Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2021 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему:

МЕТОДИКА ОЦІНЮВАННЯ ВТРАТ ПІД ЧАС РЕСТАВРАЦІЇ ТВОРІВ МИСТЕЦТВА НА ОСНОВІ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

Виконав: студент групи МІТ - 41

Лавринович Владислав Валерійович
(прізвище ім'я по-батькові) (підпис)

Керівник: професор кафедри мережевих та інтернет технологій

д.т.н. Миколайчук Р.А.
(посада, прізвище ім'я по-батькові) (підпис)

Київ 2021

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2021 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Лавринович Владислав Валерійович
(прізвище, ім'я, по батькові)

1. Тема роботи:

«Методика оцінювання втрат під час реставрації творів мистецтва на основі технологій машинного навчання»

затверджена на засіданні кафедри МІТ «11» грудня 2020 р. протокол № 6
2. Термін здачі закінченої роботи «30» травня 2021р
3. Вихідні дані до проекту (роботи)

Застосунок для виконання порогування та алгоритму кластеризації з фільтрами

Програмне забезпечення та тренована модель згорткової нейронної мережі

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-50 стор.)

1. Аналіз задачі оцінювання втрат під час реставрації творів мистецтва на основі технологій машинного навчання.
2. Реалізація та застосування алгоритмів машинного навчання для оцінювання втрат при реставрації творів мистецтва.
3. Порівняння результатів роботи алгоритмів машинного навчання
4. Рекомендації та їх економічна оцінка

5. Перелік графічного матеріалу 8-12 слайдів

Процес реставрації.

Сегментація зображень.

Короткий опис методів.

Результати роботи алгоритмів та їх порівняння.

Висновки

Дата видачі завдання _____
Керівник роботи д.т.н., професор кафедри МІТ Миколайчук Р.А.
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	29.01.2021	
2	Розділ 1	01.03.2021	
3	Розділ 2	01.04.2021	
4	Розділи 3, 4	01.05.2021	
5	Доповідь та слайди	27.05.2021	
6	Пояснювальна записка	30.05.2021	

Здобувач вищої освіти _____
(підпис) (прізвище, ім'я, по батькові)

Керівник _____
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 71с., 32 рис., 1 табл., 3 додатки, 19 джерел.

Об’єкт дослідження: алгоритми машинного навчання для сегментації зображень.

Предмет дослідження: оцінювання втрат під час реставрації творів мистецтва за допомогою алгоритмів машинного навчання.

Мета роботи (проєкту): розробити програмне забезпечення для оцінювання втрат при реставрації творів мистецтва із застосуванням алгоритмів машинного навчання та оцінити результати роботи алгоритмів.

Методи дослідження: системний підхід, методи порівняння, експеримент, опис, узагальнення.

У спеціальній частині дана характеристика сучасного стану технологій машинного навчання та процесу реставрації.

В роботі проведено аналіз існуючих технологій та підходів до процесу реставрації, також описані сучасні методи сегментації зображень.

Запропоновано використати алгоритм порогування для ручного визначення відсотку втрат, алгоритм кластеризації як алгоритм машинного навчання без учителя, та нейронну згорткову мережу як алгоритм навчання з учителем.

Розроблено застосунок для виконання експериментів із пророгуванням та кластеризацією.

Розроблено програмне забезпечення для створення, тренування та тестування моделі нейронної мережі. Створено набір даних та промарковано дані за допомогою двох методів: порогування та вручну.

Практичне значення роботи полягає у створенні програмного забезпечення для автоматизації процесу визначення втрат при реставрації творів мистецтва.

Результати здійснених у дипломному проекті досліджень можуть бути використані на другому етапі в процесі реставрації, а також для оцінки пошкоджень творів мистецтва та вартості їх відновлення, створення звітів та багатьох інших процесів.

Галузь використання – реставраційна діяльність, художня діяльність, музейна сфера, аукціонна діяльність.

Напрямки подальшого розвитку роботи визначаються збільшенням кількості типів пошкоджень для розпізнання, урізноманітнення типів творів мистецтва для оцінки пошкоджень, оцінка вартості реставрації на основі оцінки пошкоджень.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, ОЦІНКА ВТРАТ, РЕСТАВАРАЦІЯ, НЕЙРОННА МЕРЕЖА, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КЛАСТЕРИЗАЦІЯ, ПОРОГУВАННЯ.

ЗМІСТ

Стор.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT – інформаційні технології.

ІС – інформаційна система.

CNN – Convolutional Neural Network (згорткова нейронна мережа).

KMC – K-means clustering (кластеризація K-середніх).

U-Net – Universal Network (універсальна мережа).

JDK – Java Development Kit (набір інструментів Java для розробника).

RGB – Red Green Blue (адаптивна кольорова модель).

ReLu – Rectified Linear unit (лінійна функція випрямлення).

ПК – персональний комп’ютер.

HSV – Hue, Saturation, Value (адаптивна кольорова модель).

CMYK – Cyan, Magenta, Yellow, Key (адаптивна кольорова модель).

NumPy – Numerical Python (бібліотека для роботи з великими масивами чисельних даних).

OpenCV – Open Source Computer Vision Library (бібліотека комп’ютерного бачення).

ML – Machine Learning (машинне навчання).

DL – Deep Learning (глибоке навчання).

ВСТУП

Інформаційні технології розвиваються дедалі більшим темпом і знаходять все більше галузей застосування. Сьогодні все більше людей користуються перевагами реалізації інформаційних технологій в повсякденному житті.

Архітектури та моделі роботи інформаційних систем які застосовуються в наш час дуже відрізняються одна від одної, і роблять більш гнучкою можливість застосування тих чи інших технологій в досить багатьох областях від науки та зброї до повсякденного життя.

Інформаційні системи в залежності від сфери застосування мають деякі особливості, притаманні специфіці роботи з даними. Така специфіка відображається у функціональних можливостях системи, і визначає область її застосування.

В наш час для автоматизації багатьох процесів де необхідний аналіз даних використовуються алгоритми машинного навчання. Такий підхід дозволяє частково замістити інтелект людини, дозволяючи інформаційній системі, виконувати деякі операції на основі вже проаналізованої інформації, та відображати результат.

На сьогоднішній день, процес реставрації складається з багатьох кроків які вимагають точності та правильної техніки. Застосування інформаційних систем в цій області може допомогти прискорити деякі частини реставрації, а також збільшити точність оцінки експертів, що приведе до зменшення витрат, і збільшення якості проведеної роботи.

Буде створена система машинного навчання яка буде визначати пошкоджені частини творів мистецтва та давати оцінку пошкодженням і таким чином реставратор буде бачити повну картину стану твору мистецтва.

Актуальність теми даної роботи обумовлена зростанням попиту та зацікавленості населення в художніх творах мистецтва, а також необхідністю збереження і покращення стану вже існуючих історичних цінностей для забезпечення збереження національної пам’яті та історичного спадку майбутніх поколінь.

1. АНАЛІЗ ЗАДАЧІ ОЦІНЮВАННЯ ВТРАТ ПІД ЧАС РЕСТАВРАЦІЇ ТВОРІВ МИСТЕЦТВА НА ОСНОВІ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

1.1 Опис та аналіз предметної області

Реставрація живопису як процес впливу на стан експонатів з метою збереження первісних рис витвору мистецтва та відтворення пошкоджених, вже досить давно є предметом наукових досліджень та інноваційної діяльності. На даний момент цифрові технології вже широко застосовуються в процесі консервації та реставрації творів мистецтва, а саме для вироблення необхідних умов зберігання: контроль температури, освітлення, вологості повітря, підтримання параметрів музейного мікроклімату та інших.

Під впливом багатьох чинників в часі, будь-який об’єкт мистецтва має тенденцію до старіння. Застосовуючи різні технології та методи, реставратори намагаються мінімізувати або усунути вплив таких чинників з метою збереження витвору мистецтва в його первісній формі з усіма притаманними йому рисами з метою збереження оригінальності.

Реставрація має декілька різних визначень. В буквальному сенсі реставрацію можна визначити як відновлення пошкоджених фрагментів, а в більш загальному – як комплекс заходів для відновлення стану експоната близького до первісного.

Техніка та методи збереження та реставрації мистецтва йдуть рука об руку і стали досить розвиненою галуззю у 20 столітті. Вони стають дедалі важливішим аспектом роботи не лише музеїв, а й громадянської влади та всіх, хто займається витворами мистецтва, будь то художники, колекціонери чи відвідувачі галерей.

Методи художньої реставрації, що використовувались у попередні періоди, були тісно пов'язані та обмежені відомими на той час техніками художнього виробництва. Досягнення науки та техніки та розвиток природоохоронної справи як професії у 20 столітті призвели до більш безпечних та ефективних підходів до вивчення, збереження та ремонту об’єктів.

Сучасна природоохоронна практика дотримується принципу оборотності, який диктує, що реставрація не повинна спричиняти постійні зміни об’єкта. Збереження мистецтва стало важливим інструментом дослідження; серед професійних консерваторів є звичайною практикою документування процедур фотографіями та письмовими звітами.

Не існує єдиного підходу до реставрації живопису, оскільки кожна робота повинна бути ретельно вивчена, щоб визначити, який процес та техніки найкраще підходять для твору. Підготовлені історики мистецтва, хіміки та вчені з матеріалознавства поєднують великі галузі знань, щоб оцінити оригінальні напрямки твору та визначити найменш інвазивне рішення для відновлення втрат. Загальна структура процесу відображена на рисунку 1.1.

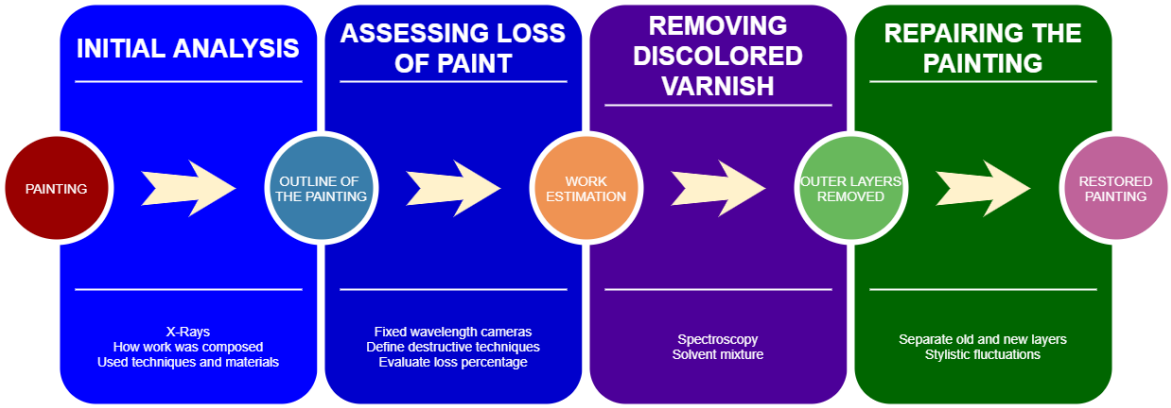


Рисунок 1.1 – Візуалізація етапів процесу реставрації живопису

Спочатку картина проходить початкову оцінку. Реставратори повинні бути добре знайомі зі стилем та періодом картини, яку вони оцінюють, оскільки ці знання допоможуть визначити техніку малювання, матеріали, доступні художникам у той час, та пігменти та тканини, які зазвичай використовувались. Рентген також виявляє, як була складена робота, що дозволяє консерватору сформулювати контур картини або роботи на основі різного поглинання фарби.

Після цього інфрачервоне зображення використовується для перегляду оригінальних малюнків та втрат фарби під поверхнею картини. Нещодавно технологічний прогрес у художній реставрації включив камери з фіксованою довжиною хвилі. Оскільки різні пігменти та матеріали по-різному відбивають або

поглинають різні довжини хвиль, ці пристрої можуть допомогти їх розрізнити. Вони дозволяють консерваторам точно визначати рисунки на основі вуглецю, наприклад, використовуючи відмінні довжини хвиль близько 1700 нанометрів. Це частина більш масштабного процесу, спрямованого на усунення раніше руйнівних методів та допомогу у визначенні шарів лаку.

Після створення точної картини оригінального малюнку наступним кроком є пошук відповідної суміші розчинників для видалення знебарвлених шарів лаку, якщо це можливо. Розвиток спектроскопії - методики, яка використовується для спостереження коливальних, обертальних та інших низькочастотних режимів у системі полегшив визначення точного складу та характеристик лаків.

Після визначення лаку та видалення зовнішніх шарів, можна приступати до відтворення пошкоджених частин картини. Прикладом одного із способів зробити це в сучасній практиці є наступний: проміжний шар лаку наноситься на оригінальну фарбу, щоб фізично відокремити нову фарбу від старої та забезпечити можливість будь-яких майбутніх реставрацій, не впливаючи на оригінальні шари. Консерватор ретельно фарбує пошкоджені ділянки, використовуючи сухий пігмент, змішаний із синтетичними розчинниками, щоб гарантувати, що професійно відновлена картина не потребувала подальшої консервації.

Також останнім часом з розвитком інформаційних технологій в цій області стала популярною віртуальна реконструкція. Ця технологія дозволяє візуалізувати можливий оригінальний стан пам'ятки. Це дозволяє оцінити можливість реставрації та результат ще до початку виконання робіт та витрат коштів.

Крім того візуальні дані від віртуальної реконструкції можна використовувати при реставрації та оформленні звітів, а також подальшій обробці даних про картину. У віртуальній реставрації застосовуються ті ж принципи, що і у фактичній, щоб уникнути розходжень з кінцевим результатом. Тим не менш, цифрова реставрація володіє низкою переваг, таких як дешевизна, можливість редагування даних, та перевірка гіпотез без ушкодження оригінального твору.

Як раніше було згадано, останнім часом інформаційні технології знаходять все більше застосування в процесах реставрації творів мистецтва. В першу чергу це стосується систем обліку, реєстрації та пошуку творів мистецтва.

Також досить широко інформаційні технології застосовуються у процесі віртуального ретушування за допомогою програми Blender. Blender широко застосовується для редагування фотографій із двовимірними зображеннями, і володіє перевагами цифрової кольорової ретуші.

На рис. 1.2 показано приклад ретуші в Blender для відтворення оригінальних рис витвору мистецтва.

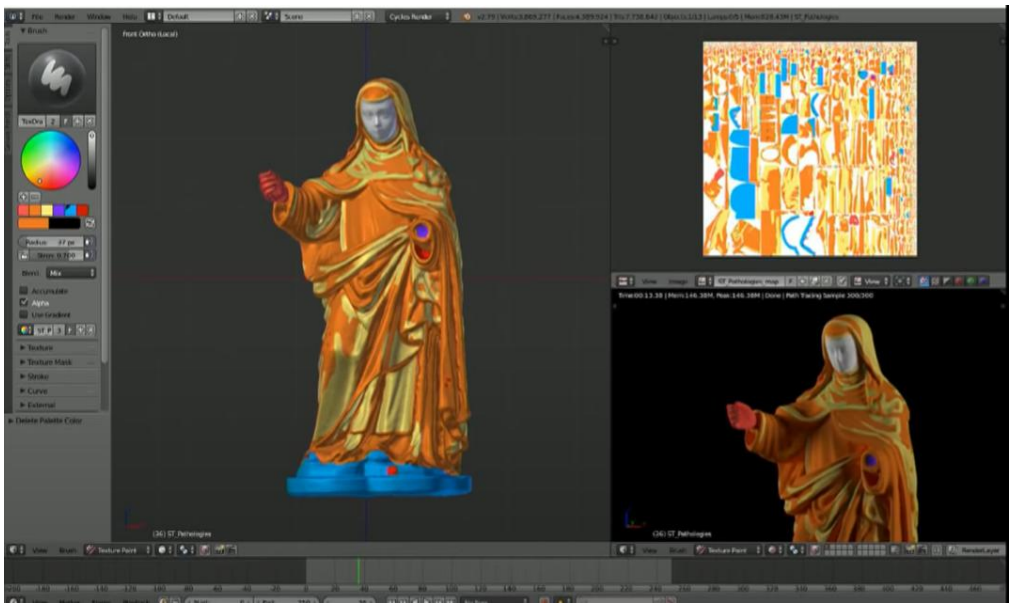


Рисунок 1.2 – Приклад ретуші в Blender

Інформаційні технології також часто застосовують для автоматизації обліку, пошуку, реєстрації та охорони творів мистецтва. Крім того, інформаційні технології використовуються для обробки даних досліджень та спеціальних аналізаторів (рентгенофлуоресцентних, портативних мікроскопів та ін.).

Незважаючи на досить широку область застосування інформаційних технологій в процесах консервації та реставрації творів мистецтва, в цій роботі особлива увага буде приділена другому етапу процесу реставрації зображеному на на рисунку 1.1, а саме – оцінка втрат зображення.

Одним з найбільш перспективних методів оцінки пошкоджень художніх творів мистецтва є використання методів аналізу даних. Алгоритми аналізу даних вимагають досить значного обсягу даних предметної області, в нашому випадку – живопису, який потребує реставрації з явними ознаками пошкоджень. Алгоритм аналізу даних складається в загальному випадку з наступних етапів:

- отримання даних;
- попередня обробка даних;
- навчання моделі;
- тестування точності тренованої моделі;
- візуалізація результатів передбачення моделі;

1.2 Аналіз існуючих підходів до вирішення задачі

Буде розглянуто кілька підходів до виявлення пошкоджених ділянок на старих картинах, які можна використовувати для оцінки дефектів, а також у цифровому процесі реставрації або на підготовчому етапі реконструкції тощо. Пошкоджені області та межі можна визначити за допомогою техніки сегментації зображень [1]. Різні методи сегментації зображень були успішно застосовані при виявленні бетонних тріщин [2] та при біомедичних виявленнях клітин [3, 4]. Також досить широко використовуються алгоритми кластеризації та класифікації та ряд інших.

Хоча для успішного навчання моделей потрібні тисячі оброблених зображень, досить часто вчені покладаються на аугментацію даних, щоб ефективніше використовувати наявні вибірки даних. Крім того, нейронні мережі демонструють високу швидкість навчання та належним чином прийняті результати прогнозування, витягнуті з дуже невеликої кількості зображень.

Спочатку слід визначити різні методи сегментації зображень, які можна використовувати для виявлення зон втрат. Зображення - це набір пікселів, отже, можна згрупувати їх разом, щоб визначити деякі подібні атрибути, використовуючи сегментацію зображення. Зазвичай згорткові нейронні мережі

застосовуються для вирішення проблем виявлення та класифікації об'єктів. Але це не дає уявлення про форму об’єкта. Можна отримати лише набір координат обмежувальної області (рисунок 1.3). У нашому випадку потрібно отримати точну форму області втрат, щоб мати можливість оцінити відсоток втрат на картині.

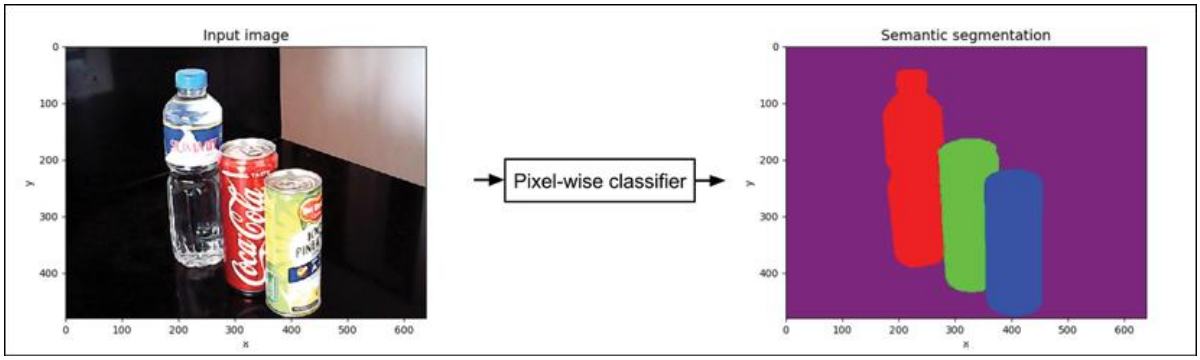


Рисунок 1.3 – Приклад сегментації зображення з виділенням доменів з подібними атрибутами

Сегментація зображення створює піксельну маску для кожного об’єкта на зображенні, тому можна отримати набагато більше інформації про форму та площу об’єкта. У нашому випадку це надзвичайно важливо, оскільки дозволяє оцінити відсоток втрат на зображенні.

Існує кілька методів сегментації зображень, таких як сегментація на основі областей, сегментація виявлення країв, сегментація кластеризації та згорткові нейронні мережі.

Метод сегментації на основі областей шукає подібності між сусідніми пікселями. Тобто пікселі, що мають подібні атрибути, згруповані в унікальні регіони. Як і у всіх методах сегментації, використання інтенсивності рівня сірого є найпоширенішим способом присвоєння подібності, але існують інші можливості, такі як дисперсія, колір та мультиспектральні особливості.

Методи сегментації областей кластеризують пікселі, що представляють однорідні ділянки зображення. Області кластеризуються шляхом групування сусідніх пікселів, властивості яких, такі як інтенсивність, відрізняються менше ніж деяка визначена величина. Кожному вирахуваному регіону на вихідному зображенні присвоюється унікальна ціла мітка. Цей клас алгоритмів, як правило,

добре працює для складних зображень, оскільки є адаптивним та менш сприйнятливим до наслідків часткової оклюзії, суміжності, шуму та неоднозначних меж.

У простих реалізаціях сегментація визначається одним параметром, відомим як поріг інтенсивності. За один прохід кожен піксель на зображенні порівнюється з цим порогом. Якщо інтенсивність пікселя перевищує порогове значення, для пікселя на виході встановлюється, скажімо, білий. Якщо воно менше порогового значення, для нього встановлюється чорний.

У більш складних реалізаціях можна вказати кілька порогових значень, так що для смуги значень інтенсивності можна встановити білий колір, а для всього іншого встановити чорний

Інший поширений варіант - встановити на чорний колір усі ті пікселі, що відповідають фону, але залишити пікселі переднього плану у вихідному кольорі / інтенсивності (на відміну від примусового їх білого кольору), щоб ця інформація не втрачалася. Приклади застосування порогування зображено на рисунку 1.4.

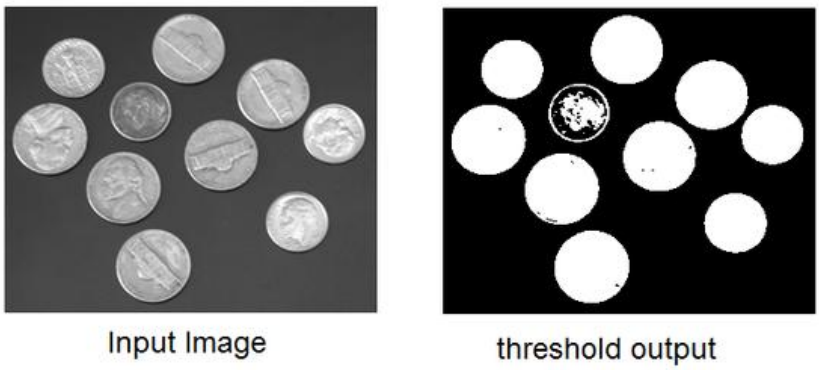


Рисунок 1.4 – Приклад застосування порогування.

Також досить поширеним є застосування алгоритмів класифікації та кластеризації для сегментації зображень. Класифікація включає широкий спектр теоретичних підходів до ідентифікації зображень (або їх частин).

Усі алгоритми класифікації базуються на припущенні, що на розглянутому зображенні зображено один або кілька ознак (наприклад, геометричні деталі у випадку виробничої системи класифікації або спектральні області у випадку дистанційного зондування) та що кожна з цих ознак належить до одного з кількох

різних та ексклюзивних класів. Класи можуть бути апріорі специфіковані аналітиком (як у контрольованій класифікації) або автоматично класифіковані (тобто як у класифікації без нагляду) у набори класів-прототипів, де аналітик просто вказує кількість бажаних категорій. (Класифікація та сегментація мають тісно пов’язані цілі, оскільки перша - це інша форма маркування компонентів, яка може призвести до сегментації різних ознак зображення.)

Класифікація зображень аналізує числові властивості різних характеристик зображення та упорядковує дані за категоріями. Алгоритми класифікації зазвичай використовують дві фази обробки: навчання та тестування. На початковому етапі навчання виділяються характерні властивості типових ознак зображення, і на їх основі створюється унікальний опис кожної категорії класифікації, тобто навчального класу. На наступному етапі тестування ці розділи простору об’єктів використовуються для класифікації об’єктів зображення.

Кластеризація K-середніх (K-means clustering, KMC) є одним із найбільш широко використовуваних алгоритмів кластеризації [6]. KMC працює, розбиваючи набір даних об’єктів (спостереження) на K різних кластерів (рис. 1.5).

K повинен бути вказаний користувачем перед запуском алгоритму. Зверніть увагу, що це метод навчання без нагляду, оскільки користувач не позначає окремі категорії / кластери. По-перше, алгоритм випадково створює K початкових центрів кластеру, які також називають насінням (seed). Далі він призначає всі інші об’єкти одному з цих насінин на основі близькості об’єкта-насіння.

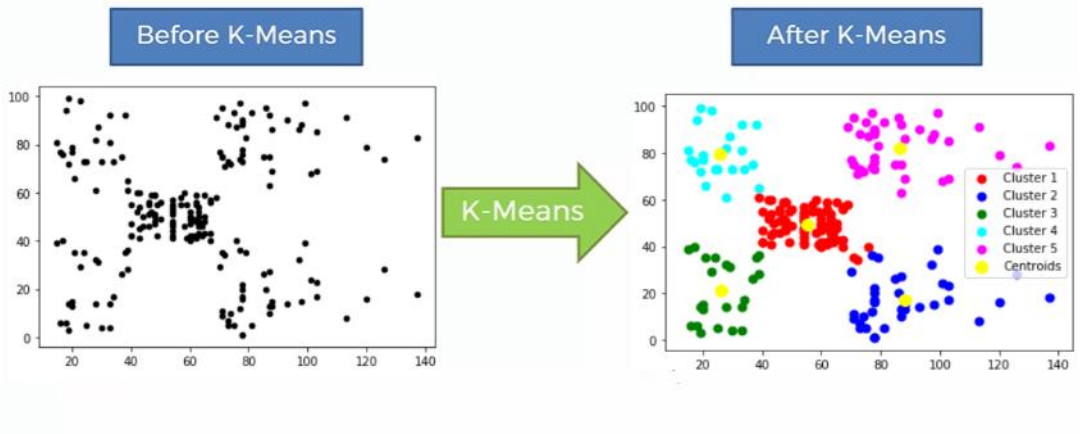


Рисунок 1.5 – Приклад KMC

Спочатку демаркаційні лінії між K кластерами проводяться шляхом нанесення перпендикулярних бісектрис між насінням. Потім обчислюються центроїди для кожного з новостворених K кластерів, і вони стають новими центрами кластерів. Процедура присвоєння об’єктів цим новим центрам кластерів повторюється, а центри кластерів K перераховуються. Вся ця процедура повторюється кілька разів і зупиняється, як тільки відстань між новим і попереднім K-центроїдами опускається нижче певного порогу.

Також досить популярною технікою є застосування згорткових нейронних мереж (Convolutional Neural Network, CNN) в сегментації зображень. CNN використовується для класифікації зображень. Вхідні зображення, які використовуються в CNN мають 3 виміри, тобто висоту, ширину та кількість каналів. Перші два виміри говорять про роздільну здатність зображення, а третій вимір представляє кількість каналів (RGB) або значення інтенсивності для червоного, зеленого та синього кольорів. Зазвичай зображення, які надходять у нейронну мережу, транслюються в розміри, які скорочують час обробки та уникають проблеми недонавчання. Загалом діаграму шарів згорткової мережі можна побачити на рисунку 1.6.

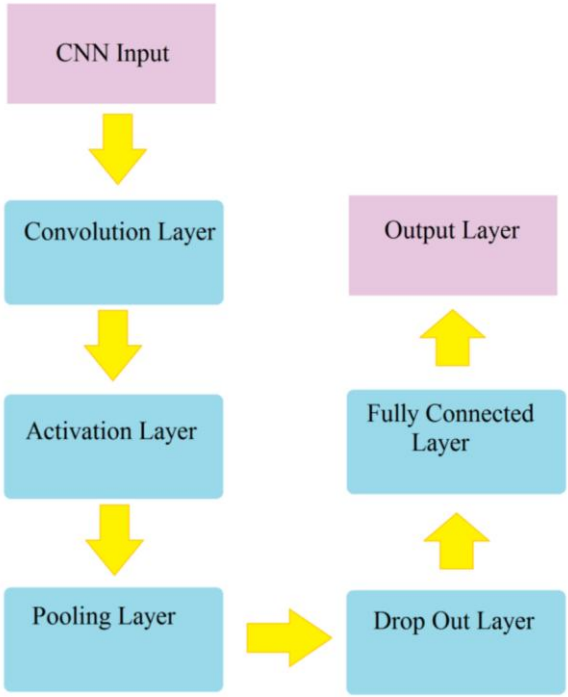


Рисунок 1.6 – Різні шари згорткової нейронної мережі

В згортковому шарі нейронної мережі використовуються фільтр або ядро для ітерації зображенням зображення через фіксований проміжок інтервалів, що називаються кроками. Вибір розміру кроку має вирішальне значення для досягнення бажаних результатів. Під час запуску фільтра над зображенням, обчислюється крапковий добуток фільтра з частиною зображення, на якому він лежить. Тоді сума всіх значень матриці результату дорівнює копіюється у відповідну позицію на згортковій карті об'єктів матриці.

Таким чином, може бути отримана карта зменшених розмірів зображення. Фільтри можуть бути різних видів, де використовується кожен фільтр щоб виділити з зображення різні види об’єктів.

Наприклад, один фільтр може бути відповідальним за вилучення одного виду функції з зображення на основі форм і країв, а інший фільтр може використовувати вилучення ознак на основі інтенсивності кольорів. Параметри, які допомагають регулювати ефективність роботи CNN:

Покроковий: визначає кількість пікселів на скільки здвинути наш фільтр на зображення, щоб можна було зосередитись на новому набір пікселів при виконанні згортки. Діапазони значень кроків від 1 до 3 залежно від величини похибки. Сума похибки у зображенні збільшується зі збільшенням значення кроку. Вибір оптимального значення кроку є дуже важливим завданням. Точність моделі сильно залежить від цього значення [7].

Набивання: це процес додавання нулів навколо межі вихідного зображення симетрично. Це допомагає отримати вихід карти об’єктів розміром відповідно до наших вимог. Зазвичай воно використовується для збереження розмірності зображення після згортки. Також досить часто застосовуються фільтри різних видів [8].

Об’єднуючий шар оперує невеликим ядром на зображенні з фіксованим кроком. Він використовується для вибору пікселя з найбільшою інтенсивністю та відкидання інших пікселів. Отримана матриця буде зменшеною розмірною матрицею зображення об’єкта. Це допомагає зменшити непотрібні розріджені

клітини зображення, від яких немає ніякої користі в класифікації. Максимальне об'єднання допомагає зменшити розмірність мережі (або зображення), але це може спричинити деяку втрату даних. Концепція, що стоїть за цим методом – що сусідні пікселі можуть бути апроксимовані максимальною інформацією несучого пікселя.

Шар активації здебільшого використовує ReLu як функцію активації. ReLu це функція, яка використовується для встановлення всіх від'ємних значень на нуль та зберігає позитивне значення таким, яке воно є. За цим кроком зазвичай слідує шар згортки та об'єднання. У численних галузях, таких як комп'ютерне бачення - CNN стають сучасним досягненням наближеним людської або кращої продуктивності.

Проектування CNN саме по собі є нелегким завданням [9]. До цього часу немає фіксованої формули для проектування CNN. Багато дослідників запропонували загальні пропозиції, але вони не завжди витримують критику, оскільки завдання залежить як від даних, так і від алгоритму.

Випадаючий шар зазвичай застосовується після шару, що містить нейрони в повністю зв'язаній мережі. Випадаючий шар – це шар регуляризації. З кожною ітерацією він випадково видаляє ваги деяких вхідних нейронів так, що інші нейрони отримують більше ваги в наступній ітерації. Зазвичай беруть шанс випадання 20-50%. За допомогою випадаючого шару, зазвичай продуктивність мережі збільшується на 1-2%.

Також повністю зв'язаний шар часто використовується в CNN. Структура мережі згори до низу зазвичай утворює піраміду, кількість параметрів у цих шарах сходиться, поки вони нарешті не досягнуть кількості бажаних класів. Збільшення кількості прихованих одиниць у шарі може підвищити здатність до навчання мережі, але існує певна межа насичення для підвищення точності мережі.

Більшість мереж в дослідженнях зазвичай добре себе показують з кількістю одиниць кратній 64. Два-три рівні мережі добре працюють, якщо для них є

достатня кількість шаблонів, яка передається в мережу після згладжування вихідних вагів згорткових шарів.

На рисунку 1.7 відображена внутрішня архітектура згорткової нейронної мережі. Зображення представлено матрицею значень пікселів і згортковою картою, яка отримується передавання і множення матриць фільтрів на вхідне зображення.

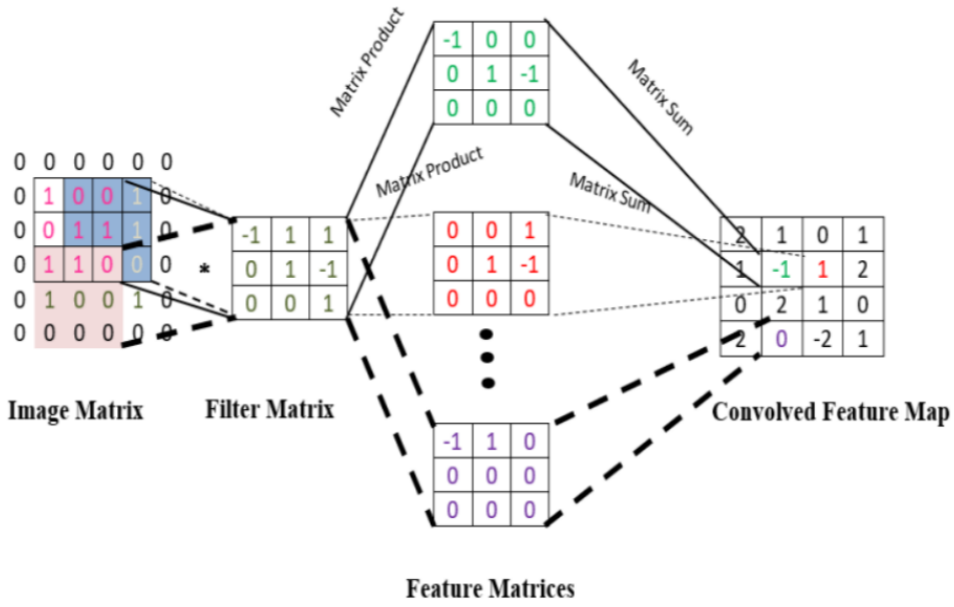


Рисунок 1.7 – Внутрішня архітектура згорткової нейронної мережі

1.3 Постановка завдання

У процесі реставрації класичних картин одним із завдань є відображення втрат фарби для документування та аналізу. Оскільки це досить значна і втомлива робота, автоматичні інструменти для вирішення цього завдання стають потрібними.

Наявні в даний час інструменти дозволяють лише грубо картографувати зони втрат фарби, в той же час вимагаючи значної ручної роботи. Буде розроблений метод навчання для виявлення втрат фарби, який використовує мультимодальні отримання зображень, і застосовується в рамках поточної реставрації відповідних зображень.

Наша архітектура нейронної мережі натхнена багатомасштабною згортковою нейронною мережею, відомою як U-Net. Особлива увага зосереджена на розробці надійного підходу з мінімальним втручанням користувача. Гарне трансферне навчання тут має вирішальне значення для того, щоб розширити застосування попередньо підготовлених моделей до картин, які не були включені до навчального набору, лише зі скромним додатковим перенавчанням.

Представлена стратегію попередньої підготовки, заснована на мультимодальному згортковому автокодері, таким чином модель точно налаштовується, із застосуванням до інших картин. Результати оцінюються, порівнюючи карти виявлених втрат фарби з анотаціями експертів вручну та порівнюючи фактично намальовані результати з фактичними фізичними реставраціями. Результати чітко вказують на ефективність запропонованого методу та його потенціал для сприяння процесам збереження та реставрації мистецтва.

Висновки до розділу 1

В цьому розділі було розглянуто різні етапи та особливості процесу реставрації. Також було проаналізовано сьогочасні тенденції впровадження інформаційних систем та технологій в автоматизацію процесу реставрації.

Була визначена ціль і поставлена основна задача цієї роботи – впровадження інформаційних технологій, а саме алгоритмів машинного навчання в процес виявлення пошкоджень на реставраційних картинах, а також виділені основні переваги застосування інформаційних систем в цій області.

Були проаналізовані різні способи та варіанти алгоритмів, можливих для застосування, охарактеризовані принципи їх роботи, а також різні деталі, які впливають на результат роботи алгоритмів. Були визначені основні техніки обробки зображень, які можуть бути використані в реалізації поставленого завдання, а також розкрите поняття основного інструменту, який буде застосований – сегментація зображень, її види та особливості кожного виду.

2. РЕАЛІЗАЦІЯ ТА ЗАСТОСУВАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ОЦІНЮВАННЯ ВТРАТ ПРИ РЕСТАВРАЦІЇ ТВОРІВ МИСТЕЦТВА

Цифровий аналіз живопису - це перспективна область досліджень, яка швидко привертає увагу з боку спільнот з обробки зображень та машинного навчання. Типові завдання включають ідентифікацію художника, виявлення підробки, виявлення тріщин, виявлення втрат фарби та віртуальне фарбування.

Втрата фарби часто є наслідком розшарування та висихання фарби внаслідок старіння, хоча грубе поводження також може це спричинити. При відновленні картини консерватори фарбують ділянки втрат фарби. Для цього потрібно знати і документувати точні зони втрат фарби. В даний час ця документація передбачає багато ручної роботи, оскільки наявне програмне забезпечення може дати лише грубу оцінку областей втрат фарби.

Таким чином, процес документування є досить втомливим, і йому допомогло б покращене автоматизоване картографування регіонів втрат фарби. Реставратори зазвичай виявляють ці регіони візуально, оглядаючи саму картину, а іноді також розглядаючи паралельно цифрові придбання в інших формах, таких як інфрачервоне та рентгенівське зображення. Ці додаткові методи візуалізації надають додаткову інформацію та дозволяють краще оцінити цікаві особливості, включаючи лакуни та більші втрати фарби.

2.1 Застосування методів машинного навчання для сегментації зображень

Сегментація - це процес кластеризації зображення у кілька когерентних субрегіонів відповідно до вилучених характеристик, наприклад, атрибутів кольору або текстури, і класифікації кожного субрегіону в один із попередньо визначених класів. Сегментація також може розглядатися як форма стиснення зображення, яка є надзвичайно важливим кроком у виведенні знань із зображень.

Загалом, методи сегментації поділяються на дві категорії (тобто, контрольовані та без нагляду).

У парадигмі сегментації без нагляду залучена лише структура зображення. Зокрема, некеровані методи сегментації покладаються на аналіз інтенсивності або градієнта зображення за допомогою різних стратегій, таких як порогове значення, вирізання графіка, виявлення країв та деформація для окреслення межі цільового об'єкта на зображенні.

Такі підходи добре працюють, коли межі чітко визначені. Тим не менше, градієнтні методи сегментації чутливі шумів та артефактів, що призводять до відсутності або дифузії меж об’єкта. Графічні моделі - ще один клас некерованих методів сегментації, які стійкі до шумів, але часто їх застосування супроводжується високими обчислювальними витратами через ітераційну схему для підвищення результатів сегментації в кілька етапів. Навпаки, контрольовані методи сегментації включають попередні знання про обробку зображення за допомогою навчальних зразків.

Як приклад алгоритму машинного навчання без нагляду буде використаний алгоритм кластеризації K-середніх, а як алгоритм навчання з учителем буде використана нейронна мережа типу U-net. Крім того, для перевірки результатів буде використаний ручний алгоритм обробки зображень порогування.

2.2 Застосування алгоритму порогування для ручного оцінювання втрат

Порогування - це найпростіший метод сегментації зображень. З зображення сірого, можна використовувати порогове значення для створення двійкового зображення. Бінарні зображення створюються з кольорових зображень шляхом сегментації. Сегментація - це процес присвоєння кожного пікселя вихідного зображення для двох або більше класів. Якщо є більше двох класів, то звичайним результатом є кілька двійкових зображень. В обробці зображень, порогове значення використовується для розбиття зображення на менші сегменти, використовуючи принаймні один колір або значення сірого для визначення їх

межі. Перевага отримання спочатку двійкового зображення полягає в тому, що воно зменшує складність даних і спрощує процес розпізнавання та класифікації.

Найпоширенішим способом перетворення зображення сірого у двійкове зображення є вибір одного порогового значення (T).

Вхідними даними для порогової операції, як правило, є зображення сірого або кольорове зображення. У найпростішій реалізації результат є двійковим зображенням, що представляє сегментацію. Чорні пікселі відповідають фоновому, а білі пікселі відповідають передньому плану (або пошкодженням, або навпаки). Цей метод сегментації застосовує єдиний фіксований критерій до всіх пікселів на зображенні одночасно.

Сегментація зображення розділяє зображення на (безперервні) області або набори пікселів. Пікселі розподіляються залежно від інтенсивності їх значення.

g (x, y) = 1, якщо f (x, y) - піксель переднього плану (або заданої характеристики), g (x, y) = 0, якщо f (x, y) - піксель фону. Приклад гістограми для порогування зображений на рисунку 2.1.

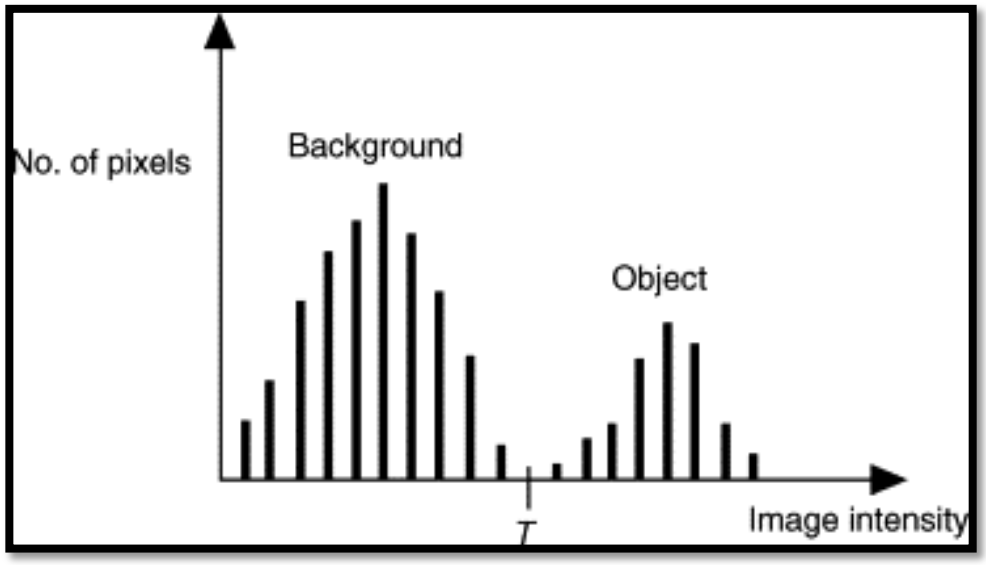


Рисунок 2.1 – Приклад гістограми інтенсивності зображення, де **T** – заданий поріг для порогування

У реальних додатках гістограми є більш складними, з великою кількістю піків та неясними долинами, і не завжди легко вибрати значення T. Ця техніка може бути виражена як:

$$T=T[x, y, p(x, y), f(x, y] \quad (2)$$

Де T - порогове значення. x, y - координати точки порогового значення. p(x, y), f (x, y) - точки пікселів рівня сірого. Порогове зображення g(x, y) можна визначити за формулою (1).

Існує три типи алгоритмів порогування:

– глобальне;

– локальне;

– адаптивне;

У нашому дослідженні буде використаний алгоритм глобального порогування, оскільки він дозволяє динамічно змінювати значення порогу для всього зображення і, таким чином краще визначати поріг для знаходження областей втрат на зображенні.

2.3 Реалізація алгоритму порогування на мові Java

Для реалізації порогування буде використана мова Java, з JDK версії 8.0.211,
а також інтегроване середовище розробки IntelliJ IDEA від JetBrains. Програма буде реалізована у вигляді додатку для ПК, в який можна буде завантажувати зображення, обирати поріг та виконувати алгоритм з заданим порогом, а результати будуть відображені в додатку разом з попередньою оцінкою втрат.

Крім того, результати будуть збережені в окремі зображення, для можливості подальшого порівняння. Також будуть реалізовані опції різних фільтрів, таких як ерозія та розмивання.

Саму функціональність порогування можна взяти з бібліотеки opencv версії 4.1.1 [10], в якій реалізовано більшість сучасних алгоритмів для обробки зображень. Бібліотека написана на мові C++, але opencv надає пітримку для Java

через окремий .jar архів, який імпортується як стороння бібліотека разом з C++ .dll бібліотекою [11].

Для реалізації інтерфейсу ПК додатку буде використовуватися фреймворк Java FX версії 11.0.1, який надає інтерфейс для реалізації додатків цього типу. Рисунок 2.2 зображує структуру проекту, де можна побачити імпортовані бібліотеки opencv_java411.dll та адаптер для мови Java opencv-411.jar. Інтерфейс визначається в файлі sample.fxml, який насправді є модифікованим xml форматом для Java FX [12].

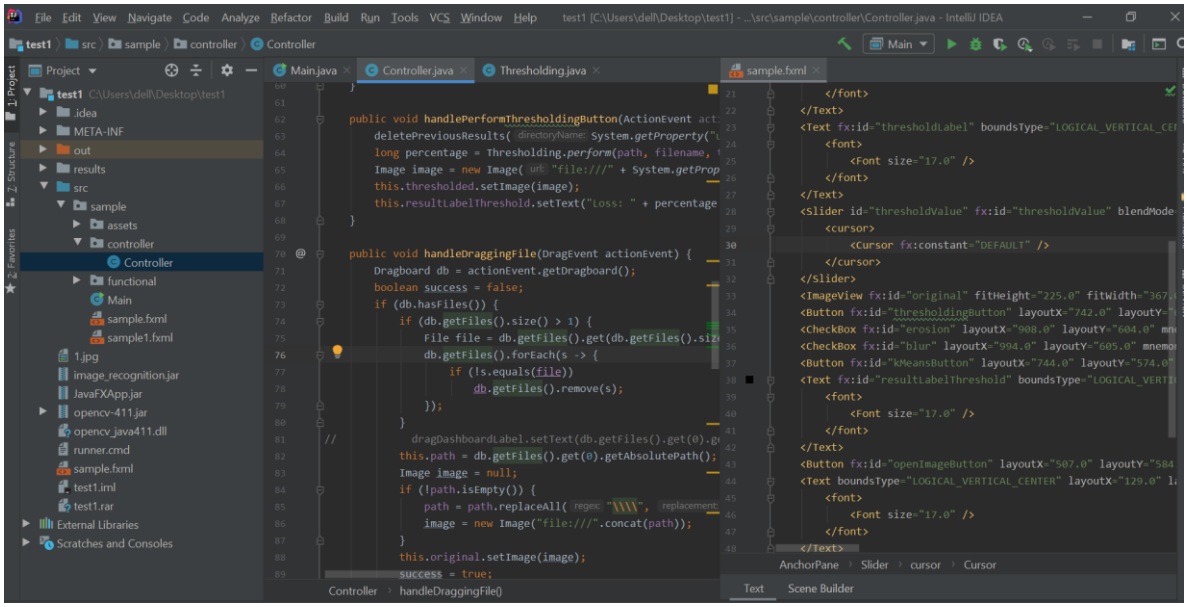


Рисунок 2.2 – Структура проекту

В папці out знаходяться вже скомпільовані в бінарники java-класи, які збираються в jar архів який безпосередньо і запускається. В папці assets знаходяться тестові зображення, які будуть використовуватись для оцінки втрат. Ці зображення є частиною картини, яка роздроблена на менші зображення для збільшення точності і спрощення проведення експериментів. Клас Controller відповідає за взаємодію з користувачем та обробку різних подій, а клас Main відповідно завантажує sample.fxml який надалі перетворюється в інтерфейс, а також вручну завантажує сторонню бібліотеку, оскільки opencv_java411.dll не є java-бібліотекою і має бути завантажена додатково.

Завантаження відбувається за допомогою передання значення статичного поля класу org.opencv.core.Core в метод, який завантажує бібліотеку в Runtime

наступним чином: System.loadLibrary(Core.NATIVE_LIBRARY_NAME).Таким чином завантажується бібліотека C++ в Java [13].

На рисунку 2.3 зображено інтерфейс додатку з функціональністю.

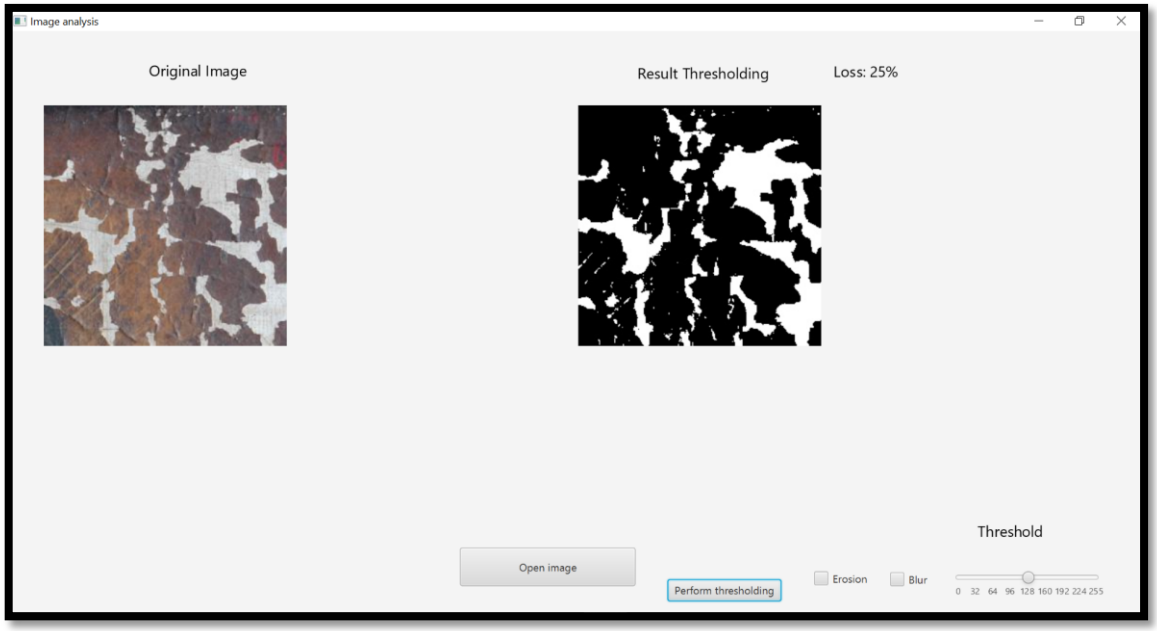


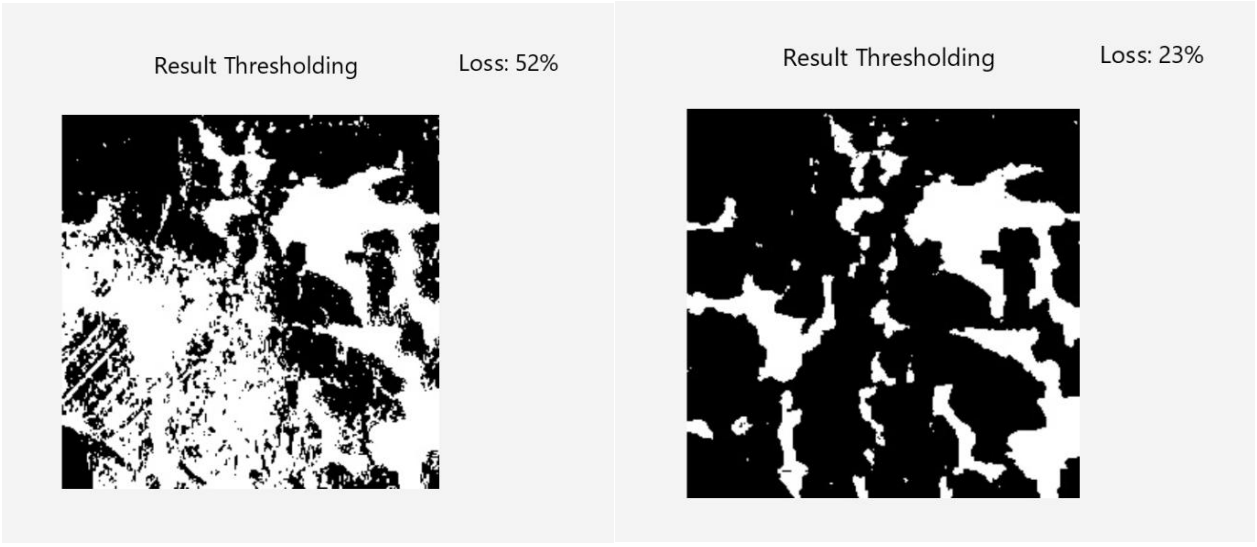
Рисунок 2.3 – Інтерфейс додатку для ручного визначення відсотку втрат за допомогою порогування, з порогом в 130

Як видно з рисунку, в лівій частині зображено вихідне зображення, де білі зони – це втрати фарби оригінальної картини. В правій зоні зображений результат порогування і відсоток втрат. Нижче є функціональні кнопки для завантаження зображення, виконання порогування, також є чекбокси для операцій розмивання та ерозії і слайдер для вибору порогового значення. В залежності від порогового значення, обраного вручну, відсоток втрат зміниться. Таким чином можна візуально визначити наскільки правдивим є результат в порівнянні з оригінальним зображенням. На рисунку 2.4 зображено порогування з пороговим значенням

а) 100, б) 150. Згідно алгоритму порогування встановлюється порогове значення пікселів а сірому зображенні.

Всі пікселі, значення яких більше порогового стають білими, а значення яких менші – чорними. Потім рахується кількість пікселів та відношення білої

частини до всього зображення, і таким чином отримується відсоток втрат, необхідний для оцінки пошкоджень під час реставрації. Порівнюючи порогування зі значеннями порогів в 100, 130 (рис.2.3) та 150 (рис. 2.4) можна дійти висновку, що залежно від порогу досить вагомо може змінитися відсоток втрат та область, яка вважається втратами, тому треба візуально визначати який поріг найбільш точний.



а) б)

Рисунок 2.4 – Результат порогування з пороргами а) 100 та б) 150

Також доступна опція ерозії під час виконання порогування. Ерозія є одним із двох основних операторів у галузі математичної морфології, іншим є розширення. Зазвичай він застосовується до двійкових зображень, але є варіації, які працюють із зображеннями у відтінках сірого. Основним ефектом оператора на двійковому зображенні є стирання меж областей пікселів переднього плану (тобто, як правило, білих пікселів). Таким чином ділянки пікселів переднього плану зменшуються в розмірі, а отвори в цих областях стають більшими.

Оператор ерозії приймає два фрагменти даних як вхідні дані. Перший - це зображення, яке слід розмити. Другий - це (як правило, невеликий) набір координатних точок, відомий як елемент структурування (також відомий як ядро). Саме цей елемент структурування визначає точний вплив ерозії на вхідне зображення. Математичне визначення ерозії для двійкових зображень є таким:

Припустимо, що X - набір евклідових координат, що відповідає вхідному двійковому зображенню, а K - набір координат для елемента структурування. Нехай Kx позначає трансформацію K так, щоб його початок був у x. Тоді ерозія X на K є просто набором усіх точок x, таких, що Kx є підмножиною X.

Для обчислення ерозії двійкового вхідного зображення ядром 3 × 3, розглядається кожен з пікселів переднього плану у вхідному зображенні по черзі. Для кожного пікселя переднього плану (який називається вхідним пікселем) накладається елемент структурування поверх вхідного зображення так, щоб початок елемента структурування збігався з координатами вхідного пікселя. Якщо для кожного пікселя в структуруючому елементі відповідним пікселем на зображенні внизу є піксель переднього плану, то вхідний піксель залишається таким, яким він є. Якщо будь-який з відповідних пікселів на зображенні є фоновим, для вхідного пікселя також встановлюється значення фону.

Для нашого прикладу елемента структурування 3 × 3 ефект цієї операції полягає у видаленні будь-якого пікселя переднього плану, який не повністю оточений іншими білими пікселями (припускаючи 8-зв’язок). Такі пікселі повинні лежати по краях білих областей, і тому практичним результатом є те, що регіони переднього плану зменшуються (і дірки всередині області зростають).

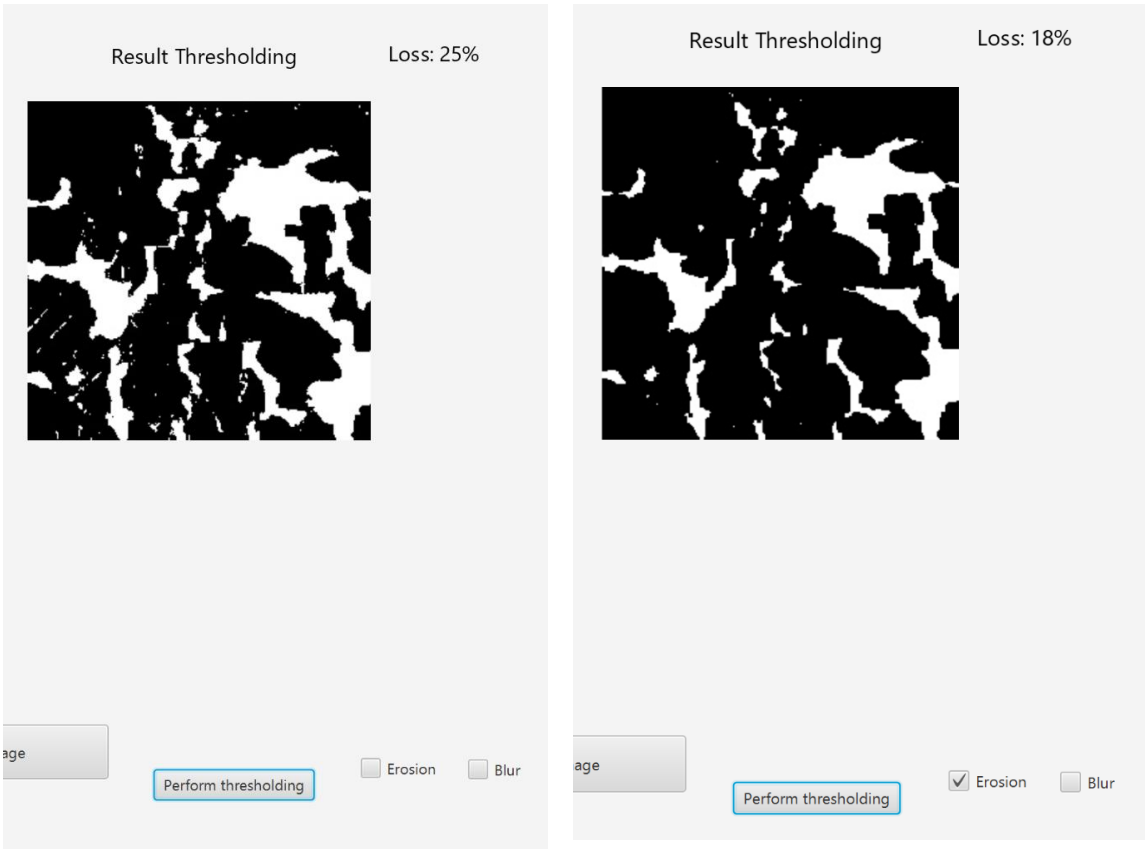
Ерозія є подвійним розширенням, тобто ерозія пікселів переднього плану еквівалентна розширенню фонових пікселів. Більшість реалізацій цього оператора очікує, що вхідне зображення буде двійковим, як правило, з пікселями переднього плану зі значенням інтенсивності 255, а пікселі фону - зі значенням інтенсивності 0. Таке зображення часто можна створити із зображення у градаціях сірого, використовуючи порогове значення. Важливо перевірити, чи правильно налаштована полярність вхідного зображення для використовуваної реалізації ерозії.

Можливо, елемент структурування повинен бути поданий у вигляді невеликого двійкового зображення або у спеціальному форматі матриці, або він може бути просто підключений до реалізації і взагалі не вимагати визначення. В

останньому випадку зазвичай передбачається 3 × 3 квадратний елемент структурування.

Квадрат 3 × 3 - це, мабуть, найпоширеніший структуруючий елемент, що використовується в операціях ерозії, але можуть бути використані інші. Більший структуруючий елемент виробляє більш екстремальний ерозійний ефект, хоча зазвичай дуже подібні ефекти можуть бути досягнуті шляхом багаторазових ерозій із використанням меншого структуруючого елемента подібної форми. З більшими структуруючими елементами досить часто застосовується приблизно дисковий структуруючий елемент, на відміну від квадратного.

Ерозію можна використовувати для видалення невеликих фальшивих яскравих плям (сольовий шум) на зображеннях. Також можна використовувати ерозію для виявлення країв, зробивши ерозію зображення, а потім віднявши її від вихідного зображення, виділивши таким чином лише ті пікселі по краях об’єктів, які були видалені ерозією. В даній роботі в якості ядра використовується матриця 3 × 3. Приклад обробки зображення з використанням ерозії та без зображено на рисунку 2.5.



а) б)

Рисунок 2.5 – Результат порогування з використанням ерозії а) та без б) з пороговим значенням 130

Крім того, була реалізована можливість використання розмиття Гауса для видалення шумів. Гаусів оператор згладжування - це 2-D оператор згортки, який використовується для розмиття зображень та видалення деталей та шуму. Він використовує ядро, яке представляє форму Гаусового (дзвоноподібного) горба.

Згортка - це проста математична операція, яка є фундаментальною для багатьох поширених операторів обробки зображень. Згортка забезпечує спосіб "множення разом" двох масивів чисел, як правило, різних розмірів, але однакової розмірності, щоб отримати третій масив чисел тієї самої розмірності. Це може бути використано в обробці зображень для реалізації операторів, значення вихідних пікселів яких є простими лінійними комбінаціями певних значень вхідних пікселів.

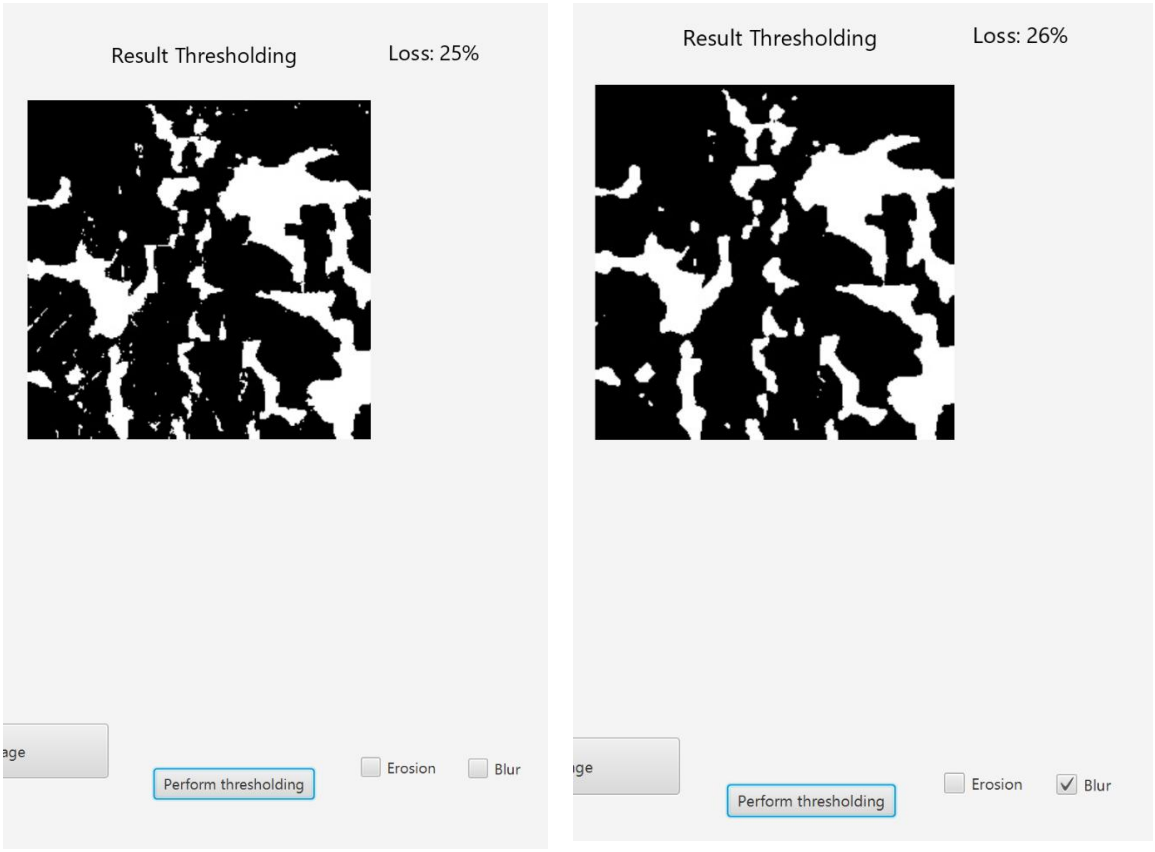
У контексті обробки зображень одним із вхідних масивів, як правило, є лише зображення рівня Грайля. Другий масив, як правило, набагато менший, а також є двовимірним (хоча може мати товщину лише в один піксель) і відомий як ядро.

Згортання здійснюється шляхом ковзання ядра по зображенню, як правило, починаючи з верхнього лівого кута, щоб перемістити ядро через усі позиції, де ядро повністю вкладається в межі зображення. (Зверніть увагу, що реалізації відрізняються тим, що вони роблять по краях зображень) Кожній позиції ядра відповідає один вихідний піксель, значення якого обчислюється множенням значення ядра та значення пікселя основного зображення для кожного клітинок ядра, а потім додавання всіх цих чисел разом.

Ідея Гаусового згладжування полягає у використанні двовимірного розподілу як функції розподілу точок, і це досягається згорткою. Оскільки зображення зберігається як сукупність дискретних пікселів, потрібно створити дискретне наближення до функції Гауса, перш ніж можна будет виконати згортку.

Теоретично, розподіл Гауса скрізь ненульовий, що вимагає нескінченно великого ядра згортки, але на практиці воно фактично дорівнює нулю більше, ніж приблизно три стандартних відхилення від середнього, і тому можна усікати ядро в цьому місці. Не очевидно, як вибрати значення маски для апроксимації Гауса. Можна використовувати значення матриці Гауса в центрі пікселя в масці, але це неточно, оскільки значення Гауса змінюється нелінійно по пікселю.

В цій роботі буде також використана розмиття Гауса для видалення шумів із зображення. Ядро для розмивання Гауса було взяте розміром 5 × 5. Результат порогування з використанням розмивання Гауса зображено на рисунку 2.6.



а) б)

Рисунок 2.6 – Результат порогування з використанням розмивання Гауса а) та без б) з пороговим значенням 130

2.4 Реалізація алгоритму кластеризації K-середніх на мові Java

Кластеризація - це метод поділу набору даних на певну кількість груп. Один із популярних методів є кластеризація k-середніх. Кластеризація k-середніх розділяє вибірку даних на k число груп даних.

Алгоритм класифікує заданий набір даних на k число неперервних кластерів. Алгоритм складається з двох окремих фаз. У першій фазі він обчислює k центроїд, а у другій фазі переносить кожну точку в скупчення, яке має найближчий центроїд з відповідної точки даних. Існують різні методи визначення відстані найближчого центроїда і одним із найбільш використовуваних методів є евклідова відстань. Після групування перераховується новий центроїд кожного кластера і на основі цього центроїда розраховується нова евклідова відстань між кожним центром і кожною точкою даних і вираховуються точки в вибірці, які мають мінімальну евклідову відстань.

Кожен кластер у вибірці визначається об'єктами які йому належать і його центроїдом. Центроїд для кожного кластера - це точка, сума відстаней всіх об’єктів цього кластера до якої, зведена до мінімуму. Отже, K-середніх - це ітераційний алгоритм, в який мінімізує суму відстані від кожного об'єкта до його центрального кластера по всіх кластерах.

Нехай є зображення з роздільною здатністю x × y, яке має бути згруповано в k кластерів. Нехай p(x, y) є вхідними пікселями, які будуть кластеризуватися, і c_k - центри кластерів. Алгоритм кластеризації k-means визначається як:

1. Ініціалізувати кількість кластерів k та їх центри.
2. Для кожного пікселя зображення обчислити евклідову відстань d між центром та кожним пікселем зображення використовуючи відношення, наведене нижче

$$d = \|p(x,y) - c_k\| \quad (3)$$

3. Призначити всі пікселі до найближчого центру залежно від відстані d.
4. Після призначення всіх пікселів перерахувати нове положення центру, використовуючи відношення, подане нижче

$$c_k = \frac{1}{k}\sum_{y \epsilon c_k}\sum_{x \epsilon c_k} p(x,y) \quad (4)$$

5. Повторити процес, поки він не задовольнить мінімальне допустиме значення похибки.
6. Перетворити кластери пікселів в зображення

В бібліотеці яка використана для порогування також наявна реалізація алгоритму кластеризації K-середніх, тож все що потрібно зробити – додати в наш застосунок ще одну функцію – кластеризації. Крім того визначається два кластери, перший – це саме зображення, інший – це втрати на зображенні. Слід зауважити що для ефективної роботи цього методу, пошкодження зображення мають досить добре вирізнятися від фонового зображення. Також можна було б додати ще третій кластер – риси та фігури на самому зображенні, але експерименти показали що кластеризація з трьома кластерами не дає точних результатів, тому було вирішено зупинитися на двох кластерах.

Незважаючи на те, що k-середніх має велику перевагу в тому, що його легко впровадити, він має деякі недоліки. Якість остаточного результату кластеризації залежить від довільного вибору початкового центроїда. Отже, якщо початковий центроїд вибраний випадково, він отримає різний результат для різних початкових центрів.

Також обчислювальна складність - це ще один термін, який потрібно враховувати при проектуванні кластеризації K-середніх. Він покладається на кількість елементів даних, кількість кластерів та кількість ітерацій. В нашому випадку у нас 2 кластери, і при зображенні розмірністю m × n у нас буде m * n ітерацій.

Інтерфейс застосунку з функцією кластеризації зображений на рисунку 2.7

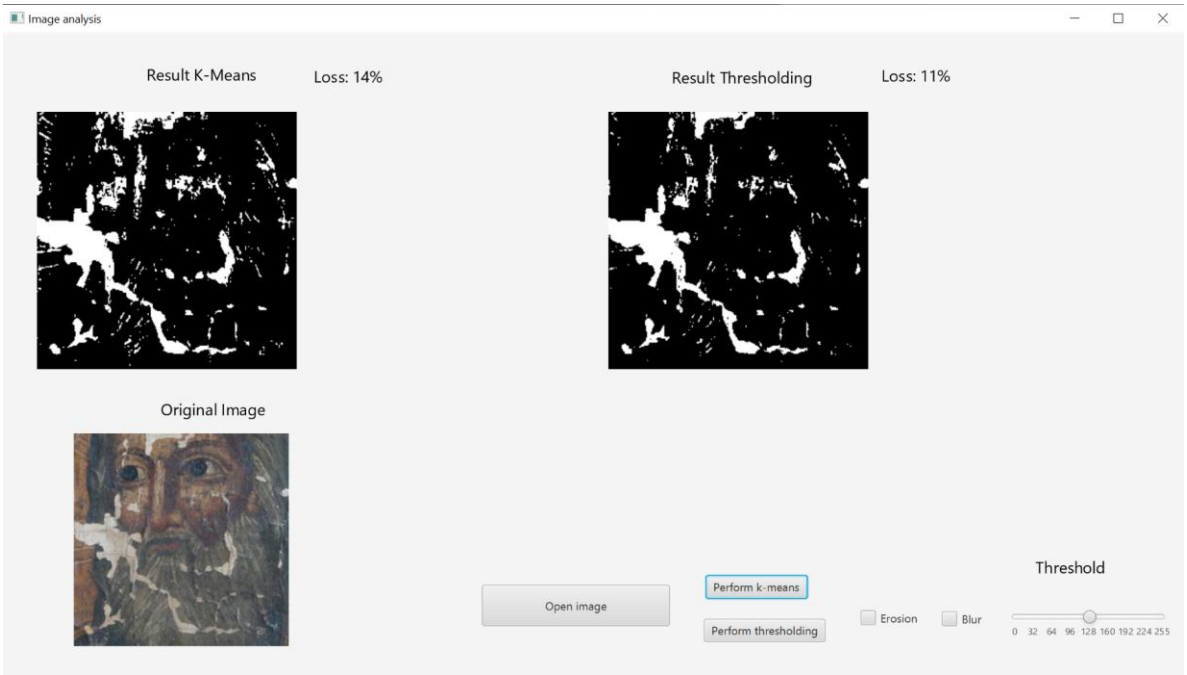


Рисунок 2.7 – Інтерфейс програми з можливістю кластеризації

2.5 Реалізація нейронної мережі типу U-Net на мові Python

Згорткова нейронна мережа (ConvNet / CNN) - це алгоритм глибокого навчання, який може приймати вхідне зображення, призначати важливість (зважувані ваги та упередження) різним аспектам / об'єктам на зображенні та мати можливість відрізняти один від іншого. Попередня обробка, необхідна в ConvNet, набагато нижча порівняно з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри розробляються вручну, при достатній підготовці, ConvNet має можливість вивчати ці фільтри / характеристики.

Архітектура ConvNet аналогічна архітектурі зв’язку нейронів в мозку людини і натхненна організацією зорової кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відомому як рецептивне поле. Колекція таких полів перекривається, щоб охопити всю зорову зону.

Зображення - це не що інше, як матриця значень пікселів. То чому б просто не згладити зображення (наприклад, матрицю зображення 3x3 у вектор 9x1) і не

подати його на багаторівневий перцептрон для класифікації як зображено на рисунку 2.8? Насправді все не так просто.

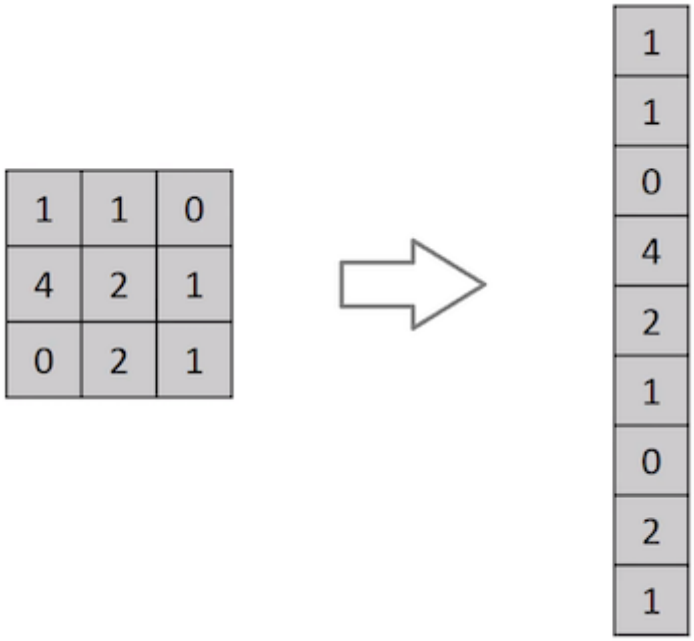


Рисунок 2.8 – Представлення матриці значень піксель у вигляді вектору

У випадках надзвичайно базових двійкових зображень метод може показувати середню оцінку точності під час виконання прогнозування класів, але коли мова йде про складні зображення, що мають піксельні залежності, точність буде прямувати до нуля.

ConvNet здатний успішно фіксувати просторові та тимчасові залежності в зображенні за допомогою відповідних фільтрів. Архітектура краще адаптується до набору даних зображення завдяки зменшенню кількості задіяних параметрів та повторному використанню вагів. Іншими словами, мережу можна навчити краще розуміти витонченість зображення.

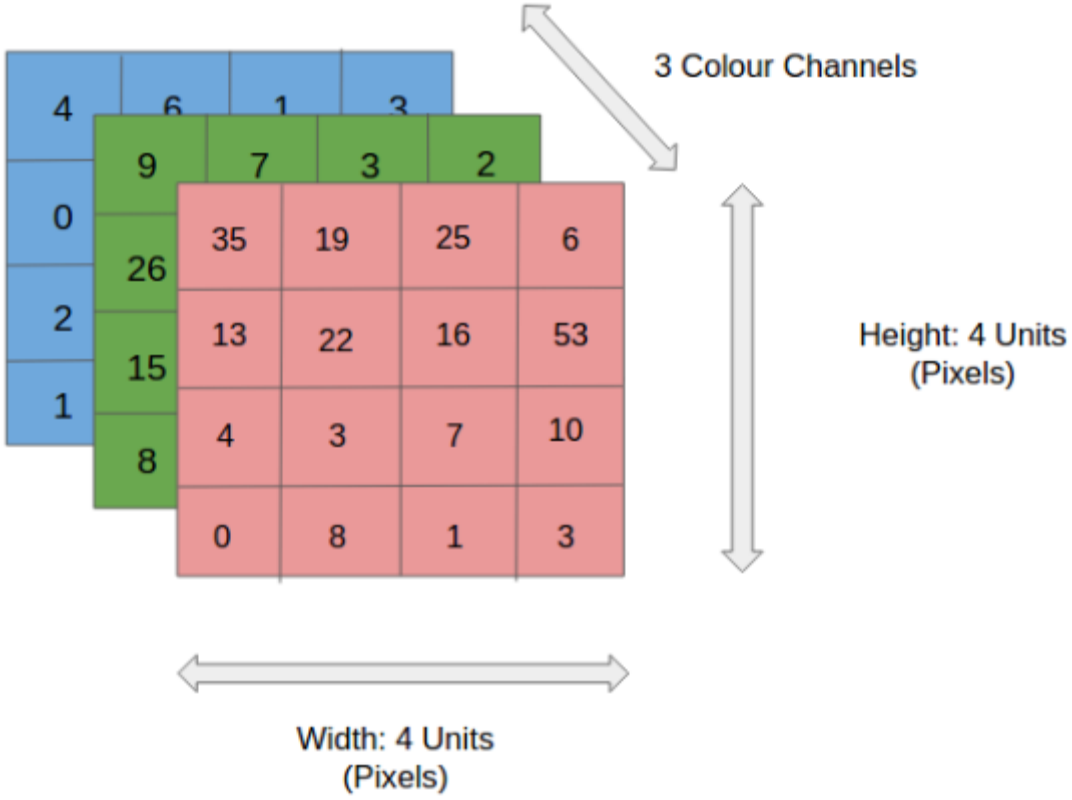


Рисунок 2.9 – Представлення RGB зображення у вигляді трьох матриць відповідних розмірностей

На рисунку 2.9 можна побачити зображення RGB, яке розділене трьома кольоровими площинами - червоною, зеленою та синьою. Існує ряд таких колірних просторів, в яких існують зображення - відтінки сірого, RGB, HSV, CMYK тощо.

Ви можете собі уявити, наскільки обчислювально затратними будуть операції, коли зображення досягнуть розмірів, скажімо, 8K (7680 × 4320). Роль ConvNet полягає у зменшенні зображень у форму, яку легше обробити, не втрачаючи особливостей, які є критично важливими для отримання хорошого передбачення. Це важливо, коли потрібно розробити архітектуру, яка не тільки добре володіє функціями навчання, але й масштабована до масивних наборів даних.

Згортковий шар є основним будівельним блоком CNN. Він несе основну частину обчислювального навантаження мережі.

Цей шар виконує точковий добуток між двома матрицями, де одна матриця - це набір параметрів, що вивчаються, інакше відомих як ядро, а інша матриця -

обмежена частина рецептивного поля. Ядро просторово менше зображення, але є більш глибоким. Це означає, що якщо зображення складається з трьох (RGB) каналів, висота та ширина ядра будуть просторово малими, але глибина сягає всіх трьох каналів. На рисунку 2.10 зображена операція згортки.

Під час проходу, ядро ковзає по висоті та ширині зображення, що створює нове зображення, яке представляє цю сприйнятливу область. Нове двовимірне зображення називається карта активації і представляє відповідь ядра на кожне просторове положення зображення. Розмір ковзання ядра називається кроком.

Згортковий шар нейронної мережі втілює три важливі ідеї дослідників комп'ютерного бачення: розріджена взаємодія, спільний доступ до параметрів та еквіваріантне представлення. Давайте опишемо кожен із них детально.

Тривіальні шари нейронної мережі використовують множення матриць на матрицю параметрів, що описують взаємодію між вхідним та вихідним блоком. Це означає, що кожна вихідна одиниця взаємодіє з кожною вхідною одиницею.

Однак нейромережі згортки мають розріджену взаємодію. Це досягається завдяки вибору розміру ядра меншим, ніж розмір вхідної матриці, наприклад, зображення може мати мільйони або тисячі пікселів, але, обробляючи його за допомогою ядра, можна виявити значущу інформацію, яка становить десятки або сотні пікселів. Це означає, що потрібно зберігати менше параметрів, що не тільки зменшує потребу пам'яті в моделі, а й покращує статистичну ефективність моделі.

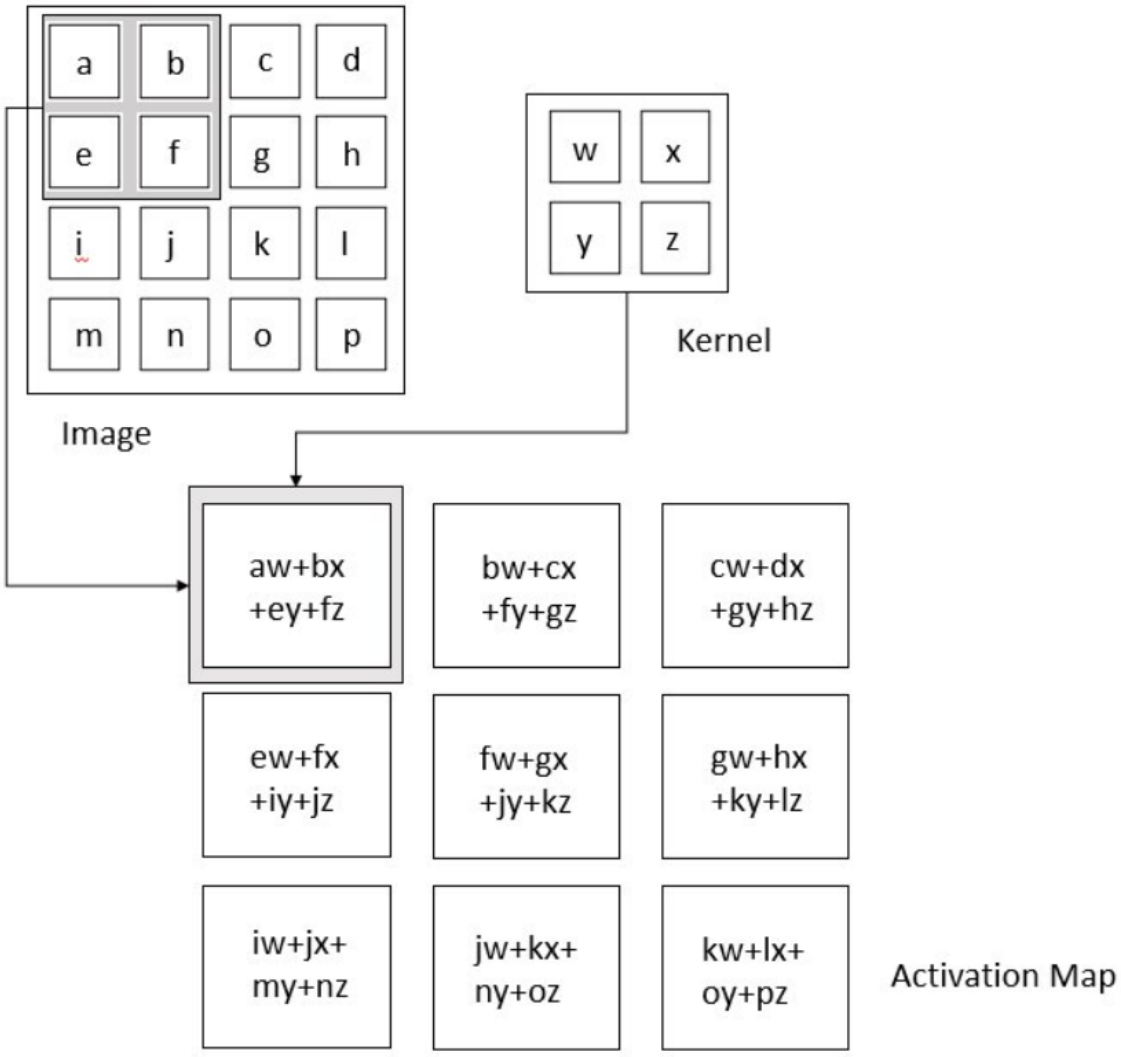


Рисунок 2.10 – Основна операція згорткового шару

Якщо існує обчислення однієї ознаки в просторовій точці (x1, y1), то також повинно бути існування ознаки і в іншій просторовій точці, скажімо (x2, y2). Це означає, що для одного двовимірного зрізу, тобто для створення однієї карти активації, нейрони мають бути обмеженими однаковим набором вагів. У традиційній нейронній мережі кожен елемент вагової матриці використовується один раз, а потім більше ніколи не переглядається, тоді як мережа згортки має спільні параметри, тобто для отримання вихідних даних ваги, застосовані до одного входу, ті ж самі, які й застосовуються у інших місцях. Завдяки спільному використанню параметрів, рівні нейронної мережі згортки матимуть властивість еквівалентності до трансляції. Тобто, якщо були змінені вхідні дані певним чином, вихідні дані також зміняться відповідно.

Рівень об'єднання замінює вихідні дані мережі в певних місцях шляхом виведення зведеної статистики найближчих виходів. Це допомагає зменшити просторовий розмір подання, що зменшує необхідну кількість обчислень і вагів. Операція об'єднання обробляється на кожному зрізі представлення окремо.

Існує кілька функцій об'єднання, такі як середнє значення прямокутного сусідства, норма L2 прямокутного регіону та середньозважене середнє на основі відстані від центрального пікселя. Однак найпопулярнішим процесом є максимальний пул [14], який повідомляє про максимальний вихід із сусідства.

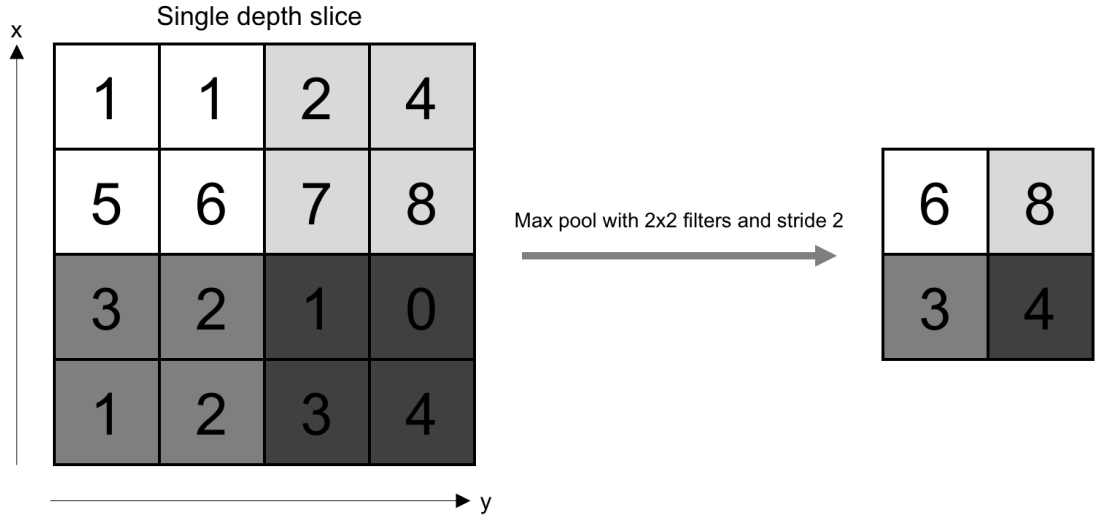


Рисунок 2.11 – Операція об’єднання

У всіх випадках застосування, об’єднання забезпечує певну незмінність трансляції, що означає, що об’єкт можна було б впізнати незалежно від того, де він з’являється на кадрі.

Оскільки згортка є лінійною операцією, і зображення далекі від лінійних, нелінійні шари часто розміщуються безпосередньо після згорткового шару, щоб внести нелінійність на карту активації.

Існує кілька типів нелінійних операцій, серед яких популярні:

1. Sigmoid
2. Tanh
3. ReLU

Сигмоїдна нелінійність має математичний вигляд $\sigma(к) = 1 / (1 + e^{-к})$. Він приймає дійсне число і "стискає" його в діапазон від 0 до 1. Однак якщо

локальний градієнт стане дуже малим, то при зворотному розповсюдженні він ефективно "вб'є" градієнт. Крім того, якщо дані, що надходять у нейрон, завжди позитивні, то на виході сигмоїда будуть або всі позитивні, або всі негативні, що призведе до зигзагоподібної динаміки оновлення градієнта ваги.

Функція тангенсу стискає дійсне число до діапазону [-1, 1]. На відміну від сигмоподібних нейронів - її вихід має нульовий центр.

Випрямлений лінійний блок (ReLU) став дуже популярним за останні кілька років. Він обчислює функцію $f(к) = \max(0, к)$. Іншими словами, активація - це просто порогування в нуль.

У порівнянні з сигмовидною та тангенсом, ReLU є більш надійним і прискорює конвергенцію в шість разів.

На жаль, є недолік, що ReLU може бути крихким під час навчання. Великий градієнт, що протікає через нього, може оновити його таким чином, що нейрон ніколи не буде оновлюватися далі. Однак можна це обійти, встановивши належний рівень навчання.

При побудові нашої нейронної мережі, буде використовуватись функція активації ReLU, та функція об’єднання максимального пулу, яка описана на рисунку 2.11.

Розпізнавання з використання нейронної мережі є досить обчислювально затратним процесом, тому він не буде входити в додаток, оскільки розпізнавання займає досить багато часу. Сама мережа, як і супутні необхідні функціональні частини будуть написані на мові Python версії 3.6. Це обумовлено тим, що в мові Python відсутня строга типізація, що дозволяє більш гнучко працювати з матрицями зображень, а також тим, що на базі мови Python вже існує досить багато бібліотек для роботи з нейромережами, що значно полегшить реалізацію та виконання задачі.

Конфігурація нейромережі буде зроблена з використанням модулю keras версії 2.3.1, а для роботи з матрицями та масивами даних буде використаний модуль numpy версії 1.18.4.

Keras - це API глибокого навчання, написаний на Python, що працює на платформі машинного навчання TensorFlow. Він був розроблений з акцентом на швидкі експерименти. Можливість якнайшвидшого переходу від ідеї до результату є ключовою для хороших досліджень [15].

NumPy - це бібліотека Python, яка забезпечує багатовимірний об'єкт масиву, різні похідні об'єкти (наприклад, масковані масиви та матриці) та асортимент підпрограм для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції з фігурами, сортування, вибір, введення / виведення, дискретні перетворення Фур'є, основна лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого [16].

Також потрібно створити датасет та конфігурувати позначки втрат на яких наша нейромережа буде навчатися. Для конфігурації та попередньої обробки зображено буде використовуватися адаптер бібліотеки OpenCV для Python версії 4.2.0. Для зображень які досить добре піддаються порогуванню він буде використовуватись для отримання лейблів, а для інших будуть позначені втрати вручну за допомогою додавання білих шарів на зображення та чорного шару фону і видалення оригінального зображення на веб сайті https://online-photoshop.org.

Слід зауважити що точність ручної обробки даних дещо менша ніж автоматизованого порогування. Також сильний акцен робиться на аугментації даних, оскільки для досягнення точності в 98% потрібний датасет з 60 000 зображень, який фізично не можна зробити вручну. Тож аугментація допоможе різноманітнити датасет і збільшити кількість екземплярів.

На рисунку 2.12 зображений приклад зображення яке входить в датасет для тренування. На рисунку 2.13 а) показаний приклад ручного лейблу для цього зображення, яке відображає втрати фарби. Лейбл зроблений вручну з використанням фотошопу. На рисунку 2.13 б) показаний результат порогування.

Але не можна використовувати це зображення як лейбл для датасету, оскільки шкала кольорів зображена як частина втрат, що є некоректним.

Слід зауважити, що хоча позначення похибки в порогуванні для даного зображення точніше, і можна було б обрізати частину з палітрою і застосувати такий варіант в якості лейблу, в даній роботі це розглядатися не буде.

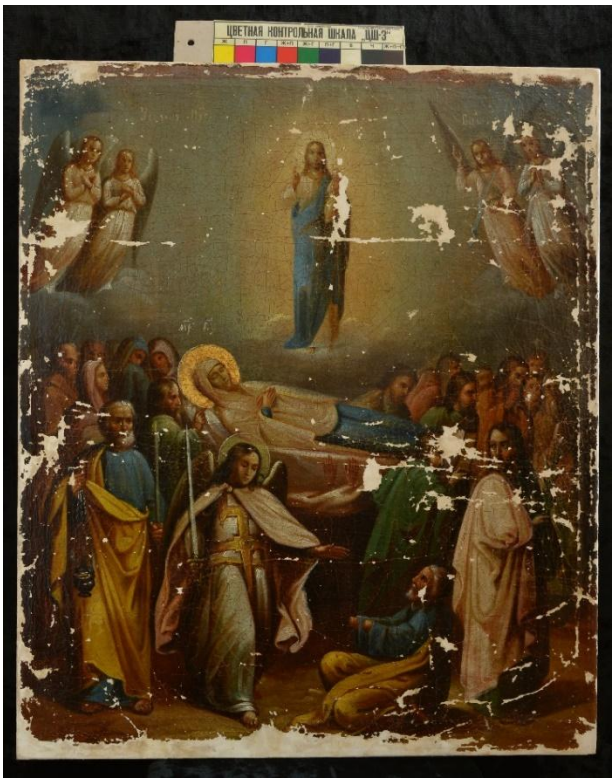
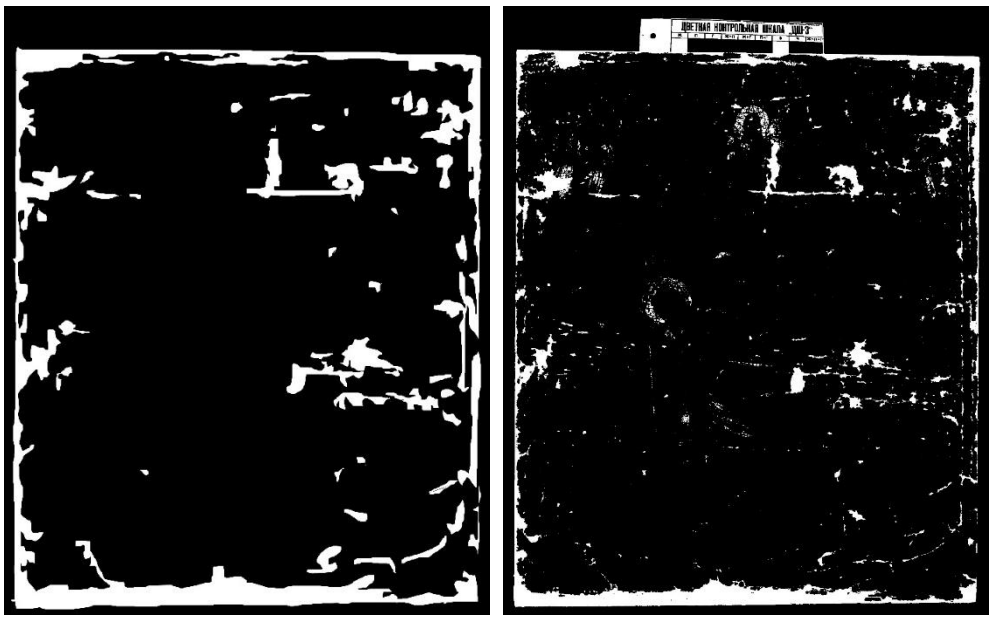


Рисунок 2.12 – Оригінальне зображення

Як видно на рис. 2.14, в порівнянні з порогуванням, ручне позначення втрат є не таким точним, зате якщо придивитись, то можна побачити, що шкала кольорів відображена як втрати врезультаті порогування, а при ручному позначенні цього немає, і таким чином нейромережа не буде навчатися на наперед некоректному датасеті.

Після цього застосовується аугментація для оригінального зображення і лейблу, і врезультаті отримуються нові зразки для тренування. Приклад такого зразка зображений на рисунку 2.15. Після аугментації зображення дещо змінюється. Аугментація - це техніка, яку можна використовувати для штучного збільшення розміру навчального набору шляхом створення модифікованих даних із існуючого.



а) б)

Рисунок 2.14 – Втрати позначені а) – вручну в фотошопі, б) – результат порогування

Хорошою практикою є використання аугментації, якщо ви хочете запобігти перенавчанню, або якщо початковий набір даних занадто малий для тренування, або навіть якщо ви хочете отримати кращу продуктивність із вашої моделі.

Аугментація використовується не лише для запобігання перенавчанню [17]. Загалом наявність великого набору даних має вирішальне значення для роботи як моделей ML, так і Deep Learning (DL). Однак можна покращити ефективність моделі, збільшивши вже наявні дані. Це означає, що збільшення даних також корисно для підвищення продуктивності моделі.

Рисунок 2.15 – Результат аугментації

Висновки до розділу 2

В цьому розділі були реалізовані різні методи машинного навчання для знаходження втрат фарби на творах мистецтва. Також були досліджені різні операції математичної морфології, застосовуючи які можна досягти певних успіхів в збільшенні точності при обчисленні результатів.

Також була розглянута та реалізована згорткова нейронна мережа для сегментації на мові Python з використанням відповідних бібліотек. Були проаналізовані різні аспекти які впливають на навчання та передбачення нейронної мережі, а також принцип її роботи.

Також був створений датасет для навчання мережі та застосована техніка агументації для збільшення розмірів датасету. Були проаналізовані недоліки трешолдингу при створенні лейблів та створені лейбли для зображення вручну за допомогою фотошопу.

В наступних розділах будуть проаналізовані результати роботи алгоритмів машинного навчання, їх якість та точність. Будуть порівняні результати та зроблені деякі підсумки, а також проаналізовані перспективи подальших досліджень.

3. ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

3.1 Порівняння результатів роботи алгоритму кластеризації K-середніх та порогування.

Тепер можна порівняти результати порогування та кластеризації залежно від застосування різних фільтрів. На рисунку 3.1 зображено порівняння результатів порогування та кластеризації без фільтрів.

Порівнюючи з оригінальним зображенням на рисунку 2.7 можна чітко визначити, що точність алгоритму K-середніх є дещо нижчою в порівнянні з ручним порогуванням, тим не менш ситуація дещо змінюється із застосуванням фільтрів.

Різниця між результатами обробки зображень на рисунку 3.1 дорівнює трьом відсоткам, не так вже і багато, але навіть візуально дуже добре видно різницю.

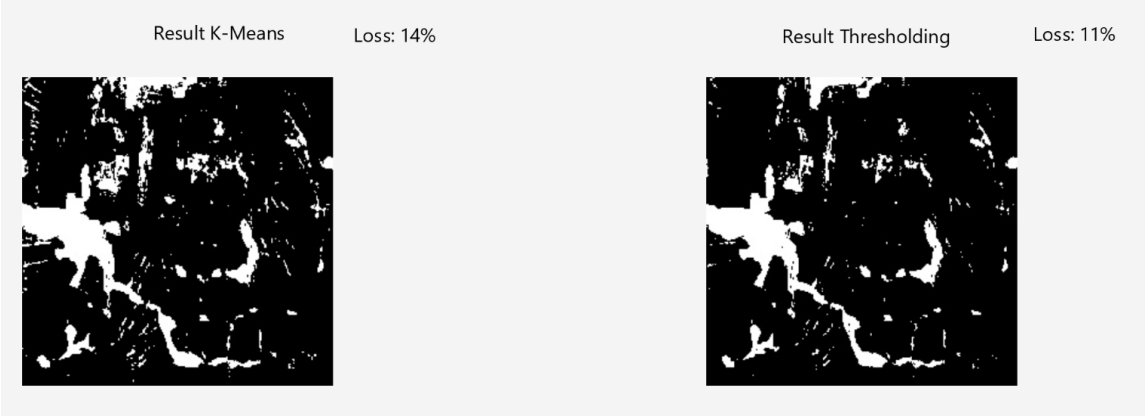


Рисунок 3.1 – Порівняння результатів порогування та алгоритму кластеризації без фільтрів.

На рисунках 3.2 і 3.3 можна побачити результати алгоритмів із застосуванням ерозії та розмиття Гауса. В обох випадках чітко можна побачити підвищення точності роботи алгоритмів і зменшення шумів, при чому розмиття крім зменшення шумів також дещо збільшує області втрат.

Це пояснюється тим, що перехідні зони під час розмиття стають ширшими і світлішими, тому алгоритми їх також розпізнають як втрати.

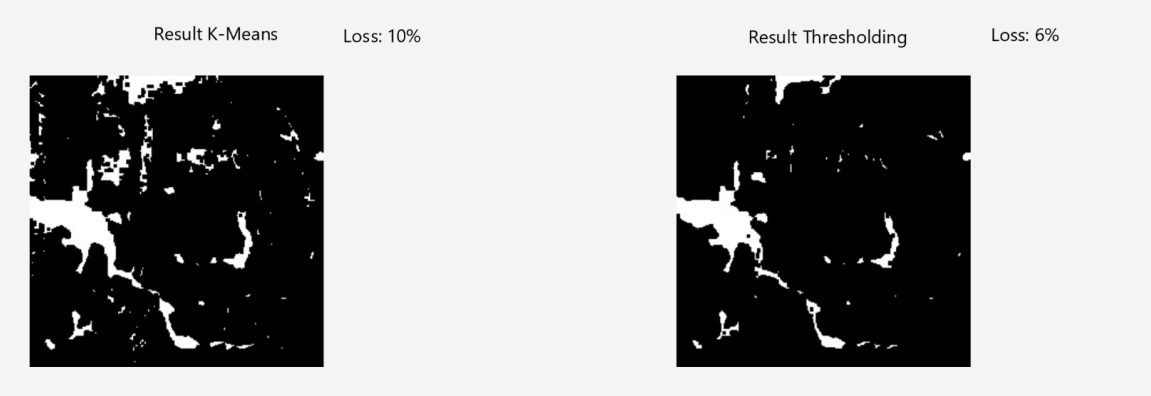


Рисунок 3.2 – Порівняння результатів порогування та алгоритму кластеризації з застосуванням ерозії.

Рисунок 3.3 – Порівняння результатів порогування та алгоритму кластеризації з застосуванням розмиття Гауса.

Різниця між результатами обробки зображень на цих рисунках вже дорівнює чотирьом відсоткам, що є наслідком застосування фільтрів і зменшення шумів.

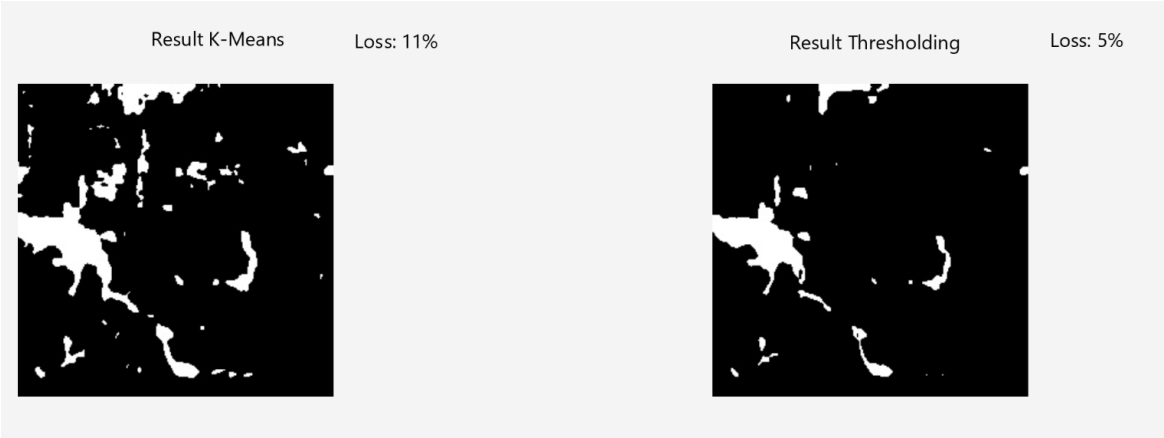


Рисунок 3.5 – Порівняння результатів порогування та алгоритму кластеризації з застосуванням ерозії та розмиття Гауса

На рисунку 3.5 зображені результати порогування та кластеризації з застосуванням обох фільтрів. При цьому різниця втрат вже дорівнює шістьом відсоткам. З цього порівняння можна зробити висновок про те, що що більше фільтрів застосовується, тим більшим стає розрив між результатами ручного порогування та автоматичної кластеризації.

На основі проведеного аналізу можна сказати, що для порогування потрібно попередньо обробити зображення, щоб отримати високу точність. Методи порогування гарну ефективність, якщо гістограма має бімодальний розподіл. Через той факт, що об'єкт не такий великий у порівнянні з фоном області, гістограма іноді не зберігає бімодальності.

Одне з найбільших обмежень порогування полягає в тому, що воно розглядає лише два класи в гістограмі але в більшості випадків існує більше двох класів пікселів в цифровому зображенні, тому в такому випадку метод порогування не підходить для сегментації.

Метод сегментації зображень K-середніх - це техніка, яка не використовує гістограми для процесу сегментації. Ось чому шуму, введеного в зображення, можна уникнути. В K-середніх оцінюються різні пікселі та подібні пікселі, або набори даних згруповані разом. Оскільки K-середніх підходить для великих наборів даних, і він працює на сферичних кластерах, він є досить ефективним і гнучким для змін. У порівнянні з іншими методи сегментації K-середніх підхід до

сегментації зображення є швидким та ефективним з точки зору вартості обчислень.

Одним з головних недоліків підходу K-середніх є рівномірний ефект, який відноситься до результатів кластерів однакових розмірів, тоді як вхідні дані можуть бути різного розміру. У K-середніх також важко передбачити початкове значення k, яке потрібно вказати у першій фазі алгоритму.

3.2 Порівняння результатів роботи нейромережі та порогування.

Слід зауважити, що порогування не є алгоритмом машинного навчання, а лише методом сегментації. сегментації.

В даному порівнянні порогування використовується як ручний метод визначення втрат фарби для зручності перевірки точності роботи інших алгоритмів. При цьому порогування обмежене зображеннями які однорідні і мають чітко виражені області втрат.

Для початку порівняємо результати роботи нейромережі та порогування на зображення, які добре підходять для операції порогування.

Рисунок 3.6 – Оригінальне зображення

На рисунку 3.6 зображено оригінальне зображення яке буде використовуватись для порівняння. На рисунку 3.7 а) відображено результат

порогування, а на рисунку 3.7 б) – передбачення нейронної мережі. При цьому слід зауважити, що результати відрізняються на один відсоток.

Така невелика різниця підкреслює доволі високу точність передбачення нейронної мережі. Таким чином на зображенні яке досить добре підходить для порогування, нейромережа та порогування дають приблизно однакові результати.

Тепер візьмемо зображення, яке досить погано піддається порогуванню. Це означає що фон картини нерівномірний, а втрати фарби не так легко сегментувати. На рисунку 3.8 а) зображено оригінальне зображення яке буде використане для подальшого порівняння.

а) б)

Рисунок 3.7 – Результат а) – порогування з порогом 130, 24% втрат, б) – передбачення нейронної мережі, 25% втрат.

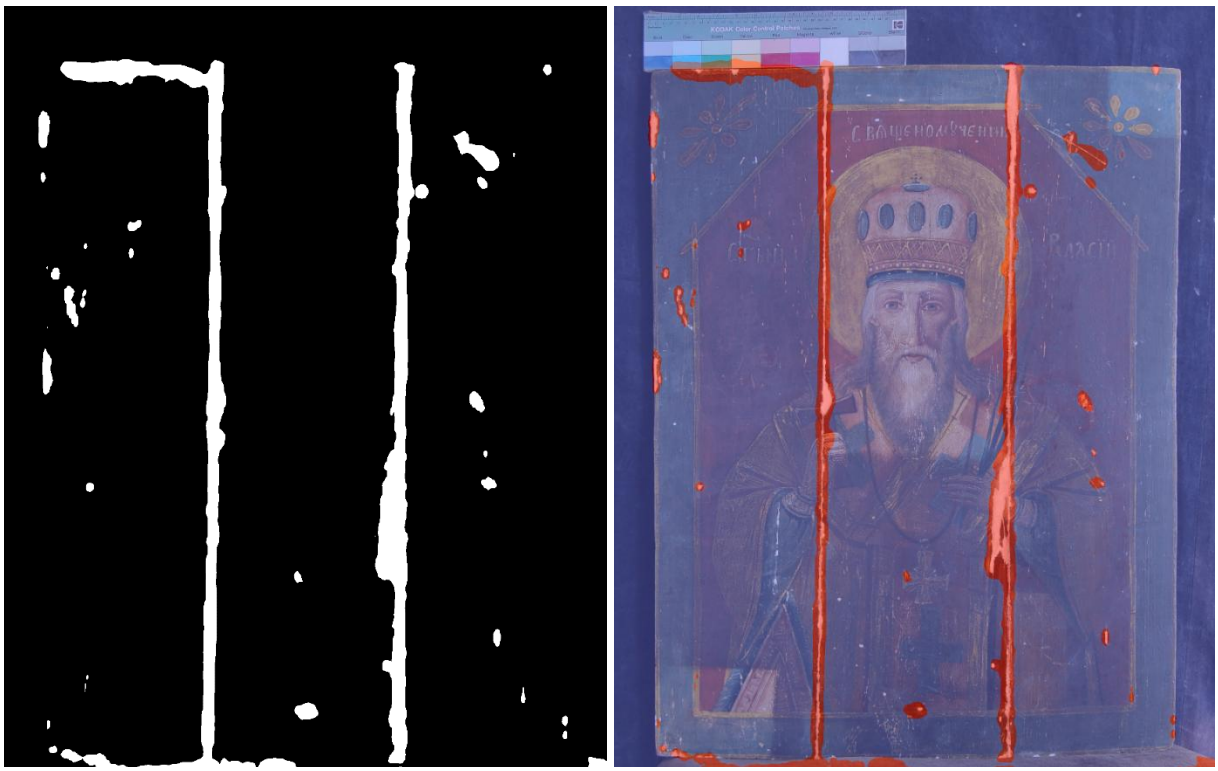
а) б)

Рисунок 3.8 а) – оригінальне зображення, б) – результат порогування з порогом 130, 8% втрат

На рисунку 3.8 б) зображений результат операції порогування на оригінальному зображенні. Як бачимо, результат порогування неточний, оскільки частина самої картини зарахувалася як втрата фарби, як і шкала кольору.

Це сталося через те, що інтенсивність значення пікселів в первних частинах картини висока, і при порогуванні їх значення є вищими заданого порогу.

На рисунку 3.9 а) зображено передбачення нейронної мережі, а на рисунку 3.9 б) передбачення накладене на оригінальне зображення. Як можна побачити, передбачення мережі є дещо точнішими результату порогування. Хоча різниця становить лише два відсотки, візуально можна побачити зовсім іншу картину.



а) б)

Рисунок 3.9 а) – передбачення нейронної мережі, 6% втрат, б) – передбачення нейронної мережі, накладене на оригінал

3.3 Порівняння результатів роботи алгоритму кластеризації K-середніх та нейромережі.

Через особливості алгоритму кластеризації, він є досить чутливим до шумів та вимагає щоб характеристики шуканого кластеру досить добре виділялися на фоні зображення. Таким чином можна зробити висновок, що якщо втрати будуть незначні, або не достатньо помітні для фону, алгоритм кластеризації включить в кластер з втратами частину зображення.

Для демонстрації цього твердження буде використане зображення на рисунку 3.10.

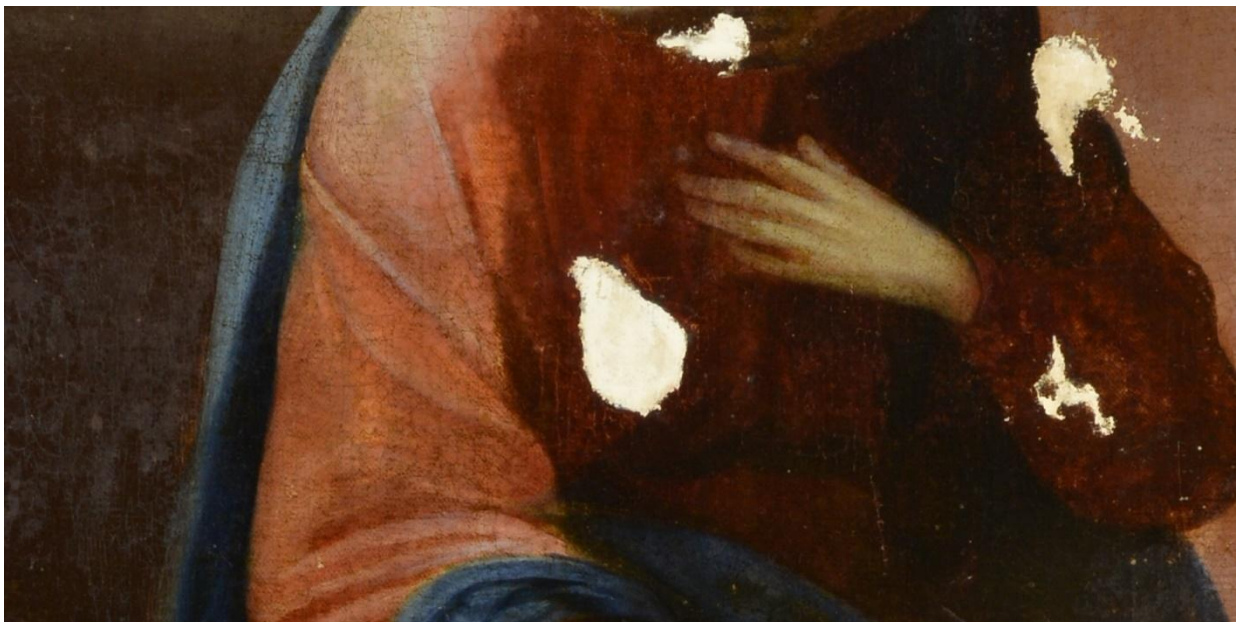


Рисунок 3.10 – Оригінальне зображення для порівняння кластеризації та нейромережі

На рисунку 3.11 а) зображений результат роботи алгоритму кластеризації, а на рисунку 3.11 б) – результат передбачення нейромережі. Як видно з рисунку 3.11 а), частина картини потрапила в кластер з втратами. Це сталося через те, що зображення досить велике, коли втрати замалі. Крім того фон зображення містить різні кольори, і темніша частина фону перейшла в кластер з зображенням а світліша у кластер із втратами. Таким чином можна побачити неефективність роботи алгоритму та його недоліки.

а) б)

Рисунок 3.11 а) – результат проходження алгоритму кластеризації, втрати – 29%, б) – передбачення нейромережі, втрати – 4%

При цьому різниця між алгоритмами – 29 та 4 відсотки. Тобто різниця в точності трішки більше ніж в сім (!) разів. Це означає, що алгоритм кластеризації

є негнучким та може застосовуватись лише до обмеженої кількості зображень, які є однорідними та містять значну частину втрат. Такими зображеннями можуть бути обрізані частини оригінального зображення, наприклад.

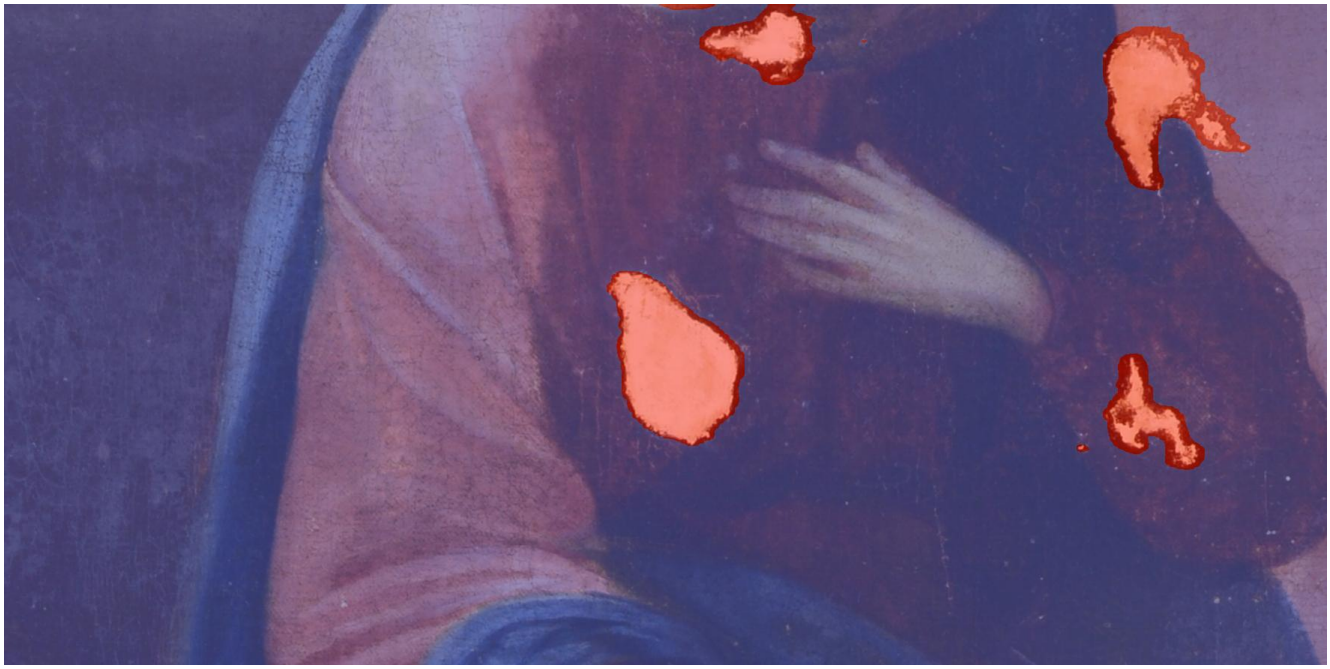


Рисунок 3.12 – Накладення передбачення нейромережі на оригінальне зображення

При цьому нейронна мережа показала себе просто чудово. Звичайно, області втрат не повністю відповідають дійсності, а є дещо ширшими за очікувані. Тим не менш, передбачення є досить точними, що і відображається на рисунку 3.12 – накладенні передбачення на оригінальне зображення. Перевага нейромережі полягає у адаптивності до різних зображень і можливості давати хороші передбачення незалежно від характеристик, до яких чутливий алгоритм кластеризації.

Порівняння методів з перевагами та недоліками кожного наведені в таблиці 3.1. Таким чином можна визначити переваги та недоліки різних методів сегментації при застосуванні для визначення втрат фарби на реставраційних картинах. Порогування – досить ефективний алгоритм але вимагає чіткого контрасту між об’єктом та фоном зображення. Алгоритм кластеризації є менш ефективним з точки зору продуктивності та обчислювальної складності, але може застосовуватись на попередньо оброблених зображеннях. Нейронна мережа є

найбільш ефективним алгоритмом машинного навчання, адаптивним до різних типів зображень, але вимагає великих ресурсів для навчання.

Таблиця 3.1 – Порівняння методів сегментації зображень

Назва методу	**Переваги**	**Недоліки**
Порогування	Прості обчислення Висока швидкість Коли об’єкт і фонове зображення мають високу контрастну роздільну здатність, метод працює добре	Неефективний, коли немає значної різниці між об’єктом та фоновим зображеннями відтінків сірого
Кластеризація K-середніх	Добре працює на невеликих наборах даних з однорідним фоном	Неефективний, якщо на зображенні занадто багато країв або коли фонове зображення занадто різноманітне
Нейронна мережа	Простий, гнучкий, загальний підхід Сучасний сучасний рівень сегментації зображень Висока адаптивність до різних типів зображень	Високий час тренувань Потрібен хороший різноманітний набір даних із мітками для навчання

Висновки до розділу 3

В цьому розділі була порівняна ефективність різних методів сегментації зображень для оцінювання втрат при реставрації зображень. Були виявлені переваги та недоліки різних методів, а також на реальних прикладах продемонстровані сильні та слабкі сторони алгоритмів.

Були визначені відсотки втрат на зображеннях для реставрації при різних картинах із досить різними рисами. Також в цьому розділі була визначена різниця між відсотком втрат при застосуванні алгоритмів на різних зображеннях з використанням слабких та сильних сторін алгоритмів.

4. РЕКОМЕНДАЦІЇ ТА ЇХ ЕКОНОМІЧНА ОЦІНКА

4.1 Загальні рекомендації та подальші напрямки досліджень

Як можна визначити із попередніх розділів, різні методи сегментації дають різну ефективність при їх застосуванні. Кожен з методів володіє своїми перевагами та недоліками.

Застосування алгоритмів машинного навчання та сегментації зображень в цілому є досить новим в процесі реставрації. При поточних темпах досліджень в цій області можна сказати, що досить скоро, в декілька найближчих десятиліть, алгоритми штучного інтелекту будуть в змозі оцінювати збитки для творів мистецтва, ціну реставрації, а може навіть і виконувати сам процес реставрації.

Поки що сам процес реставрації це клопітка ручна робота. В нашу чергу можна принести користь в процес реставрації застосовуючи алгоритми для автоматизації частини цього процесу. В цьому дослідженні був автоматизований процес обчислення та визначення втрат живопису.

Крім оцінки втрат, є ще цілий ряд етапів в реставрації які можуть бути автоматизовані в подальшому, хоча це може вимагати вузькоспеціалізованих фахівців та знань в області хімії.

Крім того, в цьому дослідженні були оцінені пошкодження лише живопису. Тим не менш, процес реставрації, як і історичні пам’ятки включає не тільки живопис, а й скульптуру, архітектуру та багато інших областей. При оцінюванні пошкоджень в скульптурі та будівлях існує своя специфіка, спрямована на якомога детальніший аналіз старіння саме матеріалів скульптури, чи будівництва.

Це можуть бути тріщини та пошкодження інших видів, які призводять до руйнування пам’яток. В наш час вчені вже робили подібні дослідження із застосуванням нейронних згорткових мереж в будівництві для оцінювання пошкоджень будівель [18, 19].

Із рекомендацій можна визначити деякі чинники які можуть полегшити застосування алгоритмів, подібних до тих які були застосовані в цій роботі в подальшому.

Слід зазначити, що алгоритми машинного навчання діляться на дві групи – навчання з учителем та без. Специфіка області застосування вимагає від алгоритму надання точних передбачень, що неможливо при роботі алгоритму без учителя, оскільки пошкодження є дуже різноманітними. Вони мають різну форму, походження, колір, розташування.

Таким чином такі алгоритми, як наприклад кластеризація добре працюють коли пошкодження займають значну частину зображення, і таким чином чітко можна відділити зображення від пошкоджень і розділити їх на окремі кластери.

Але то й же самий алгоритм виявляється неефективним коли мова іде про аналіз великих зображень з незначними втратами, або такими втратами які не чітко відображені на зображенні. Тут алгоритм кластеризації через специфіку роботи не може дати точних передбачень. В таких випадках потрібно застосовувати алгоритми машинного навчання з учителем.

Тобто потрібно створити датасет подібних зображень до тих, які потрібно проаналізувати, вручну виділити зони втрат та провести навчання моделі. В такому разі незалежно від кількості втрат, їх положення, форми та розташування, алгоритм завжди буде давати гарні передбачення.

Тим не менш, для повноцінного використання такої технології, необхідний датасет на 60 000 зображень із позначками втрат, що є надзвичайно дорогим та трудомістким процесом.

Крім того в цій роботі був розглянутий лише один вид втрат – втрати фарби, хоча існують і інші типи пошкоджень, наприклад від пересихання фарби – кракелюри. Ці та інші пошодження можуть бути розпізнані з додаванням нових класів для передбачення в модель, але вони будуть вимагати і нових зображень з мітками в датасеті.

4.2 Економічна оцінка рекомендацій

В наш час технології машинного навчання розвиваються дедалі більшими темпами, і кількість спеціалістів цього профілю невпинно росте з кожним днем.

Використання технологій які були наведені в цьому дослідженні є виправданим лише у випадку великої кількості зображень. Це обумовлено тим, що розробка відповідного ПЗ, а також утримання фахівців для його підтримування коштує чималих грошей.

Примітивні алгоритми типу порогування, або кластеризації не вимагають великих затрат коштів, і можуть бути використані багаторазово, хоча написати додаток потрібно лише один раз, як у випадку з нашим застосунком. Хоча використання таких алгоритмів є дешевим та легким, це має свої недоліки – точність роботи алгоритмів та результатів та необхідність адаптації зображень до алгоритмів.

Коли говориться про алгоритми машинного навчання типу нейронних мереж, ситуація стає зовсім іншою Тренування такої мережі займає значно більше часу та ресурсів. Також необхідно зробити датасет, написати саму мережу, натренувати її та інтегрувати із застосунком, щоб користувачі могли ним користуватися без будь-яких складнощів.

Крім того цю інфраструктуру потрібно підтримувати і забезпечувати нові та нові функції для користувачів, наприклад новий тип пошкоджень для оцінки, або на основі оцінки можна відновити пошкоджені області.

Таким чином така інфраструктура є дійсно необхідною лише за двох умов, або великої кількості зображень, або високої вартості картин, які будуть реставруватися і необхідності точної детальної машинної оцінки.

Висновки до розділу 4

В цьому розділі були проаналізовані подальші перспективи досліджень та застосування алгоритмів машинного навчання для оцінки втрат при реставрації творів мистецтва. Також була зроблена економічна оцінка рекомендацій, проаналізувана доцільність застосування тих чи інших методів, а також області та особливості можливого застосування.

ВИСНОВКИ

Розвиток сучасних інформаційних технологій призводить до збільшення областей для їх застосування. Таким чином з’являється все більше перспектива для автоматизації різних процесів з допомогою інформаційних технологій. Однією з таких областей є реставрація творів мистецтва. Метою цієї роботи було дослідження впровадження алгоритмів машинного навчання для оцінювання втрат при реставрації творів мистецтва.

У цій роботі було:

1. Проаналізовано предметну область та особливості процесу реставрації. Проведено аналіз етапів реставрації та виділено етап оцінки втрат зображення, автоматизацію якого і зроблено в подальшому. Було проаналізовано вже існуючі підходи до вирішення задачі та визначили подальші кроки для реалізації.
2. Реалізувано застосунок на мові Java з використанням сторонніх бібліотек для порогування та алгоритму кластеризації. Також був реалізований зручний інтерфейс в цьому додатку з можливістю динамічно змінювати порогове значення та додавати різні фільтри. Були визначені переваги та недоліки застосування фільтрів та описані принципи їх роботи.
3. Була розроблена згорткова нейронна мережа для сегментації зображень на мові Python з використанням сучасних бібліотек для конфігурації та роботи з мережею та зображеннями загалом. Створений датасет з наявних зображень для тренування мережі. Описана та застосована техніка аугментації, яка збільшила датасет та урізноманітнила екземпляри в ньому, що збільшило точність передбачень.
4. Алгоритми сегментації були застосовані на реальних зображеннях які в подальшому піддавались процесу реставрації. Були описані сильні та слабкі сторони алгоритмів. Для демонстрації описаних тверджень були представлені різні типи зображень, які використовують сильні та слабкі сторони алгоритмів.

5. Були проаналізовані отримані результати роботи алгоритмів. Були порівняні отримані відсотки втрат від роботи різних алгоритмів. Проведена оцінка доцільності застосування алгоритмів в різних умовах.
6. Були проаналізовані можливості напрямків подальших досліджень та надані рекомендації із застосування даних алгоритмів. Також була надана економічна оцінка впровадження технологій машинного навчання в процес реставрації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ravi Kaushik, Shailender Kumar Image Segmentation Using Convolutional Neural Network [Електронний ресурс]. – Режим доступу: http://www.ijstr.org/final-print/nov2019/Image-Segmentation-Using-Convolutional-Neural-Network.pdf

2. Shengyuan Li and Xuefeng Zhao Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique. [Електронний ресурс]. – Режим доступу: https://www.hindawi.com/journals/ace/2019/6520620

3. Olaf Ronneberger, Philipp Fischer, and Thomas Brox U-Net: Convolutional Networks for Biomedical Image Segmentation. [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/1505.04597.pdf

4. Hyunseok Seo, Masoud Badiei Khuzani, Varun Vasudevan, Charles Huang, Hongyi Ren, Ruoxiu Xiao, Xiao Jia, and Lei Xing Machine Learning Techniques for Biomedical Image Segmentation: An Overview of Technical Aspects and Introduction to State- of- Art Applications. [Електронний ресурс]. – Режим доступу: https://arxiv.org/ftp/arxiv/papers/1911/1911.02521.pdf

5. K. Bhargavi, S. Jyothi A Survey on Threshold Based Segmentation Technique in Image Processing [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/309209325_A_Survey_on_Threshold_Based_Segmentation_Technique_in_Image_Processing

6. Nameirakpam Dhanachandra, Khumanthem Manglem and Yambem Jina Chanu Image Segmentation using K-means Clustering Algorithm and Subtractive Clustering Algorithm. [Електронний ресурс]. – Режим доступу: https://cyberleninka.org/article/n/501681.pdf

7. Swanand Mhalagi The Quest of Higher Accuracy for CNN Models. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/the-quest-of-higher-accuracy-for-cnn-models-42df5d731faf

8. Georgy Gimel'farb Image Filtering and Segmentation. [Електронний ресурс]. – Режим доступу: https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/2013/CS373-IP-03.pdf

9. Raimi Karim, Illustrated: 10 CNN Architectures. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

10. Open Source Computer Vision. [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/master/d1/dfb/intro.html

11. OpenCV Java documentation (4.5.2-dev). [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/master/javadoc/index.html

12. Getting Started with JavaFX. [Електронний ресурс]. – Режим доступу: https://openjfx.io/openjfx-docs/

13. How to Load a Java Native/Dynamic Library (DLL). [Електронний ресурс]. – Режим доступу: https://www.chilkatsoft.com/java-loadLibrary-Windows.asp

14. Jason Brownlee A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. [Електронний ресурс]. – Режим доступу: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

15. Image segmentation with a U-Net-like architecture. [Електронний ресурс]. – Режим доступу: https://keras.io/examples/vision/oxford_pets_image_segmentation/

16. NumPy v1.20 Manual. [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/stable/

17. Alexandra Deis Data Augmentation for Deep Learning. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9

18. Jung Jin Kim, Ah-Ram Kim and Seong-Won Lee, Artificial Neural Network-Based Automated Crack Detection and Analysis for the Inspection of Concrete Structures. [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/2076-3417/10/22/8105/pdf

19. Arun Mohan, Sumathi Poobal Crack detection using image processing: A critical review and analysis. [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1110016817300236

ДОДАТОК А

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger
from util import path, data, misc, generator as gen
from dip import dip
import setting.constant as const
import importlib
import sys

class NeuralNetwork():
    def __init__(self):
        self.arch = importlib.import_module("%s.%s.%s" % (const.dn_NN,
const.dn_ARCH, const.MODEL))

        self.fn_logger = path.fn_logger()
        self.fn_checkpoint = path.fn_checkpoint()

        self.dn_image = path.dn_train(const.dn_IMAGE)
        self.dn_aug_image = path.dn_aug(const.dn_IMAGE, mkdir=False)

        self.dn_label = path.dn_train(const.dn_LABEL)
        self.dn_aug_label = path.dn_aug(const.dn_LABEL, mkdir=False)

        self.dn_test = path.dn_test()
        self.dn_test_out = path.dn_test(out_dir=True, mkdir=False)

        try:
            self.model = self.arch.model(self.has_checkpoint())
            if (self.has_checkpoint()):
                print("Loaded: %s\n" % self.fn_checkpoint)
        except Exception as e:
            sys.exit("\nError loading: %s\n%s\n" % (self.fn_checkpoint, str(e)))

    def has_checkpoint(self):
        return self.fn_checkpoint if path.exist(self.fn_checkpoint) else None

    def prepare_data(self, images, labels=None):
        if (labels is None):
            for (i, image) in enumerate(images):
                number = ("%0.3d" % (i+1))
                path_save = path.join(self.dn_test_out, mkdir=True)

                image, _ = dip.preprocessor(image, None)
                original_name = (const.fn_PREPROCESSING % (number))
                data.imwrite(path.join(path_save, original_name), image)
```

```
                yield self.arch.prepare_input(image)
            else:
                for (image, label) in zip(images, labels):
                    (image, label) = dip.preprocessor(image, label)
                    yield self.arch.prepare_input(image), self.arch.prepare_input(label)

        def save_predict(self, original, image):
            path_save = path.join(self.dn_test_out, mkdir=True)

            with open(path.join(path_save, (const.fn_SEGMENTATION)), 'w+') as f:
                for (i, image) in enumerate(image):
                    number = ("%0.3d" % (i+1))
                    image_name = (const.fn_PREDICT % (number))

                    image = dip.posprocessor(original[i], self.arch.prepare_output(image))
                    data.imwrite(path.join(path_save, image_name), image)

                    seg = (image == 255).sum()
                    f.write(("Image %s was approximately %f segmented (%s pixels)\n" %
(number, (seg/image.size), seg)))

                    original_name = (const.fn_ORIGINAL % (number))
                    data.imwrite(path.join(path_save, original_name), original[i])

                    overlay_name = (const.fn_OVERLAY % (number))
                    overlay = dip.overlay(original[i], image)
                    data.imwrite(path.join(path_save, overlay_name), overlay)
            f.close()

    def train():
        nn = NeuralNetwork()

        total = data.length_from_path(nn.dn_image, nn.dn_aug_image)
        q = misc.round_up(total, 100) - total

        if (q > 0):
            print("Dataset augmentation (%s increase) is necessary (only once)\n" % q)
            gen.augmentation(q)

        images, labels = data.fetch_from_paths([nn.dn_image, nn.dn_aug_image],
[nn.dn_label, nn.dn_aug_label])
        images, labels, v_images, v_labels = misc.random_split_dataset(images, labels,
const.p_VALIDATION)
```

```
        epochs, steps_per_epoch, validation_steps =
misc.epochs_and_steps(len(images), len(v_images))

        print("Train size:\t\t%s |\tSteps per epoch: \t%s\nValidation size:\t%s
|\tValidation steps:\t%s\n"
            % misc.str_center(len(images), steps_per_epoch, len(v_images),
validation_steps))

        patience, patience_early = const.PATIENCE, int(epochs*0.25)
        loop, past_monitor = 0, float('inf')

        checkpoint = ModelCheckpoint(nn.fn_checkpoint, monitor=const.MONITOR,
save_best_only=True, verbose=1)
        early_stopping = EarlyStopping(monitor=const.MONITOR,
min_delta=const.MIN_DELTA, patience=patience_early, restore_best_weights=True,
verbose=1)
        logger = CSVLogger(nn.fn_logger, append=True)

        while True:
            loop += 1
            h = nn.model.fit_generator(
                shuffle=True,
                generator=nn.prepare_data(images, labels),
                steps_per_epoch=steps_per_epoch,
                epochs=epochs,
                validation_steps=validation_steps,
                validation_data=nn.prepare_data(v_images, v_labels),
                use_multiprocessing=True,
                workers=0,
                callbacks=[checkpoint, early_stopping, logger])

            val_monitor = h.history[const.MONITOR]

            if ("loss" in const.MONITOR):
                val_monitor = min(val_monitor)
                improve = (past_monitor - val_monitor)
            else:
                val_monitor = max(val_monitor)
                improve = (val_monitor - past_monitor)

            print("\n###################")
            print("Finished epoch (%s) with %s: %f" % (loop, const.MONITOR,
val_monitor))

            if (abs(improve) == float("inf") or improve > const.MIN_DELTA):
```

```
            print("Improved from %f to %f" % (past_monitor, val_monitor))
            past_monitor = val_monitor
            patience = const.PATIENCE
            test(nn)
        elif (patience > 0):
            print("Did not improve from %f" % (past_monitor))
            print("Current patience: %s" % (patience))
            patience -= 1
        else:
            break
        print("####################\n")

def test(nn=None):
    if nn is None:
        nn = NeuralNetwork()

    if (nn.has_checkpoint()):
        images = data.fetch_from_path(nn.dn_test)
        generator = nn.prepare_data(images)

        results = nn.model.predict_generator(generator, len(images), verbose=1)
        nn.save_predict(images, results)
    else:
        print(">> Model not found (%s)\n" % nn.fn_checkpoint)
```

ДОДАТОК Б

```
from keras.models import Input
from keras.engine.training import Model
from keras.layers import Conv2D, MaxPooling2D, Dropout, UpSampling2D,
Concatenate
from keras.optimizers import Adam
import numpy as np

IMAGE_SIZE = (256,256,1)

def model(weights_input=None):

    inputs = Input(IMAGE_SIZE)
    conv1 = Conv2D(64, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(inputs)
    conv1 = Conv2D(64, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(128, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(pool1)
    conv2 = Conv2D(128, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(256, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(pool2)
    conv3 = Conv2D(256, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(512, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(pool3)
    conv4 = Conv2D(512, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(pool4)
    conv5 = Conv2D(1024, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv5)
    drop5 = Dropout(0.5)(conv5)
```

```
        up6 = Conv2D(512, 2, activation="relu", padding="same",
kernel_initializer="he_normal")(UpSampling2D(size = (2,2))(drop5))
        merge6 = Concatenate(axis=3)([drop4,up6])
        conv6 = Conv2D(512, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(merge6)
        conv6 = Conv2D(512, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv6)

        up7 = Conv2D(256, 2, activation="relu", padding="same",
kernel_initializer="he_normal")(UpSampling2D(size = (2,2))(conv6))
        merge7 = Concatenate(axis=3)([conv3,up7])
        conv7 = Conv2D(256, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(merge7)
        conv7 = Conv2D(256, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv7)

        up8 = Conv2D(128, 2, activation="relu", padding="same",
kernel_initializer="he_normal")(UpSampling2D(size = (2,2))(conv7))
        merge8 = Concatenate(axis=3)([conv2,up8])
        conv8 = Conv2D(128, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(merge8)
        conv8 = Conv2D(128, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv8)

        up9 = Conv2D(64, 2, activation="relu", padding="same",
kernel_initializer="he_normal")(UpSampling2D(size = (2,2))(conv8))
        merge9 = Concatenate(axis=3)([conv1,up9])
        conv9 = Conv2D(64, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(merge9)
        conv9 = Conv2D(64, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv9)
        conv9 = Conv2D(2, 3, activation="relu", padding="same",
kernel_initializer="he_normal")(conv9)

        conv10 = Conv2D(1, 1, activation="sigmoid")(conv9)

        model = Model(inputs=inputs, outputs=conv10)
        model.compile(optimizer=Adam(lr=1e-4), loss="binary_crossentropy",
metrics=["accuracy"])

        if weights_input:
            model.load_weights(weights_input)

        return model
```

```
def prepare_input(image):
    image = np.reshape(image, image.shape+(1,))
    image = np.reshape(image,(1,)+image.shape)
    image = np.clip(image, 0, 255)
    return np.divide(image, 255)

def prepare_output(image):
    image = image[:,:,0]
    image = np.clip(image, 0, 1)
    return np.multiply(image, 255)
```

ДОДАТОК В

```
package sample.functional;

import org.opencv.core.*;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class KMeans {
    private static long counter = 0;

    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    public static long perform(String path, String filename, boolean erosion, boolean
blurSelected) {

        counter = 0;
        System.out.println(path);
        Mat img = Imgcodecs.imread(path);
        Imgproc.cvtColor(img, img, Imgproc.COLOR_BGR2GRAY);

        if (erosion) {
            Imgproc.erode(img, img,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(3, 3)));
        }

        if (blurSelected) {
            Imgproc.GaussianBlur(img, img, new Size(5, 5), 0);
        }

        Mat clusters =
                cluster(img, 2).get(0);
        countLoss(clusters);
        String currentDirectory = System.getProperty("user.dir").replaceAll("\\\\",
"\\\\\\\\");
        counter = (long) (((double) counter) / (img.rows() * img.cols()) * 100);
        counter = Math.min(counter, 100 - counter);
```

```
        Imgcodecs.imwrite(currentDirectory + "\\results\\" + filename + "-kmeans-" +
counter + ".jpg", clusters);
        return counter;
    }

    private static void countLoss(Mat clusters) {

        for (int i = 0; i < clusters.rows(); i++) {

            for (int j = 0; j < clusters.cols(); j++) {
                if (clusters.get(i, j)[0] == 0) {
                    counter++;
                }
            }
        }
    }

    public static List<Mat> cluster(Mat cutout, int k) {
        Mat samples = cutout.reshape(1, cutout.cols() * cutout.rows());
        Mat samples32f = new Mat();
        samples.convertTo(samples32f, CvType.CV_32F, 1.0 / 255.0);

        Mat labels = new Mat();
        TermCriteria criteria = new TermCriteria(TermCriteria.COUNT, 100, 1);
        Mat centers = new Mat();
        Core.kmeans(samples32f, k, labels, criteria, 15, Core.KMEANS_PP_CENTERS,
centers);

        return showClusters(cutout, labels, centers);
    }

    private static List<Mat> showClusters(Mat cutout, Mat labels, Mat centers) {
        centers.convertTo(centers, CvType.CV_8UC1, 255.0);
        centers.reshape(2);

        List<Mat> clusters = new ArrayList<Mat>();
        for (int i = 0; i < centers.rows(); i++) {
            clusters.add(Mat.zeros(cutout.size(), cutout.type()));
        }

        Map<Integer, Integer> counts = new HashMap<>();
        for (int i = 0; i < centers.rows(); i++) {
            counts.put(i, 0);
        }
        int rows = 0;
```

```
        for (int y = 0; y < cutout.rows(); y++) {
            for (int x = 0; x < cutout.cols(); x++) {
                int label = (int) labels.get(rows, 0)[0];
                counts.put(label, counts.get(label) + 1);
                clusters.get(label).put(y, x, 255);
                rows++;
            }
        }
        System.out.println(counts);
        return clusters;
    }
}
```