastic Beanstalk і Google App Engine.

Постачальники SaaS пропонують клієнтам можливість використовувати хмарну програмну програму. Додатки SaaS зазвичай доставляються через веб-браузер; клієнтам не потрібно встановлювати або керувати програмним забезпеченням. До популярних програм SaaS належать Google Docs, Microsoft Office 365 і Salesforce.

Порівняльний аналіз інструментів для DevOps наведено у таблиці 1.3.

Таблиця 1.3 - Порівняльний аналіз інструментів для DevOps

Інструмент	Основне призначення	Основні особливості
Jenkins	Автоматизація процесів CI/CD	Підтримка великої кількості плагінів, відкритий вихідний код, настроюваний пайплайн
GitLab CI/CD	Інтеграція, тестування і доставка кодів	Вбудовано в GitLab, має інтеграцію з репозиторіями коду, можливість паралельного виконання завдань
Docker	Контейнеризація додатків	Легка контейнеризація додатків, підтримка різних ОС, Docker Hub для розподілу контейнерів
Kubernetes	Оркестрація контейнерів у кластері	Автоматизація розгортання, масштабування та керування контейнерами, масштабованість

Terraform	Керування інфраструктурою через код	Система збору метрик, підтримка багатьох джерел, інструменти для створення дашбордів та сповіщень

Terraform – це інструмент з відкритим вихідним кодом, який використовується для управління інфраструктурою як кодом (IaC, Infrastructure as Code). Розроблений компанією HashiCorp, Terraform дозволяє визначати, розгортати і керувати інфраструктурою в різних хмарних середовищах та локальних середовищах за допомогою декларативної мови конфігурації, що робить управління великими та складними інфраструктурами значно простішим і зручнішим.

CI/CD (Continuous Integration/Continuous Delivery або Continuous Deployment) – це сукупність практик, які дозволяють автоматизувати процес розробки програмного забезпечення від написання коду до його розгортання. Цей підхід значно підвищує ефективність та якість розробки, а також дозволяє швидше доставляти продукт кінцевим користувачам. Автоматизація процесів збірки, тестування та розгортання дозволяє швидко інтегрувати нові зміни в код і доставляти їх користувачам. Це особливо важливо для проектів з швидкими темпами розробки. Завдяки регулярному тестуванню на кожній стадії розробки виявляються і усуваються помилки на ранніх етапах, що значно знижує ризик випуску дефектного продукту. Автоматизація рутинних завдань мінімізує людський фактор і зменшує ризик виникнення помилок під час ручного виконання операцій. Завдяки детальним логам і звітам про виконання конвеєру CI/CD, всі учасники проекту можуть відстежувати стан розробки та швидко виявляти проблеми. Автоматизація рутинних завдань дозволяє розробникам зосередитися на написанні коду та вирішенні більш складних задач.

Ключові етапи CI/CD:

- безперервна інтеграція (CI);
- безперервна доставка (CD);

- безперервне розгортання (CD) – автоматичне розгортання всіх змін в виробниче середовище після успішного проходження всіх стадій конвеєра.

Jenkins - один з найпопулярніших серверів безперервної інтеграції, що дозволяє автоматизувати процес збірки, тестування та розгортання програмного забезпечення.

Вбудована в GitLab система CI/CD, яка дозволяє легко налаштувати конвеєри доставки.

GitHub Actions вбудована в GitHub система CI/CD, що дозволяє автоматизувати робочі процеси без необхідності встановлення додаткового програмного забезпечення.

Docker -платформа для створення та управління контейнерами, що дозволяє упакувати додаток та всі його залежності в один образ.

Kubernetes - система оркестрації контейнерів, яка дозволяє автоматично розгортати, масштабувати та керувати контейнерними додатками.

Prometheus - система моніторингу, яка дозволяє збирати метрики з різних джерел і створювати дашборди для візуалізації даних.

Grafana - інструмент для створення інтерактивних дашбордів на основі даних, зібраних Prometheus та інших систем моніторингу.

Terraform – це потужний інструмент для управління інфраструктурою як кодом (IaC). Його основна мета – автоматизувати створення, налаштування та управління інфраструктурою в різних хмарних платформах та локальних дата-центрах. Terraform підтримує велику кількість провайдерів (AWS, Azure, GCP, DigitalOcean та багато інших), що дозволяє використовувати один інструмент для управління інфраструктурою в різних хмарних платформах. Конфігурації Terraform зберігаються у версійних системах (наприклад, Git), що дозволяє відстежувати зміни, повертатися до попередніх версій та співпрацювати в команді. Перед внесенням змін Terraform створює план виконання, який показує, які ресурси будуть створені, змінені або видалені. Це дозволяє перевірити, що всі зміни будуть виконані відповідно до ваших очікувань.

1.4 Аналіз завдання магістерської роботи та постановка задач дослідження

На основі аналітичного підходу було проведено аналіз алгоритмів попереднього оброблення зображень та сегментації, а також досліджено інструменти для розгортання програмного забезпечення у хмарних сервісах, використовуючи DevOps підходи.

Метою кваліфікаційної роботи є розробка підходів до розгортання програмного забезпечення для опрацювання зображень на хмарних сервісах.

Для виконання мети потрібно вирішити наступні задачі:

- проаналізувати алгоритми попереднього оброблення зображень;
- проаналізувати алгоритми сегментації зображень;
- розробити адаптивний алгоритм попереднього оброблення зображень;
- розробити адаптивний алгоритм сегментації зображень;
- програмно реалізувати механізм безперервної доставки коду на хмарне середовище для опрацювання зображень;
- провести порівняльний аналіз створеної системи з іншими онлайн-платформами;

2. АЛГОРИТМИ ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ НА ХМАРНИХ СЕРВІСАХ

2.1 Адаптивний алгоритм попереднього оброблення зображень

Недолік більшості систем опрацювання зображень полягає у відсутності механізмів попереднього оброблення зображень, що спричиняє наявність шумів, нечітких контурів, нерівної гістограми, пересвічування, затемнення певних областей тощо. З метою уникнення подібних ситуацій необхідно застосування спеціалізованих алгоритмів опрацювання зображень, що дозволять підвищити їх якість, зокрема зниження рівня шуму, адаптивне коригування рівня яскравості. Контрастності, вирівнювання гістограми.

Предсставимо запропонований алгоритм адаптивного покращення якості зображень у такому вигляді.

Нехай задано *Im* – вхідне зображення. Представимо дане зображення у матричній формі (2.1).

$$\text{Im}= \begin{bmatrix} a_{0,1} & \cdots & a_{0,N-1} \\ \vdots & \ddots & \vdots \\ a_{M-1} & \cdots & a_{M-1,N-1} \end{bmatrix}, \qquad (2.1)$$

де a_{ij} – елемент зображення.

На першому етапі необхідно відфільтрувати зображення шляхом застрсування медіанного фільтру. Даний фільтр призначений зокрема і для зменшення рівня шуму «сіль-перець», який ще називають імпульсним шумом. Зазвичай, даний фільтр використовує вікно розміром 3 на 3 пікселі або 5 на 5 пікселів. На кожному кроці відбувається збір значень пікселів – сусідів для подальшого оброблення. В центр області потрапляють пікселі, що будуть відфільтровані. Тобто їхнє значення зміниться, наприклад з 150 на 137 в градаціях сірого. Для обчислення медіани відбувається сортування окола пікселя та вибірка центрального значення, що дозволяє уникнути імпульсних

шумів. Приклад використання алгоритму наведено у 2.2.

$$Im^I = M(Im) \tag{2.2}$$

Формулу для двох-вимірної медіанної фільтрації наведено у формулі 2.3:

$$Im^I_{i,j} = med\left[Im_{i+s,j+t}; (s,t) \in W\right]; i,j \in Z^2 , \tag{2.3}$$

де $Im^I_{i,j}$ - елемент матриці зображення після фільтрації;

$W_{s,t}$ - елемент масиву апертути зображення з розміром m x n;

$Im_{i,j}$ - елемент матриці вхідного зображення. Для проведення фільтрації даним фільтром обрано вікно зі співвідношенням сторін 3 на 3 пкселі. Процес здійснюється ітеративно для всього зображення та усіх пікселів, таким чином отримується нова версія зображення. До недоліків можна віднести високий час обробки великих за розміром зображень та неефективність для подалання Гаусового шуму[40].

Вибір алгоритму фільтрації та параметри алгоритмів є складним завданням та потребує проведення великої кількості експериментів. Для цього необхідно визначити рівень шуму кількісно, тобто оцінити рівень шуму з допомогою числа. Для цього зазвичай використовують стандартне відхилення. Тобто визначають значення яскравості пікселів загалом та значення окремих пікселів. Іншим відомим підходом є визначення рівня шуму на зображенні шляхом використання високочастотних компонентів. Для цього використовують фільтр Лапласа та Собеля.

До інших загальноприйнятих підходів до визначення рівня шуму можна віднести такі:

- оцінка рівня шуму за ентропією;
- методи на основі статистики градієнтів;
- підходи на оснвоі аналізу текстур;

- візуальний аналіз з допомогою людини-експерта;
- методи глибокого навчання.

Одним із найпопулярніших підходів є використання співвідношення сигнал/шум. Даний підхід оцінює інформацію між корисною інформацією та шумом.

Для обчислення цього значення пікового відношення сисгналу до шуму потрібно розрахувати середньоквадратичне відхилення (MSE) між двома зображеннями.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |Im^I(i,j) - Im\,(i,j)\,|^2\,, \tag{2.4}$$

де Im^I та Im - результуюче та вхідне зображення, розміром m на n пікселів. Величина PSNR визначається так:

$$PSNR = 10log_{10}\left(\frac{MAX_I^2}{MSE}\right), \tag{2.5}$$

де MAX_I - це максимальне значення, яке приймається пікселем зображення.

Особливість опрацювання зображень полягає ще й у тому, що необхідно розробити механізми для зменшення рівня як Гаусового так і імпульсного шумів.

На основі експериментального підходу було підібрано такі правила для підбору алгоритмів фільтрації та параметрів фільтрації:

$$\begin{cases} mw = 7x7,\ \ gw = 3x3; PSNR \leq 20\ dB \\ mw = 5x5,\ \ gw = 3x3; 24 < PSNR > 20\ dB \\ mw = 3x3,\ \ gw = 5x5; PSNR > 24\ dB \end{cases}, \tag{2.6}$$

де mw – розмір вікна у пікселях медіанного фільтра, gw – розмір вікна у пікселях гаусового фільтра.

Фільтрація зображень.

Адитивний шум можна зменшити завдяки використанню Гаусового фільтру. Гаусовий фільтр використовується для таких цілей:

- зниження рівня шуму;
- попередня обробка перед детектуванням контурів;
- сегментація;
- створення розмиття (ефекту мякості);
- підготовка до аналізу текстур.

Застосування фільтру до попереднього зображення наведено у (2.7):

$$Im^{II} = gw * Im^{I} \tag{2.7}$$

Операцію згортки гаусового фільтру для координат матриці наведено у 2.8.

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.8}$$

де σ – радіус вікна згортки.

Застосуємо медіанний фільтр з розміром вікна mw. Це дозволить зменшити рівень імпульсного шуму.

$$Im^{III} = mw * Im^{II}, \tag{2.9}$$

де Im^{II} – вхідне зображення, mw – вікно фільтру, Im^{III} – зображення після фільтрації.

Вирівнювання рівня гістограми зображення наведено у 2.10:

$$Im^{IV} = H(Im^{III}), \tag{2.10}$$

де Im^{IV} – результуюче зображення після обробки, Im^{III} – вхідне зображення.

Важливим етапом є забезпечення оптимального коригування рівня яскравості та контрастності:

$$Y = \frac{1}{n}\sum_{i=0}^{n} 0.299 * R_i + 0.587 * G_i + 0.114 * B_i\,, \qquad (2.11)$$

де n – кількість пікселів, R_i, G_i, B_i - значення червоного, зеленого та синього каналів і – го пікселя зображення відповідно.

$$R = \frac{1}{N}\sum_{p=1}^{N} R_p \qquad (2.12)$$

$$G = \frac{1}{N}\sum_{p=1}^{N} G_p \qquad (2.13)$$

$$B = \frac{1}{N}\sum_{p=1}^{N} B_p \qquad (2.14)$$

На основі експериментального підходу було підібрано такі параметри α:

$$\alpha = \begin{cases} 55; \mathrm{Y} \leq 1 \\ 45;\ 1 < \mathrm{Y} \leq 2 \\ 40;\ 2 < \mathrm{Y} \leq 4 \\ 37;\ 4 < \mathrm{Y} \leq 7 \\ 25;\ 7 < \mathrm{Y} \leq 13 \\ 22;\ 13 < \mathrm{Y} \leq 30 \\ 19;\ 30 < \mathrm{Y} \leq 100 \\ 12;\ 100 < \mathrm{Y} \leq 200 \\ 3;\ \mathrm{Y} > 200 \end{cases} \qquad (2.15)$$

Коригування яскравості зображення.

На основі визначеного параметру α здійснюємо таке перетворення зображення:

$$Im^V = \alpha * Im^{IV} \quad (2.16)$$

Графічно алгоритм адаптивного покращення якості зображень наведено на рисунку 2.1.

Рисунок 2.1 - Алгоритм адаптивного покращення якості зображень

В результаті роботи алгоритму отримуємо вихідне зображення. Правили формування підбору параметрів опрацювання для коригування рівня шумів та яскравості є статичними, однак мають можливість до розширення вибірки.

2.2 Адаптивний алгоритм сегментації зображень

Сегментація зображень є важливим елементом та кроком в процесі опрацювання зображень, зокрема в галузі комп’ютерного зору та глибокого навчання. Однак зображення характеризуються різним рівнем складності, тому розробка універсального підходу до сегментації зображень є важливим завданням[41-42]. Сегментацію використовують зокрема для вирішення наступних задач:

- обробка та аналіз біомедичних зображень;
- розпізнавання об'єктів та розширена реальність;
- автономне водіння та системи допомоги водіям;
- агробізнес та моніторинг урожайності;
- відстеження та розпізнавання людей;
- розширений аналіз текстур та поверхонь;
- обробка супутникових зображень.

Серед алгоритмів сегментації можна виділити такі:

- підходи на основі поділу та злиття. Принцип полягає у розподілу зображення на окремі області і аналізують схожість пікселів.
- підходи на основі контурного аналізу. Серед них виділяють алгоритм Собеля, Кенні.
- підходи на основі графів. Уявляють зображення як граф, де кожен піксель є вершиною, а їхня схожість або відстань є ребром між вершинами.

- методи на основі розпізнавання шаблонів. U-Net, Mask R-CNN, Fully Convolutional Networks (FCN), які виконують семантичну або інстансну сегментацію.
- методи на основі текстурних характеристик.

Алгоритм вибору алгоритмів і параметрів сегментації такий:

1. Завантаження вхідного зображення у пам'ять.
2. Визначення вхідних характеристик.
3. Збереження отриманих результатів у базі даних за ідентифікатором зображення.
4. Формування масиву із параметрами:
   - значення яскравості;
   - середні значення червоного каналу;
   - середні значення зеленого каналу;
   - середні значення синього каналу.
5. На етапі сегментації застосовуються такі алгоритми:
   - порогова сегментація;
   - метод водорозподілу;
   - метод k – середніх.
   - U-net мережа;

На етапі використання порогової сегментації вибирається нижній поріг у діапазоні від 25 до 185, кроком – 10.

Метод k – середніх використовує різні прапорці для тестування кращого результату.

На етапі використання U-net – мереж використовуються різні архітектури та розміри вхідних зображень.

6. Проміжні результати досліджень зберігаються в базу даних для подальшого використання.
7. Для оцінки якості сегментації, отримане зображення порівнюється із еталонним, оброблене експертом. В якості метрик використовується

параметр FRAG, метрика Фреше, Громова – Фреше. Також використовується експертна оцінка.

8. В базу даних записується результат у вигляді:
   «вхідні параметри» - «кращий результат (алгоритм , параметри алгоритму)»
9. При надходженні нового зображення відбувається визначення вхідних параметрів цього зображення.
10. Підбір кращого алгоритму та його параметрів для конкретного зображення.

Структуру підходу до автоматичної сегментації зображень наведено на рисунку 2.2.

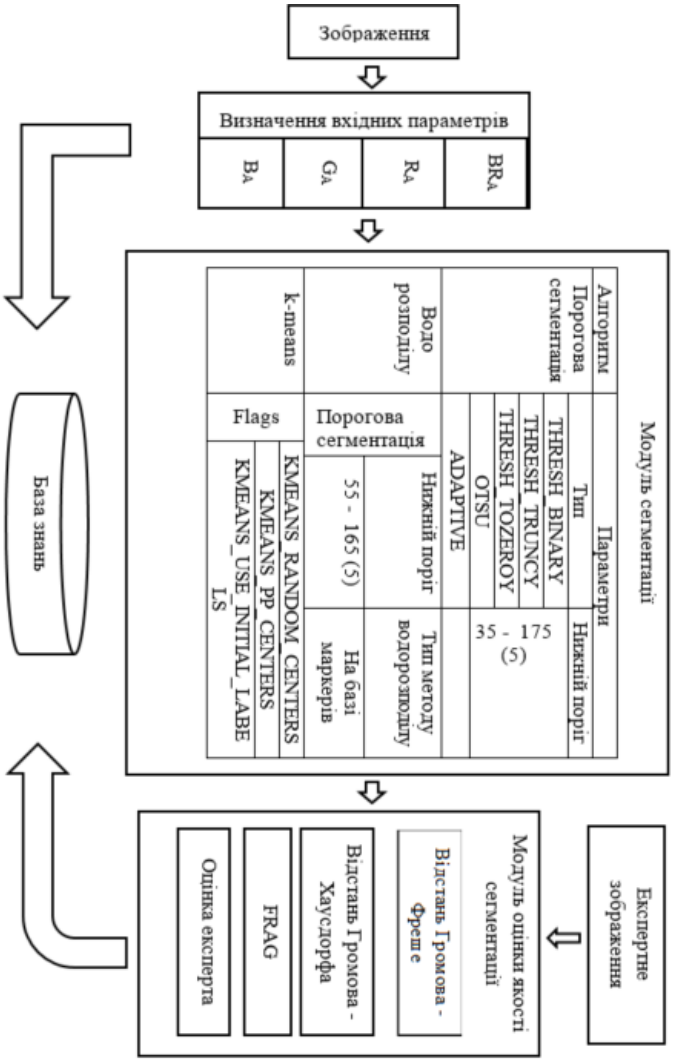


Рисунок 2.2 - Структуру підходу до автоматичної сегментації

Ключовим елементом у даній системі є визначення кращого алгоритму та його параметрів та зберігання проміжних результатів у базі даних.

Структуру таблиці бази даних для зберігання параметрів сегментації наведено у таблиці 2.1.

Таблиця 2.1 - Структура таблиці бази даних

Таблиця «AlgorithmParameters»	
Поля	Тип
id	integer
Image id	integer
Algorithm id	integer
Parameters	String
Expert	integer

Архітектурі U-net мережі наведено на рисунку 2.3.

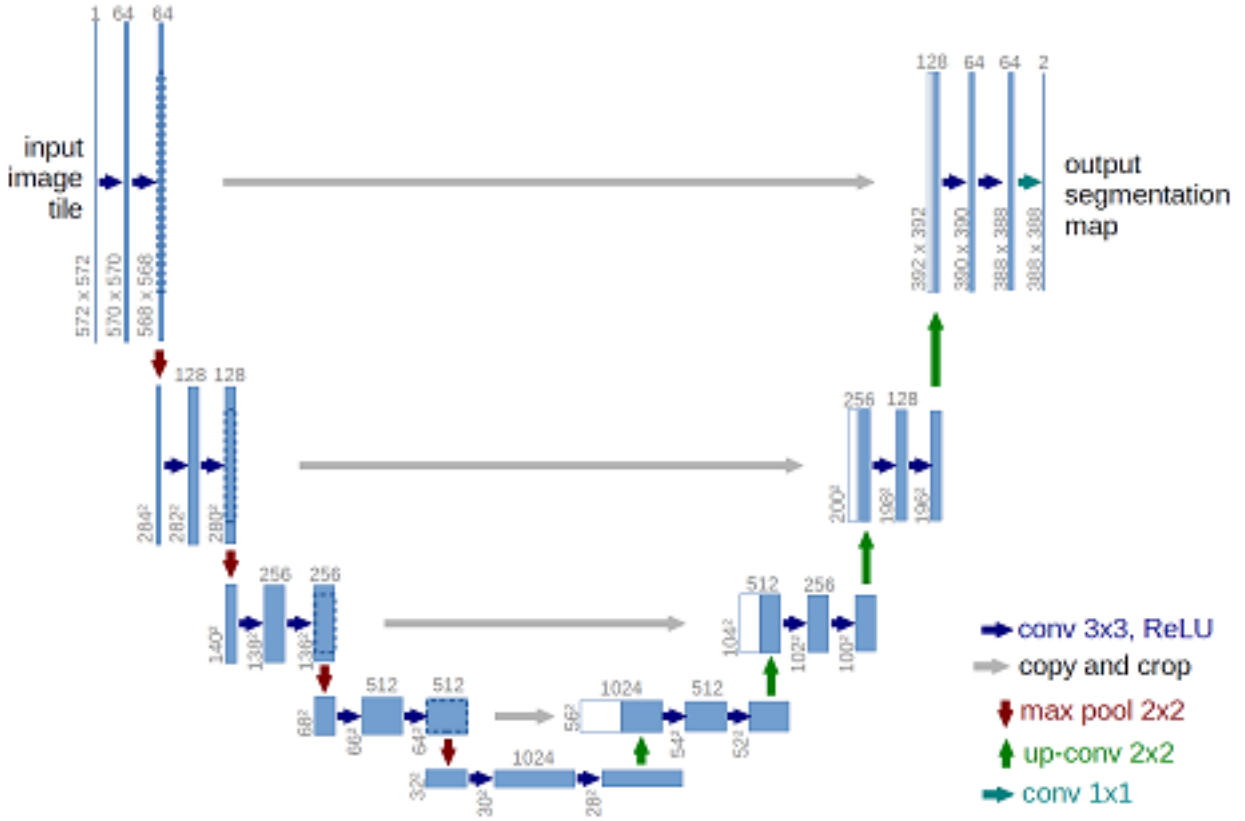
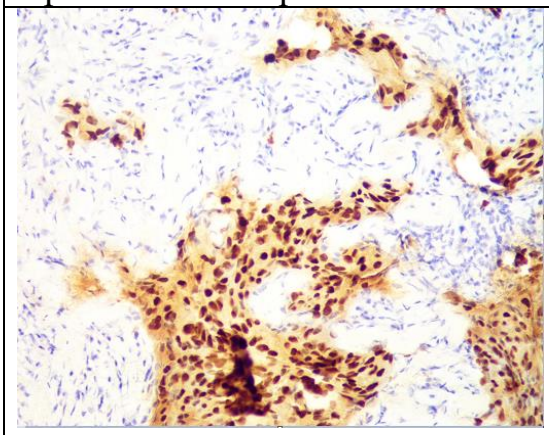
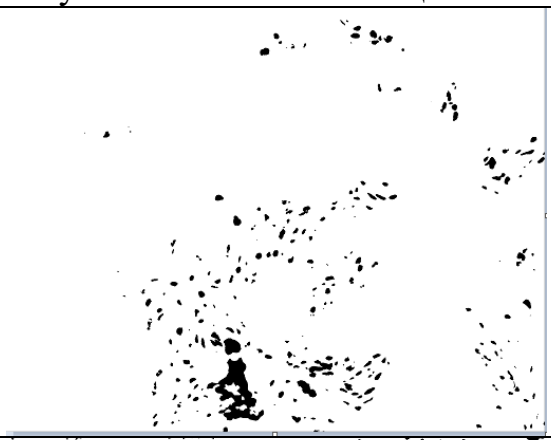
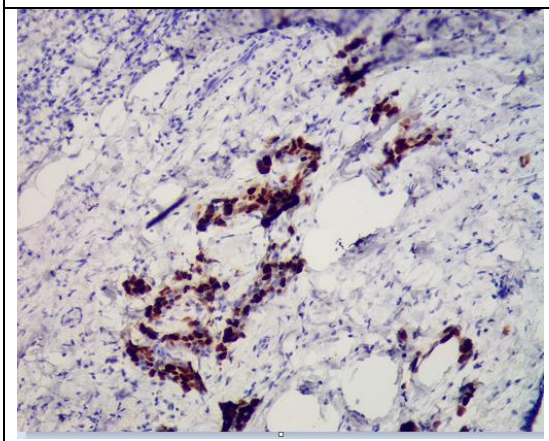
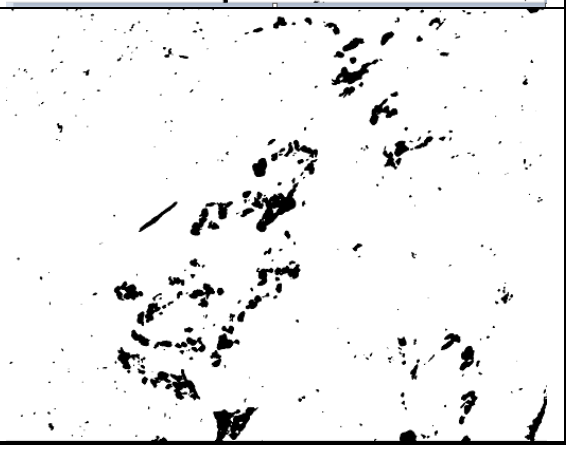


Рисунок 2.3 - Архітектурі U-net мережі

U-net мережу використовують для сегментації зображень з використанням алгоритмів глибинного навчання.

Приклади зображень, які відносяться до класу «Люмінальний Б» наведено таблиці 2.2.

Таблиця 2.2 – Клас «Люмінальний Б»

Оригінальне зображення	Результат після сегмантації
	
	

Енкодер — це частина мережі, яка вибірково витягує найважливішу інформацію з вхідних даних і зменшує їх розмірність, залишаючи основні характеристики.

Декодер відновлює стиснені дані, створені енкодером, до вихідного вигляду або нової структури, схожої на оригінальні дані.

У деяких архітектурах (наприклад, U-Net) використовується з’єднання з енкодером, яке дозволяє передавати інформацію з ранніх шарів енкодера до декодера, покращуючи просторову точність вихідного результату.

Архітектуру енкодера наведено на рисунку 2.4.

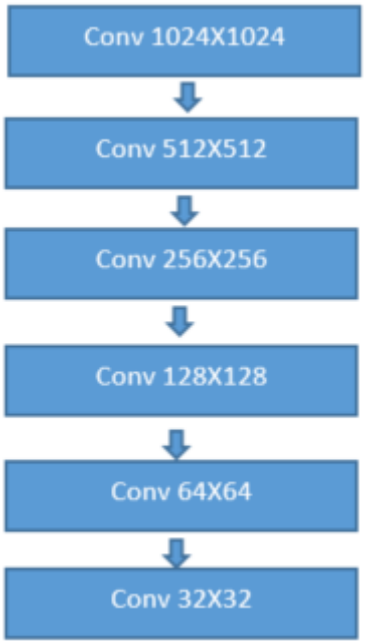


Рисунок 2.4 - Архітектура енкодера

Енкодер-декодер в CNN допомагає стискати дані до їх основних рис та відновлювати їх з максимальною точністю, що є корисним для завдань, де потрібна глибока обробка просторових характеристик.

2.3 Алгоритм розгортання ПЗ на хмарному сервісі

У обробці зображень MLOps може значно спростити розробку та підтримку моделей, які виконують задачі з аналізу та покращення зображень. Збір та анотування зображень, перетворення їх у потрібний формат, розбиття на навчальні, валідаційні та тестові набори даних

Робочий процес MLOps для сегментації біомедичних зображень показаний на рисунку 2.5.

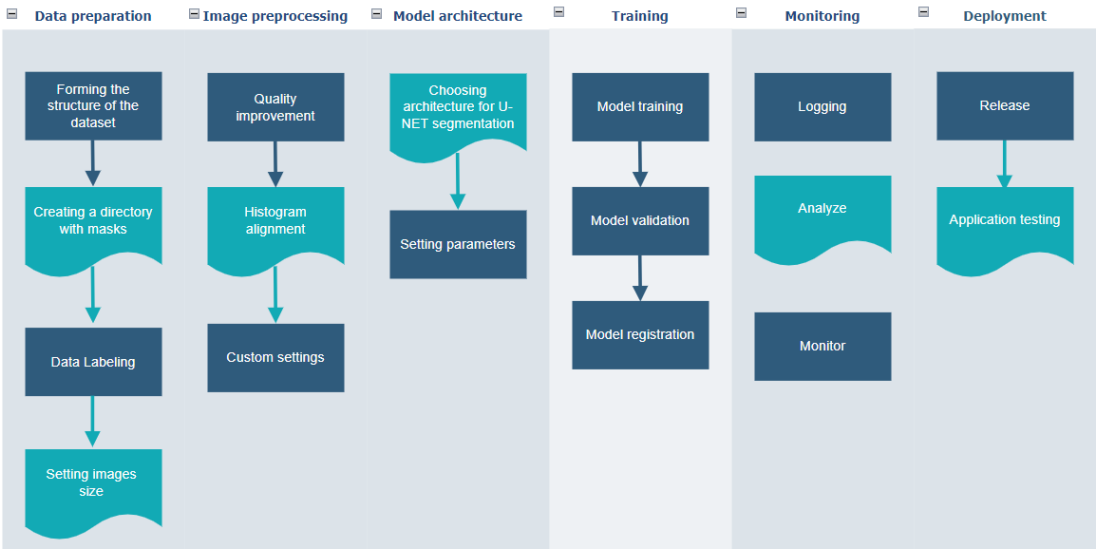


Рисунок 2.5 - Робочий процес MLOps для сегментації біомедичних зображень

Важливою складовою програмної системи є модуль оцінки кількісних характеристик мікрооб'єктів. На відміну від інших модулів, цей модуль може часто змінювати код. Це пов’язано з необхідністю задавати параметри для різних типів зображень. Для автоматизації процесу передачі налаштувань параметрів пропонується використовувати окремий файл deploy.yml (рисунок 2.6).

```yaml
name: Build & Deploy
on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Deploy Cell Parameters
        uses: actions/checkout@v3
        with:
          host: ${{secrets.SSH_HOST}} # IP address of the server you wish to ssh into
          key: ${{secrets.SSH_KEY}} # Private or public key of the server
          username: ${{ secrets.SSH_USERNAME }} # User of the server you want to ssh into
          passphrase: ${{ secrets.SERVER_PASSPHRASE }}

          script: |
            mkdir experiments
            cd experiments
            git clone https://github.com/olehpitsun/AutoImageSegmentation.git
            echo 'Deployment successful to digitsal ocean'

```

Рисунок 2.6 - deploy.yml

Файл deploy.yml часто використовується для автоматизації процесу розгортання додатків або сервісів у середовищі розробки, тестування чи продакшені. Він зазвичай написаний на мові YAML і містить інструкції для розгортання, які можуть включати визначення конфігурацій, середовищ виконання та параметрів.

2.4 Висновки до розділу 2

На основі алгоритмічного підходу розроблено алгоритм покращення якості зображень з використанням етапів фільтрації та коригування рівня яскравості, що дозволить застсоувати алгоритми та їх параметри в залежності від вхідних параметрів.

На основі алгоритмічного підходу розроблено підхід до сегментації зображень на основі їх вхідних параметрів, що дозволило підбирати алгоритми сегментації та їх параметри.

Розроблено структуру конфігураційного файлу deploy.yml можливості розгортання алгоритмів у хмарному середовищі.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ НА ХМАРНОМУ СЕРВІСІ

3.1 Структура програмного модуля опрацювання зображень

Архітектура програмних модулів – це фундамент, на якому будується будь-яке програмне забезпечення. Вона визначає структуру, взаємодію та функціональність окремих компонентів системи. Архітектура програмних модулів – це фундамент, на якому будується будь-яке програмне забезпечення. Вона визначає структуру, взаємодію та функціональність окремих компонентів системи. Для невеликих проектів можуть підходити простіші рішення, тоді як для великих систем потрібні більш складні архітектури. Вибір технологій впливає на можливі архітектурні рішення. Особливості предметної області визначають специфічні вимоги до системи. Досвід і навички команди також впливають на вибір архітектури. Архітектура повинна бути зрозумілою для всіх учасників проекту. Архітектура повинна підтримувати функціональні та нефункціональні вимоги до системи. Система повинна легко інтегруватися з іншими системами та компонентами та повинна бути доступною для користувачів з різними рівнями знань.

Монолітна архітектура – це традиційний підхід до розробки програмного забезпечення, де всі компоненти системи тісно пов'язані між собою і функціонують як єдине ціле. Характерні риси монолітної архітектури є те, що вся система складається з одного великого кодового базису, компоненти сильно залежать один від одного, зміна в одному компоненті може призвести до проблем в інших, використання однієї бази даних для зберігання всіх даних системи. Вся система розробляється на одній технологічній платформі.

Мультисервісна архітектура – це сучасний підхід до розробки програмного забезпечення, який полягає в розбитті системи на дрібні, незалежні сервіси. Система розбивається на дрібні, самодостатні сервіси, які взаємодіють між собою через чітко визначені інтерфейси (зазвичай REST API

або gRPC). Різні сервіси можуть бути розроблені на різних технологічних платформах, що дозволяє використовувати найкращі інструменти для кожної задачі. Кожен сервіс може мати свою базу даних, що забезпечує більшу автономність і масштабованість. Завдяки децентралізованій структурі, відмова одного сервіса не призводить до збою всієї системи. Вибір між монолітною і мультисервісною архітектурою залежить від конкретних вимог проекту. Для невеликих проектів з обмеженим функціоналом монолітна архітектура може бути достатньою. Для великих і складних систем мультисервісна архітектура зазвичай є більш кращим вибором.

Java є однією з найпоширеніших мов програмування, яку часто обирають для задач обробки зображень. Програми на Java компілюються в байт-код, який може виконуватися на будь-якій платформі, де встановлена JVM. Додатки для обробки зображень, написані на Java, можуть працювати на Windows, macOS, Linux та інших операційних системах без значних змін. Існує велика кількість безкоштовних і відкритих бібліотек, спеціалізованих на обробці зображень (наприклад, OpenCV, ImageJ). Вони надають готові інструменти для виконання різних операцій, таких як фільтрація, сегментація, розпізнавання об'єктів тощо. Сучасні JVM оптимізовані для виконання Java-коду, що забезпечує високу продуктивність.

OpenCV - це одна з найпопулярніших бібліотек з відкритим кодом, призначена для обробки зображень та відео. Вона широко використовується в різних галузях, таких як комп'ютерний зір, робототехніка, медична візуалізація та багато інших. OpenCV працює на різних операційних системах (Windows, Linux, macOS), що забезпечує гнучкість у розробці.

Бібліотека володіє широким спектром алгоритмів для обробки зображень:

- виявлення об'єктів - виявлення облич, очей, усмішок тощо;
- розпізнавання образів - розпізнавання облич, номерних знаків, штрих-кодів;
- ссегментація зображень - розділення зображення на окремі області;

- фільтрація зображень - згладжування, посилення контрасту, шумозаглушення;
- геометричні перетворення - поворот, масштабування, проектування;
- калібрування камер - визначення внутрішніх параметрів камери;
- 3D реконструкція - створення тривимірних моделей на основі двовимірних зображень.

Структура модулю для опрацювання зображень на низькому рівні комп'ютерного зору наведено на рисунку 3.1.

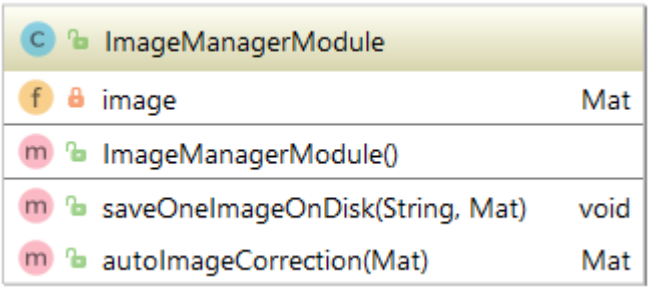


Рисунок 3.1 - Структура модулю для опрацювання зображень на низькому рівні комп'ютерного зору

Метод «saveOneImageOn Disk» відповідає за збереження зображення на дисковий простір. Вхідними параметрами є шлях до місця збереження та безпосередньо саме зображення в форматі Mat бібліотеки openCV.

Важливим елементом будь якої програмної системи є механізм для зберігання даних, зокрема база даних. Структура модулю для зберігання даних наведено на рисунку 3.2.

Клас «SQLDatabase» вміщує методи для таких функцій:

- зберігання інформації;
- оновлення даних;
- створення даних;
- видалення даних.

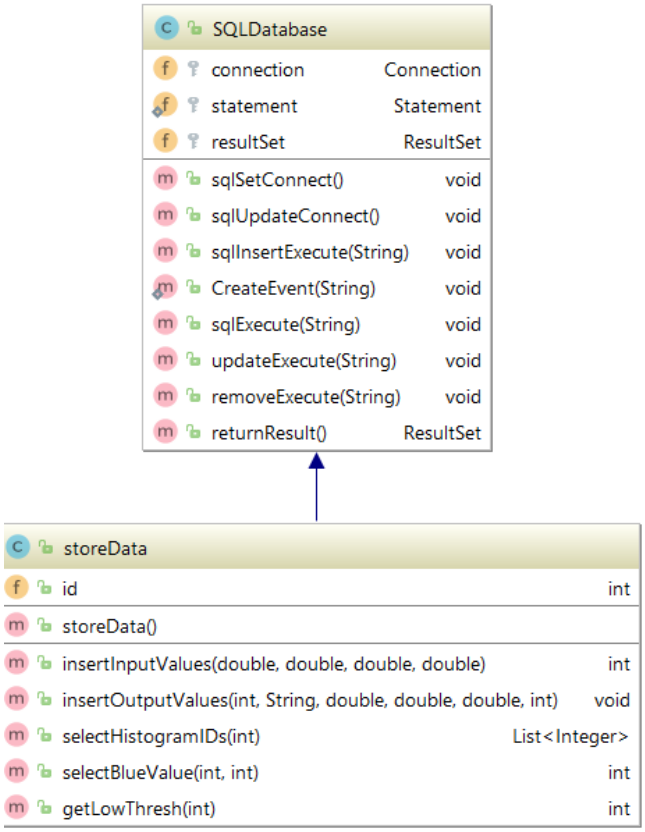


Рисунок 3.2 – Структура модулю для зберігання даних

Також наявні поля для під’єднання до драйвера бази даних SQL та ResultSet.

Клас «StoreData» складється з методів для додавання вхідних рараметрів зображення в форматі double, зокрема значення рівнів каналів червоного, зеленого, синього, середнього значення гістограми та інші параметри.

Наявні методи методи «selectHistogram» та «selectBlueValue» для окремого зберігання даних параметрів. Також наявне поле «id» для можливості подальшого зберігання та використання запису в системі. На рисунку 3.3 наведено структуру модулю для корекції зображень.

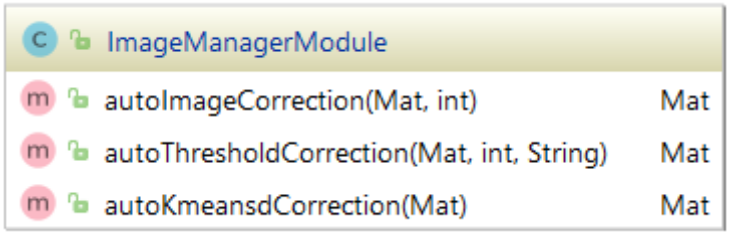


Рисунок 3.3 - Структура модулю для корекції зображень

У даному класі наявні методи для автоматичного покращення якості зображення. На вхід передається формат Mat для зберігання самого зображення та int для зберігання коефіцієнту коригування. Метод «autoThresholdCorrection» призначений для автоматичної сегментації алгоритмами порогової сегментації. Метод «autoKmeansCorresction» призначений для автоматичної сегментації зображень методами K-means. Приклад програмного коду для виклику методу автоматичної сегментації наведено нижче:

Mat result = Segmentation.watershed (contrastMat, lowLevel); Highgui.imwrite("E:\\IH_images\\190-20_LA\\Her2n\\1_watershed.jpg" , result);

У даному випадку клас «Segmentation» володіє статичними методами для зберігання алгоритмів сегментації.

Приклад програмного коду для автоматичного покращення якості зображень наведено нижче:

FiltersOperations filtroperation = new FiltersOperations(src, "4", "5", "", "", ""); // медіанний фільтр

Mat brightMat = PreProcImgOperations.bright (filtroperation.getOutputImage(), 10); // яскравість

filtroperation.getOutputImage().release();

Mat contrastMat = PreProcImgOperations.contrast(brightMat, 1.0);

brightMat.release();

Даний код містить методи класу «*PreProcImgOperations*», який використовується для коригування рівня яскравості, контрастності. А також, вміщує вхідні параметри для даних методів, наприклад розмір вікна тощо.

Хмарні сервіси для обробки даних надають потужні інструменти та інфраструктуру, які дозволяють компаніям та організаціям ефективно збирати, зберігати, аналізувати та візуалізувати великі обсяги даних. Ці сервіси пропонують гнучкість, масштабованість та доступність.

Переваги використання хмарних сервісів для обробки даних:

- масштабованість - легко збільшувати або зменшувати обчислювальні ресурси;
- гнучкість - швидке розгортання нових додатків і сервісів;
- економія - оплата лише за використані ресурси;
- надійність - високий рівень доступності та безпеки даних.

MLOps життєвий цикл для автоматичної сегментації наведено на рисунку 3.4.

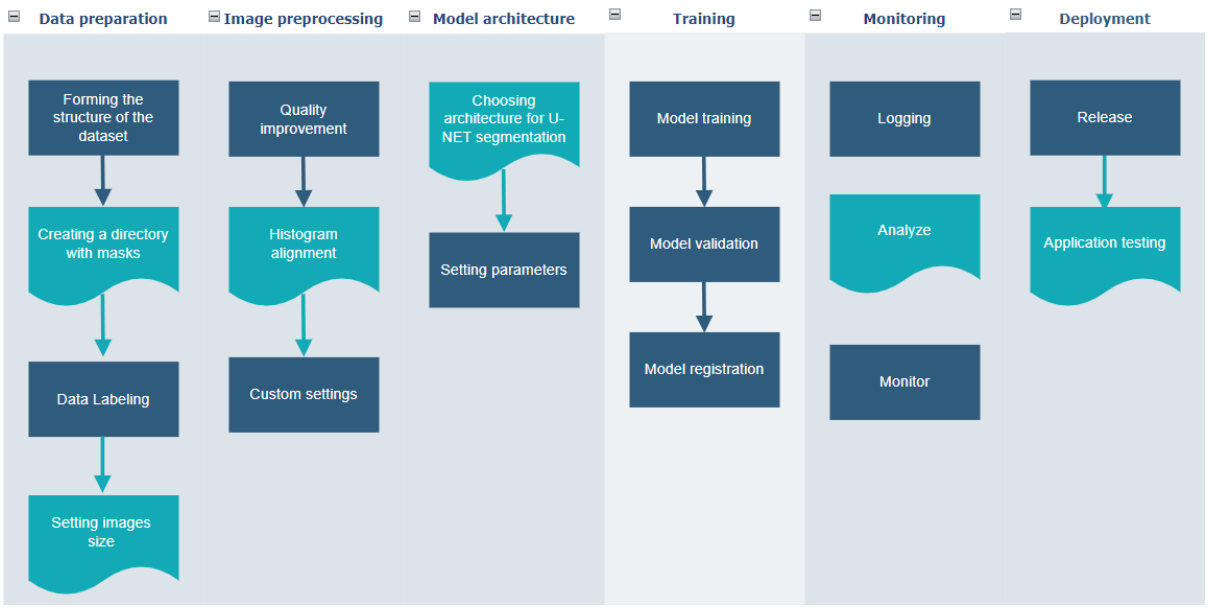


Рисунок 3.4 - MLOps життєвий цикл для автоматичної сегментації

У даному життєвому циклі розробки ПЗ на явні такі блоки:

- підготовка даних;
- попередня обробка зображення;

- вибір архітектури Unet мережі для сегментації зображення;
- навчання нейронної мережі;
- моніторинг;
- деплой.

На етапі попереднього оброблення відбувається підготовка структури даних.

3.2 Реалізація CI/CD

Безперервна інтеграція – це процес, при якому розробники часто інтегрують свій код у спільний репозиторій. Кожна інтеграція перевіряється автоматизованими тестами, щоб виявити помилки на ранніх етапах розробки. Для невеликих проектів може бути достатньо Continuous Delivery, тоді як для великих і складних систем може знадобитися більш консервативний підхід. Якщо вимоги до якості продукту дуже високі, може бути доцільно використовувати більш ретельне тестування перед розгортанням в продакшен. Для реалізації CI/CD необхідно створити проект в системі контролю версій, приклад створення проекту та директорії workflows наведено на рисунку 3.5.

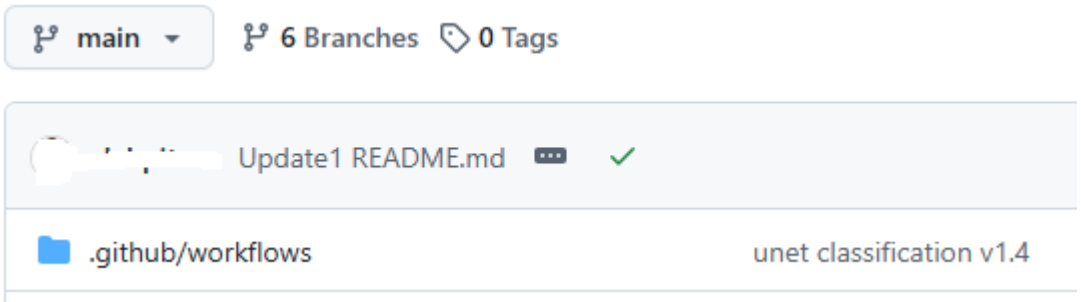


Рисунок 3.5 - Приклад створення проекту та директорії workflows

Вміст розробленого файлу «.github/workflows/deploy.yml» наведено на рисунку 3.6.

Code Blame 22 lines (20 loc) · 700 Bytes

```yaml
name: Build & Deploy
on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Deploy Unet
        uses: actions/checkout@v3
        with:
          host: ${{secrets.SSH_HOST}} # IP address of the server you wish to ssh into
          key: ${{secrets.SSH_KEY}} # Private or public key of the server
          username: ${{ secrets.SSH_USERNAME }} # User of the server you want to ssh into
          passphrase: ${{ secrets.SERVER_PASSPHRASE }}

          script: |
            mkdir experiments
            cd experiments
            git clone https://github.com/olehpitsun/Pytorch-UNet.git
            echo 'Deployment successful to digitsal ocean'
```

Рисунок 3.6 - Вміст розробленого файлу «.github/workflows/deploy.yml»

Даний проект містить базові та додаткові пункти для розгортання проекту на хмарному сервісі digitalocean. Першочерговою метою є під’єднання до мережі по захищеному з’єднанні з використанням SSH ключів та паролю. Також у даному скрипті вказано команди для створення директорій у хмарному сховищі та посилання на гітхаб – репозиторій з кодом проекту. Додаткові файли призначені для розгортання необхідних бібліотек та залежностей.

Ще одним підходом до реалізації DevOps підходу є використання terraform. Конфігурацію файлу terraform наведено на рисунку 3.7.

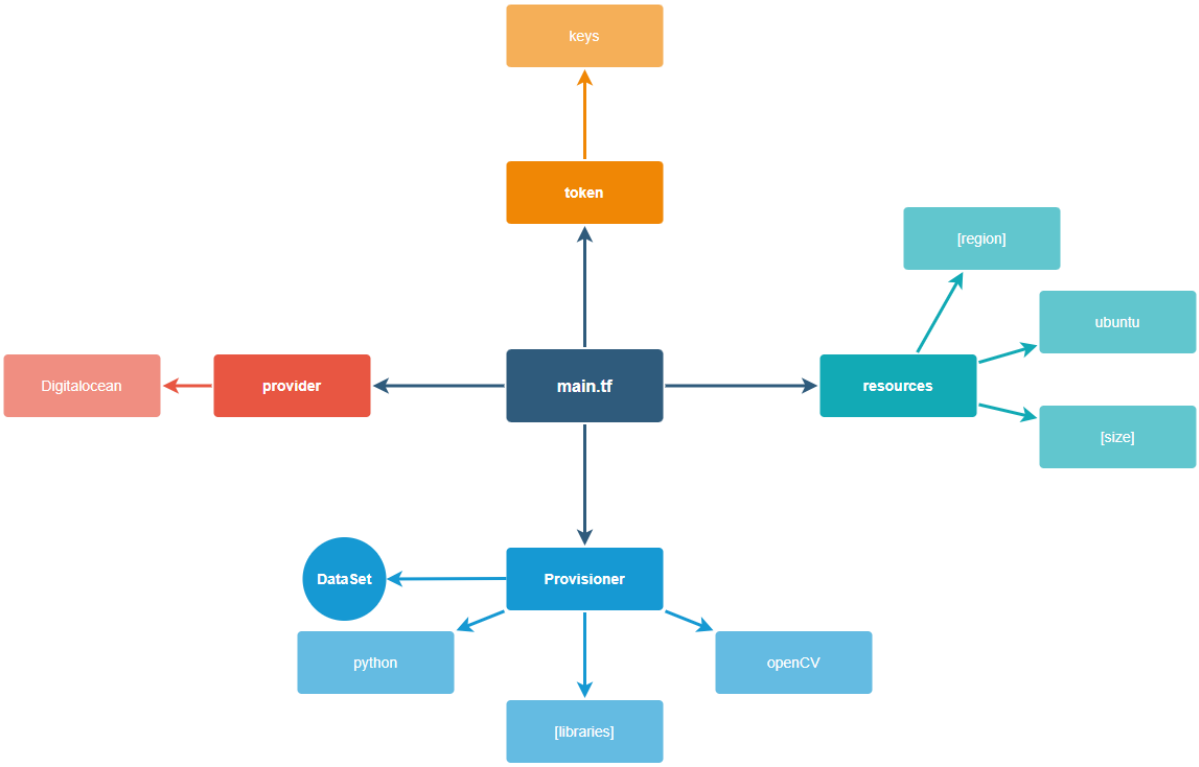


Рисунок 3.7 - Конфігурація файлу terraform

Провайдери визначають, з яким хмарним сервісом або платформою працюватиме Terraform (AWS, Azure, Google Cloud, тощо). Ресурси описують, які об'єкти потрібно створити чи управляти ними (наприклад, сервери, бази даних, мережеві компоненти). Variables використовуються для параметризації конфігурації, що дозволяє зробити її більш гнучкою. Outputs дозволяють експортувати певні дані після виконання конфігурації (наприклад, IP-адресу створеного інстансу). Модулі використовуються для повторного використання конфігурації. Модулі дозволяють структурувати код для складних проєктів. Data Sources дозволяють отримувати інформацію про наявні ресурси. Локальні змінні використовуються для обчислення або збереження значень, які не змінюються під час виконання конфігурації.

Результатом використання терраформ є створений дроплет у сервісі digitalocean, зображений на рисунку 3.8.

Рисунок 3.8 – Створений дроплет у сервісі digitalocean

Droplet у DigitalOcean - це віртуальний приватний сервер (VPS), індивідуальна віртуальна машина, яку можуf налаштувати та використовувати для запуску веб-сайтів, додатків та інших сервісів.

Основні характеристики Droplet:

- індивідуальність - кожен Droplet - це окремий сервер, який не ділить ресурси з іншими користувачами;
- гнучкість дозволяє вибрати операційну систему, кількість ядер процесора, об'єм оперативної пам'яті та дискового простору відповідно до ваших потреб;
- швидкість - Droplets зазвичай працюють на швидких SSD-дисках, що забезпечує високу продуктивність;
- масштабованість – дозволяє збільшити або зменшити ресурси Droplet в залежності від навантаження на ваш сервер.

3.3 Порівняльний аналіз результатів

У даному підрозділі наведено аналіз результатів роботи розробленого підходу до опрацювання зображень в хмарному середовищі.

Зашумлення зображень і алгоритми розмиття використовуються в різних областях комп'ютерного зору, обробки зображень та машинного навчання для вирішення різних завдань. Зашумлення є однією з технік аугментації, що дозволяє збільшити розмір навчального набору, підвищуючи узагальнювальну здатність моделі. Додавання шуму до зображень (наприклад, гаусового шуму або "солі з перцем") використовується для тренування моделей, щоб зробити їх більш стійкими до реальних умов, коли дані можуть бути зашумленими.

Розмиття згладжує текстури та зменшує шум перед застосуванням інших алгоритмів, наприклад, для виявлення контурів.

Приклад зашумленого зображення наведено на рисунку 3.9.

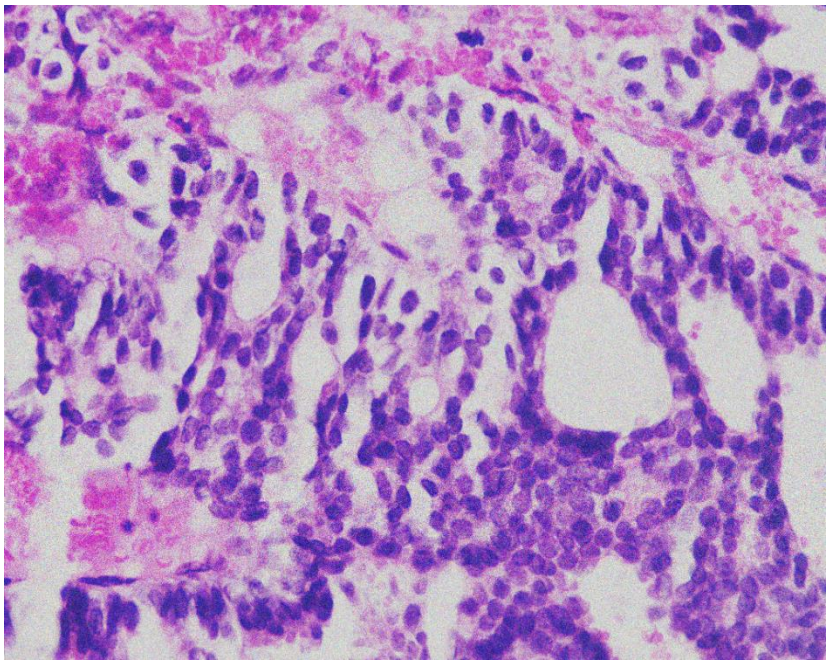


Рисунок 3.9 – Приклад зашумленого зображення

Результат використання Гаусового фільтру наведено на рисунку 3.10.

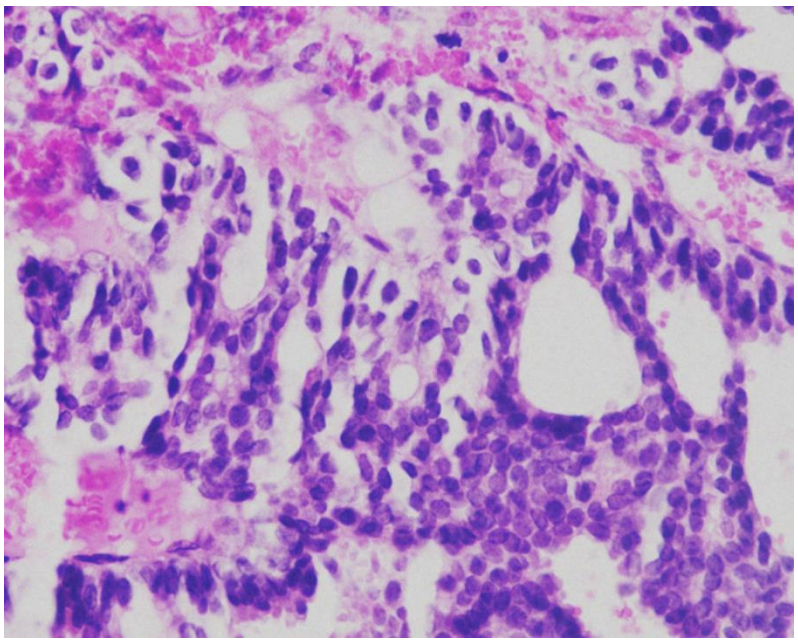
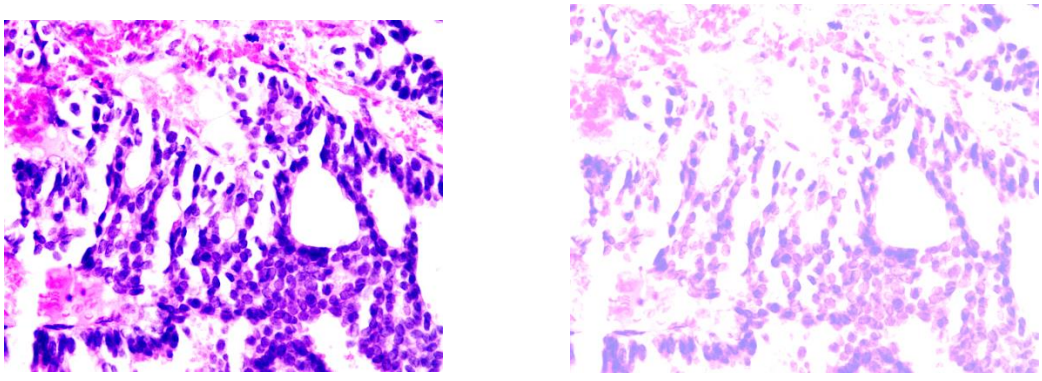


Рисунок 3.10 - Результат використання Гаусового фільтру

Коригування рівня яскравості та контрастності зображення значно впливає на його сприйняття, аналіз та обробку в різних задачах комп’ютерного зору. Ці параметри визначають, наскільки добре видно деталі на зображенні, і впливають на ефективність алгоритмів, що обробляють зображення.

На рисунку 3.11 наведено результат обробки зображень шляхом коригування рівня контрастності та яскравості.



А) контраст Б) яскравість

Рисунок 3.11 – Коригування рівня яскравості, контрастності

Приклад автоматичної корекції зображення на низькому рівні комп’ютерного зору наведено на рисунку 3.12.

Збільшення яскравості допомагає зробити темні ділянки зображення більш помітними, що полегшує аналіз слабоконтрастних або недостатньо освітлених об’єктів.

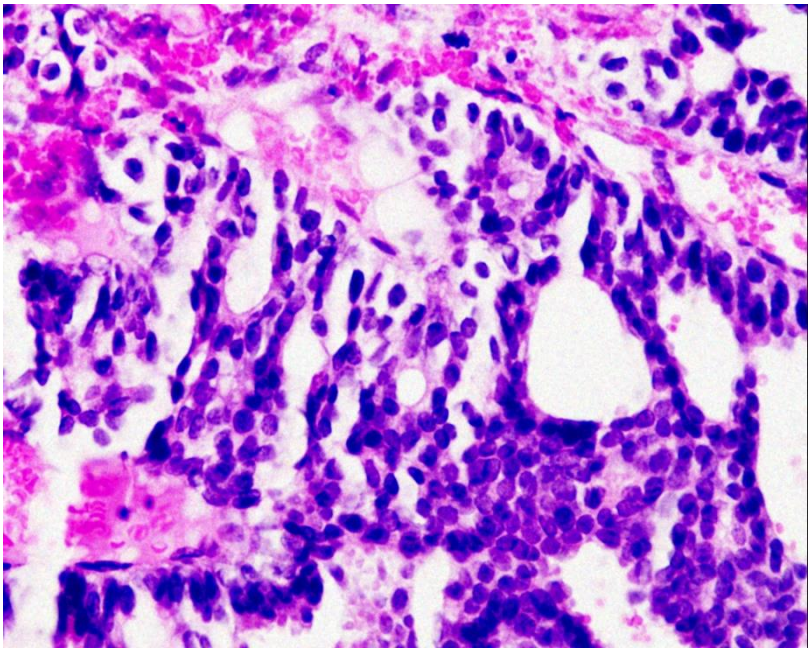


Рисунок 3.12 - Приклад автоматичної корекції зображення на низькому рівні комп’ютерного зору

Приклад сегментації зображень без попередньої обробки наведено на рисунку 3.13.

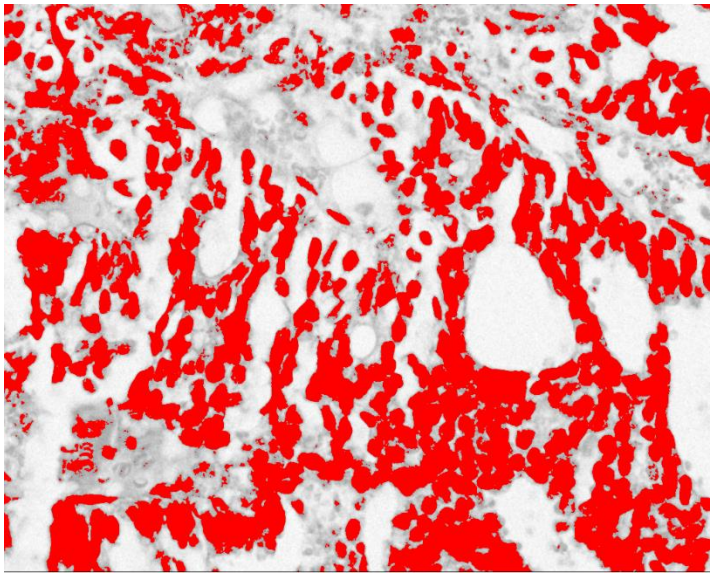


Рисунок 3.13 - Приклад сегментації зображень без попередньої обробки

Приклад автоматичної сегментації зображення наведено на рисунку 3.14.

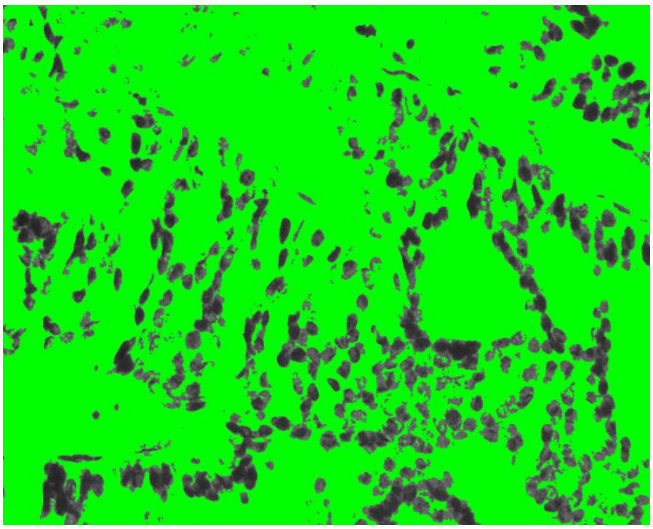


Рисунок 3.14 - Приклад автоматичної сегментації зображення

На основі візуального аналізу можна дійти висновку, що автоматична сегментація забезпечує більш якісне виділення досліджуваних об'єктів, зокрема ядер клітин, і дозволяє уникнути помилкового віднесення фону до мікрооб'єктів.

Порівняльний аналіз алгоритмів адаптивного покращення якості зображень за критерієм SSIM наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз алгоритмів адаптивного покращення якості зображень за критерієм SSIM

№ зображення	Сегментація без попереднього оброблення	HE	CLAHE	MSR	Розроблений підхід
1	0.48	0.54	0.82	0.83	0.89
2	0.54	0.23	0.76	0.55	0.8
3	0.23	0.56	0.63	0.72	0.81
4	0.59	0.55	0.57	0.72	0.73

Порівняльний аналіз розробленого підходу із аналогами наведено у таблиці 3.2.

Таблиця 3.2 - Порівняльний аналіз розробленого підходу із аналогами

Тип	Roboflow	Levity	Hasty	Розроблений підхід
Фільтрація	+	-	-	+
яскравість	+	+	+	+
сегментація	+	+/-	+/-	+
Terraform	+/-	+/-	+/-	+

В результаті аналізу можна зробити висновок, що розроблений підхід найбільш підходить для опрацювання зображень в хмарних середовищах.

3.4 Висновки до розділу

У даному розділі наведено структуру програмного модулю для опрацювання зображень на основі хмарних сервісів, що дозволило удосконалити процес розгортання програмного забезпечення для опрацювання зображень.

Реалізовано механізм безперервної доставки та інтеграції програмного коду для опрацювання зображень низькому та середньому рівнях комп’ютерного зору в хмарному середовищі.

Наведено порівняльний аналіз розроблених алгоритмів на основі адаптивного підходу для автоматичного покращення якості та точності виділення мікрооб’єктів. Результати свідчать про перевагу розробленого підходу з відомими аналогами в задачах опрацювання зображень.

ВИСНОВКИ

1. На основі аналітичного підходу проведено аналіз сучасних алгоритмів опрацювання зображень на низькому рівні комп’ютерного зору, що дозволило виокремити підходи до видалення шуму, коригування рівнів яскравості та контрастності, що дозволило розробити алгоритм для попереднього оброблення зображень.
2. На основі аналітичного підходу проведено аналіз сучасних алгоритмів сегментації зображень, що дозволило виокремити алгоритми та вхідні параметри для розробки адаптивного алгоритму в залежності від вхідних параметрів зображення.
3. На основі алгоритмічного підходу розроблено адаптивний алгоритм покращення якості зображення на основі вхідних параметрів зображень, зокрема рівнів каналів червоного, синього та зеленого каналів, що дозволило покращити якість обробки.
4. На основі алгоритмічного підходу розроблено адаптивний алгоритм сегментації зображень на основі вхідних параметрів зображення, що дозволило покращити процес сегментації на основі алгоритмів k-means, водорозподілу, порогової сегментації.
5. На основі технологій DevOps розроблено життєвий цикл функціонування програми для опрацювання зображень на рівні попередньої обробки та сегментації, що дозволяє розгортати програмний код на різних хмарних сервісах. Такий підхід дозволяє зробити програму універсальною та легкою у інтеграції.
6. Проведений порівняльний аналіз демонструє кращий рівень якості опрацювання за критерієм SSIM у порівнянні з іншими аналогами. Порівняльний аналіз DevOps технологій демонструє вищий рівень за багатьма показниками для систем на основі DevOps технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. O. Berezsky, L. Dubchak, N. Batryn, T. Datsko, K. Berezska, O. Pitsun, Y. Batko «Fuzzy System For Breast Disease Diagnosing Based On Image Analysis». Proceedings of the II International Workshop Informatics & Data-Driven Medicine (IDDM 2019). Lviv, Ukraine. 11-13 November, 2019.
2. C. Mazo, E. Orue-Etxebarria, I.Zabalza,; M.M. Vivanco, R.M.Kypta, A. Beristain «In Silico Approach for Immunohistochemical Evaluation of a Cytoplasmic Marker in Breast Cancer». Cancers 2018, 10, 517
3. O. Berezsky, L. Dubchak, N. Batryn, T. Datsko, K. Berezska, O. Pitsun, Y. Batko «Fuzzy System For Breast Disease Diagnosing Based On Image Analysis» Proceedings of the II International Workshop Informatics & Data-Driven Medicine (IDDM 2019). Lviv, Ukraine. 11-13 November, 2019
4. G. Stålhammar, , N. Fuentes Martinez, M. Lippert, et al. “Digital image analysis outperforms manual biomarker assessment in breast cancer”. Mod Pathol 29, 318–329 . - 2016.
5. M.Vandenberghe, M.Scott, P. Scorer, et al. “Relevance of deep learning to facilitate the diagnosis of HER2 status in breast cancer” Sci Rep 7, 45938 . - 2017
6. Xin-Ming Zhang, Qiang Kang, Jin-Feng Cheng, Xia Wang “ Adaptive Four-dot Median Filter for Removing 1-99% Densities of Salt-and-Pepper Noise in Images ” Journal of Information Technology Research (JITR) 11(3), 2018, P. 15
7. S. Robertson, H. Azizpour, K. Smith, J. Hartman Digital image analysis in breast pathology—from image processing techniques to artificial intelligence/ // Translational Research Volume 194, April 2018, Pages 19-35
8. M.A. Aswathy, M.Jagannath Detection of breast cancer on digital histopathology images: Present status and future possibilities . Informatics in Medicine Unlocked Volume 8, 2017, Pages 74-79

9. R. C. Gonzalez, R. E. Woods "*Digital Image Processing*, 4 th ed.,Pearson Education Limited 2018, P. 1022
10. O. Berezsky, L. Dubchak, O. Pitsun "Access distribution in automated microscopy system", 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 21-25 Feb. 2017, Lviv, Ukraine, 2017, pp. 241 – 243
11. Березький О.М. Системи автоматизованої мікроскопії: стан та перспективи розвитку / О.М. Березький, О.Й. Піцун, С.О. Вербовий // Вісник Хмельницького національного університету. Технічні науки. – 2016. – № 2 (235). – С. 61–68.
12. Zebari, D. A., Ibrahim, D. A., Zeebaree, D. Q., Haron, H., Salih, M. S., Damaševičius, R., & Mohammed, M. A. (2021). Systematic review of computing approaches for breast cancer detection based computer aided diagnosis using mammogram images. Applied Artificial Intelligence, 35(15), 2157-2203.
13. Moon, W. K., Lee, Y. W., Ke, H. H., Lee, S. H., Huang, C. S., & Chang, R. F. (2020). Computer-aided diagnosis of breast ultrasound images using ensemble learning from convolutional neural networks. *Computer methods and programs in biomedicine*, *190*, 105361.
14. Tanaka, H., Chiu, S. W., Watanabe, T., Kaoku, S., & Yamaguchi, T. (2019). Computer-aided diagnosis system for breast ultrasound images using deep learning. *Physics in Medicine & Biology*, *64*(23), 235013.
15. Raghavendra, U., Gudigar, A., Rao, T. N., Ciaccio, E. J., Ng, E. Y. K., & Acharya, U. R. (2019). Computer-aided diagnosis for the identification of breast cancer using thermogram images: A comprehensive review. *Infrared Physics & Technology*, *102*, 103041.
16. Visiopharm [Електронний ресурс]. – Режим доступу: https://visiopharm.com/
17. Augmentiqs.Main page [Електронний ресурс]. – Режим доступу: https://www.augmentiqs.com/
18. Leicabiosystems [Електронний ресурс]. – Режим доступу: https://www.leicabiosystems.com/digital-pathology/

19. Arterys. Breast AI [Електронний ресурс]. – Режим доступу: https://www.arterys.com/breast-women-radiology-ai-platform
20. Epredia . Digital Pathology [Електронний ресурс]. – Режим доступу: https://epredia.com//digital-pathology-workflow/
21. Pitsun O. Graphical interface of hybrid intelligent systems for biomedical imaging analysis / Y. Batko, G. Melnyk, O. Pitsun // Proceedings of the 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP) 23-27 August, 2016, Lviv, Ukraine, pp. 121-124 doi: 10.1109/DSMP.2016.7583521
22. Berezsky, Oleh, Oleh Pitsun, Grygoriy Melnyk, Tamara Datsko, Ivan Izonin, and Bohdan Derysh. 2023. "An Approach toward Automatic Specifics Diagnosis of Breast Cancer Based on an Immunohistochemical Image" Journal of Imaging 9, no. 1: 12. https://doi.org/10.3390/jimaging9010012
23. M. Testi et al., "MLOps: A Taxonomy and a Methodology," in IEEE Access, vol. 10, pp. 63606-63618, 2022, doi: 10.1109/ACCESS.2022.3181730.
24. Nazir, Sajid, Diane M. Dickson, and Muhammad Usman Akram. "Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks." Computers in Biology and Medicine (2023): 106668. https://doi.org/10.1016/j.compbiomed.2023.106668
25. Dhar, Tribikram, Nilanjan Dey, Surekha Borra, and R. Simon Sherratt. "Challenges of Deep Learning in Medical Image Analysis—Improving Explainability and Trust." IEEE Transactions on Technology and Society 4, no. 1 (2023): 68-75. https://doi.org/10.1109/TTS.2023.3234203
26. Vega, Carlos, Miroslav Kratochvil, Venkata Satagopam, and Reinhard Schneider. "Translational challenges of biomedical machine learning solutions in clinical and laboratory settings." In International Work-Conference on Bioinformatics and Biomedical Engineering, pp. 353-358. Cham: Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-07802-6_30
27. Bennetot, Adrien, Ivan Donadello, Ayoub El Qadi, Mauro Dragoni, Thomas Frossard, Benedikt Wagner, Anna Saranti et al. "A Practical guide on Explainable

AI Techniques applied on Biomedical use case applications." arXiv preprint arXiv:2111.14260 (2021). http://dx.doi.org/10.2139/ssrn.4229624

28. Granlund, Tuomas, Vlad Stirbu, and Tommi Mikkonen. "Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product." SN computer Science 2, no. 5 (2021): 342. https://doi.org/10.1007/s42979-021-00726-1
29. Reddy, Manjunatha, Brahmanand Dattaprakash, Sandesh Kammath, Subramanya Kn, Sumathra Manokaran, and Rangaswamy Be. "Application of mlops in prediction of lifestyle diseases." ECS Transactions 107, no. 1 (2022): 1191. DOI 10.1149/10701.1191ecst
30. Jain, Archit, Adarsh Malviya, Disha Bajaj, Revati Bhavsar, and Amit Savyanavar. "Brain Tumor Detection using MLops and Hybrid Multi-Cloud." In 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), pp. 1-6. IEEE, 2022. https://doi.org/10.1109/ICBDS53701.2022.9936020
31. Stirbu, Vlad, Tuomas Granlund, and Tommi Mikkonen. "Continuous design control for machine learning in certified medical systems." Software Quality Journal 31, no. 2 (2023): 307-333. https://doi.org/10.1007/s11219-022-09601-5
32. Berezsky, Oleh, Oleh Pitsun, Grygory Melnyk, Yuriy Batko, Bohdan Derysh, and Petro Liashchynskyi. "Application Of MLOps Practices For Biomedical Image Classification." In IDDM, pp. 69-77. 2022.
33. Berezsky Oleh, Pitsun Oleh, Grygoriy Melnyk, Tamara Datsko, Ivan Izonin, and Bohdan Derysh. "An Approach toward Automatic Specifics Diagnosis of Breast Cancer Based on an Immunohistochemical Image." Journal of Imaging 9, no. 1 (2023): 12. https://doi.org/10.3390%2Fjimaging9010012
34. Berezsky, Oleh, Pitsun Oleh, Bohdan Derysh, Ihor Pazdriy, Grygory Melnyk, and Yuriy Batko. "Automatic segmentation of immunohistochemical images based on U-net architecture." In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, pp. 29-32. IEEE, 2021. https://doi.org/10.1109/CSIT52700.2021.9648669

35. Alexandrova, Sonya, Zachary Tatlock, and Maya Cakmak. "RoboFlow: A flow-based visual programming language for mobile manipulation tasks." In 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 5537-5544. IEEE, 2015. https://doi.org/10.1109/ICRA.2015.7139973
36. Ciaglia, Floriana, Francesco Saverio Zuppichini, Paul Guerrie, Mark McQuade, and Jacob Solawetz. "Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark." arXiv preprint arXiv:2211.13523 (2022). https://doi.org/10.48550/arXiv.2211.13523
37. Moreschini, S., Lomio, F., Hästbacka, D., & Taibi, D. (2022, March). MLOps for evolvable AI intensive software systems. In 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 1293-1294). IEEE. https://doi.org/10.1109/SANER53432.2022.00155
38. Kreuzberger, Dominik, Niklas Kühl, and Sebastian Hirschl. "Machine learning operations (mlops): Overview, definition, and architecture." IEEE Access (2023). https://doi.org/10.1109/ACCESS.2023.3262138
39. Kumara, Indika, Rowan Arts, Dario Di Nucci, Willem Jan Van Den Heuvel, and Damian Andrew Tamburri. "Requirements and Reference Architecture for MLOps: Insights from Industry." (2022).
40. Recupito, Gilberto, Fabiano Pecorelli, Gemma Catolino, Sergio Moreschini, Dario Di Nucci, Fabio Palomba, and Damian A. Tamburri. "A multivocal literature review of mlops tools and features." In 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 84-91. IEEE, 2022.
41. Зварич Р., Шимчук М.М. Адаптивний алгоритм попереднього оброблення зображень/Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі». 5 листопада 2024 р. Тернопіль. Україна- с. 107
42. Зварич Р. MLOPS ПІДХОДИ ДЛЯ ОПРАЦЮВАННЯ ЗОБРАЖЕНЬ / Всеукраїнська науково-практична конференція студентів, аспірантів та

молодих вчених «Інтелектуальні комп’ютерні системи та мережі». 5 листопада 2024 р. Тернопіль. Україна-с. 109