ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Кафедра дискретного аналізу

Видобуток даних даних з використанням ШНМ

Індивідуальне завдання по темі: Саморганізаційні карти Кохонена

Виконала:

Студентка групи

ПМі-43

Шувар Софія

Викладач:

Колос Н.М..

2023

Постановка задачі: Розробити систему для кластеризації пікселів зображення на кластери з використанням алгоритму самоорганізаційних карт Кохонена. Кластеризація повинна бути реалізована таким чином, щоб кожен кластер містив пікселі зі схожими кольорами і візуально відображав сутність зображення. Порівняти результати кластеризації отримані власним алгоритмом та вбудованим.

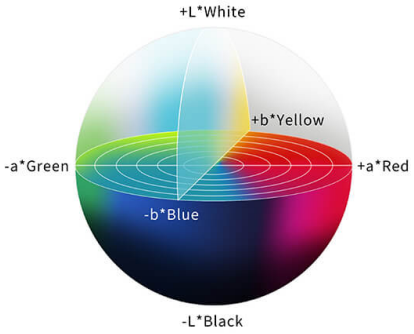
Теоретичні відомості:

Завдання кластеризації зображення на кластери по кольорах має декілька практичних застосувань:

- **Сегментація зображень**: Кластеризація зображення може бути використана для сегментації зображень, тобто розділення зображення на різні регіони або області залежно від кольорів. Наприклад, для автоматичного видалення фону з зображень або виділення об'єктів на зображенні.
- **Стиснення зображень**: Кластеризація зображення може бути використана для стиснення зображень, тобто зменшення обсягу зображення без втрати суттєвих деталей. Замість зберігання кольору кожного пікселя окремо, можна зберігати кластери центрів мас.
- **Обробка зображень**: Кластеризація зображення може бути використана для покращення якості зображень, наприклад, для видалення шуму або зменшення кількості кольорів на зображенні.
- **Рекомендації для створення дизайну**: Кластеризація зображення на кластери по кольорах може бути використана для аналізу кольорових схем і підказки дизайнерам, які кольори найкраще підходять для створення гармонійного дизайну.

Першим кроком у вирішенні завдання кластеризації є підготовка даних для навчання таким чином, щоб по них можна було точно обчислити подібність між прикладами (у даній задачі - пікселі). Тому для групування пікселів за кольором доцільним є перевести оригінальне зображення з формату RGB у формат LAB. Колірний простір LAB (Lab*) - це тривимірний колірний простір, який розділяє інформацію про колір на три канали: яскравість (L), червоно-зелений (a) і синьо-жовтий (b) (мал. 1). Під час кластеризації кольорів у просторі кольорів LAB евклідова відстань між кольорами в

просторі LAB більше відповідає перцептивним відмінностям між кольорами, ніж евклідова відстань в інших просторах кольорів. Це означає, що схожі за сприйняттям кольори будуть згруповані разом, навіть якщо вони не близькі один до одного в колірному просторі RGB або HSL.



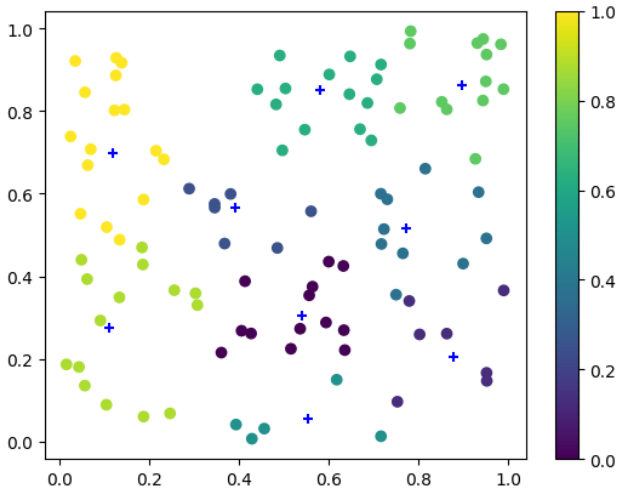
(мал. 1 Представлення кольорів у LAB форматі)

Хід роботи

1. Програмна реалізація виконана мовою програмування Python, у середовищі Google Colab. Для роботи використовувались бібліотеки Numpy - для роботи з багатовимірними масивами, MatPlotlib - для візуалізації отриманих результатів, OpenCV - для роботи з зображеннями. Для тестування вбудованого алгоритму використовувався клас MniSom із бібліотеки minisom.

2. **Тестування бібліотечної функції:**
   Для тестування бібліотечного класу MniSom спочатку генерую 100 рандомнох точок на площині. Потім задала розмір сітки SOM як 3x3 і створила об'єкт MiniSom з відповідними параметрами: input_len = 2 - розмірність вхідних даних, sigma = 0.3 - радіус округлення для пошуку найближчих вузлів в процесі навчання SOM (чим більше значення цього параметра, тим менш впливає на навчання точки, які віддалені від найближчих вузлів SOM), learning_rate = 0.5 - швидкість навчання SOM (визначає, наскільки далеко від сусідніх вузлів SOM можна перемістити ваги в процесі навчання). Початкові ваги ініцілізую рандомним чином.
   Для кожної точки даних обчислюється найближчий вузол SOM, асоційований з нею, і зберігається як кластерна приналежність точки.

Кластерна приналежність відображається на графіку, де кожна точка має колір, вказаний за допомогою мапи кольорів "viridis", що відповідає кластерам, вузли SOM, які представляють центри кожного кластера, позначені знаком "+".

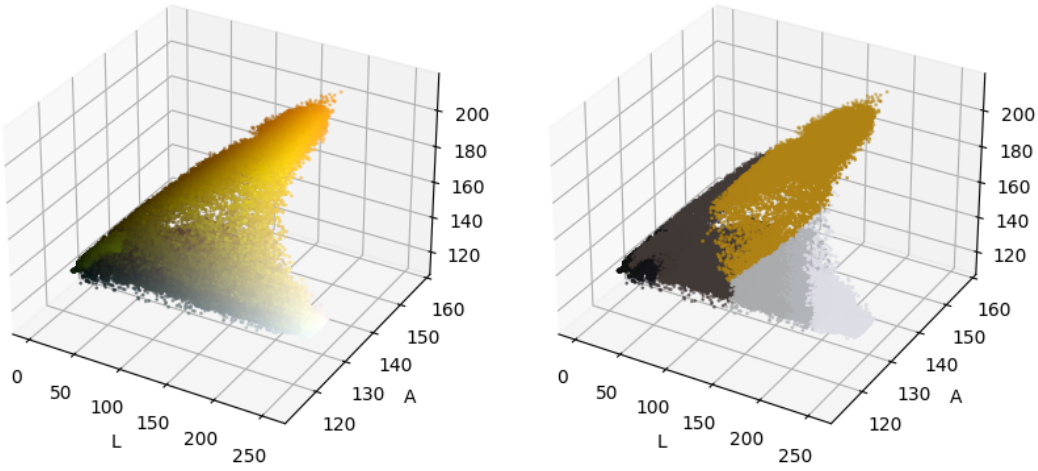


(мал. 2 Тестування класу MiniSom з бібліотеки minisom)

3. **Застосування функції:**

За допомогою бібліотеки OpenCV зчитала кольорове зображення у форматі BGR та перевела його у формат Lab. Отриманий три вимірний масив перевела у двовимірний масив точок розмірності nx3, де n - добуток ширини та висоти зображення. Створила об'єкт MiniSom вказавши розмір сітки 3x3 та розмірність даних як 3. Решта параметрів залишаються як у попередньому пункті. Треную SOM на масиві точок lab, потім для кожної точки даних присвоюється номер кластеру, що визначається розташуванням нейрона переможця у lab просторі. Перетворюю масив lab точок назад у зображення у форматі RGB. Таким чином отримали кластеризоване зображення (мал. 3), також демонструю графік який ілюструє як кластеризувались пікселі зображення (по осях відкладені кожен з каналів lab відповідно) (мал. 4).

(мал. 3 Зображення до та після кластеризації)

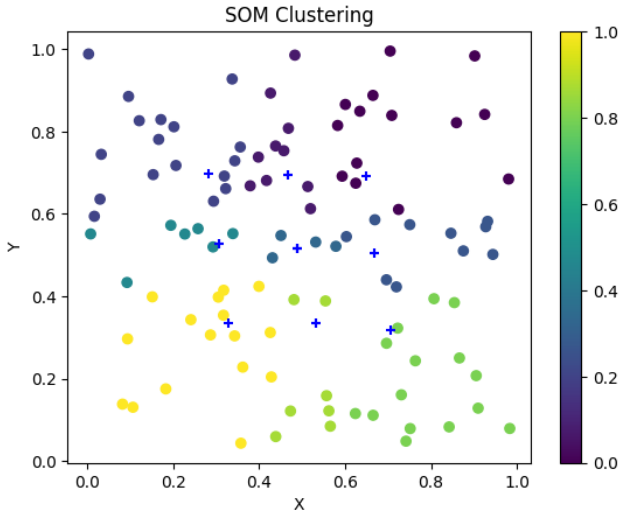


(Мал 4. Графічне представлення зміни кольору кожного пікселя після кластеризації)

4. **Програмна реалізація самоорганізаційних карт Кохонена**.
   Клас SOM містить такі методи:
   - **init()**: ініціалізує SOM зі заданими параметрами: розмірність n_input, розмір сітки n_output та швидкість навчання learning_rate. Ваги ініціалізуються випадковими значеннями.
   - **train()**: навчає SOM на даних протягом num_epochs епох. Для кожного зразка даних виконується пошук кращої відповідності, оновлення вагів та зниження швидкості навчання.
   - **find_best_matching_unit():** знаходить найближчий до вхідного зразка нейрон в вихідному шарі.

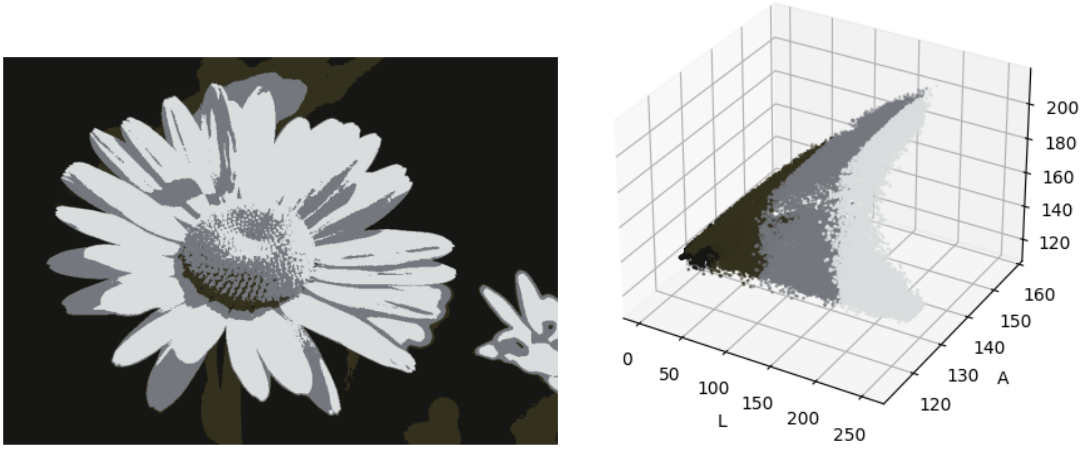
- **update_weights()**: оновлює ваги для кожного нейрону відповідно до близькості між ним і вибраним найближчим нейроном.
- **predict()**: прогнозує клас для вхідних даних на основі ближчого нейрона.

Тестую алгоритм аналогічно як у пункті 2 (мал. 5).



(мал. 5 Тестування власного класу)

Застосовую алгоритм до кластеризації зображення (мал 6). Треную мережу лише на 10 епохах, так як тренування на власному алгоритмі є досить часозатратне.



(мал. 6 Зображення та Графічне представлення зміни кольору кожного пікселя після кластеризації)