Majestic

Majestic team presents

ANN project

Image compression

Table of contents

01

Project Description

02

project mechanism

03

code

04

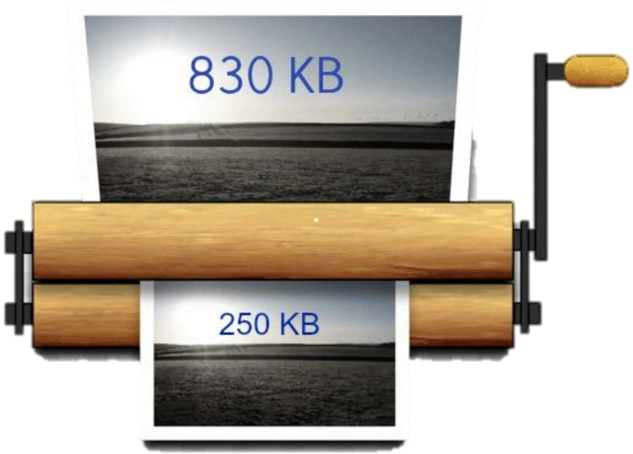
Project application area

01

Project Description

Project Description

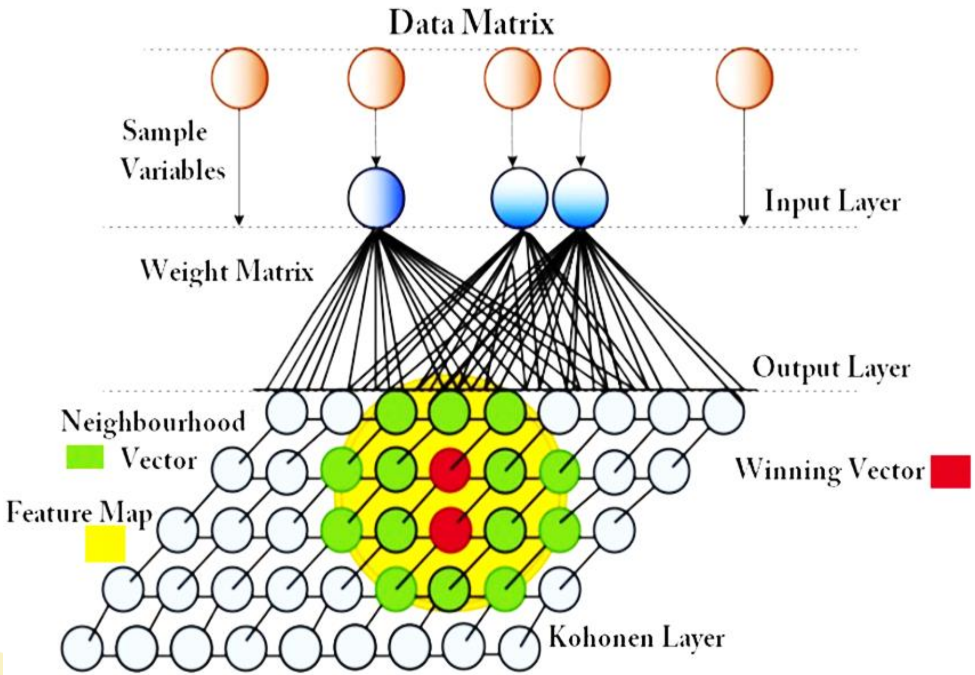
Project name	Image Compression
The Artificial Neural Network algorithm used “Compression algorithm”	SOM algorithm
Learning type	Unsupervised learning Grid clustering
Input	Digital image
Output	The compressed version of the input image.
Compression type	Lossy Compression
The purpose of project	To reduce the size of the image while maintaining an acceptable level of visual quality for storage or transmission.



What does image compression mean?

Image compression is a type of data compression applied to digital images without degrading the quality of the image to an unacceptable level.

What is SOM algorithm?



"SOM" stands for "Self-Organizing Map." It's a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional representation of the input space of the training samples, called a map.

Why do we use SOM algorithms to compress images?

Spatial Preservation

SOM algorithms maintain spatial relationships in images, ensuring visually coherent compression.

Flexibility

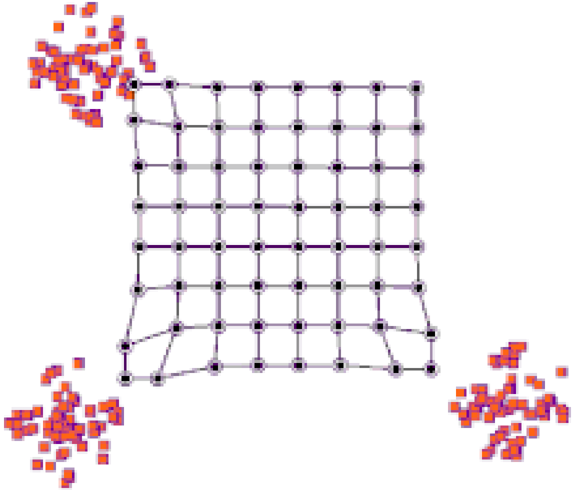
SOMs can adapt to different data distributions and structures, making them versatile for various types of images and compression requirements.

Good performance

It achieves high compression ratios and acceptable visual quality, particularly for images with large uniform areas. It's useful for compressing maps, diagrams, and charts.

02

project mechanism



The steps of the SOM algorithm for image compression

Step 1: Create a grid of neurons, each with randomly initialized weights.

Step 2: Training Phase: Iterate through each pixel in the input image.

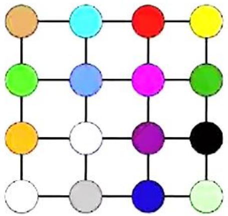
- **a.** Find the neuron with the weights most similar to the current pixel vector "BMU".
- **b.** Update the weights of the winning neuron and its neighbors to better match the input pixel.
- **c.** Repeat this process for a set number of iterations.

Step 3: Quantization Phase:

- **a.** Map each pixel of the original image to the neuron in the SOM grid with the most similar weights.
- **b.** Replace each pixel's value with the value of the corresponding winning neuron's weights.

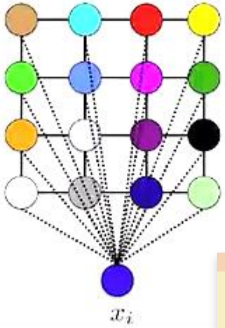
Step 4: The compressed image is generated, where each pixel has been replaced by the value of the winning neuron in the SOM grid.

Explain SOM in figures



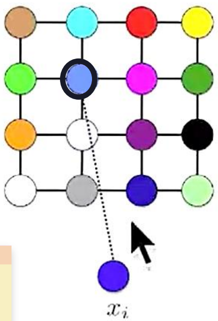
Initialize 4x4 Grid

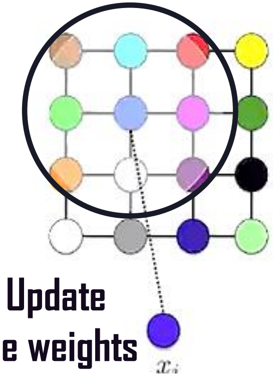
Step 1



Find BMU

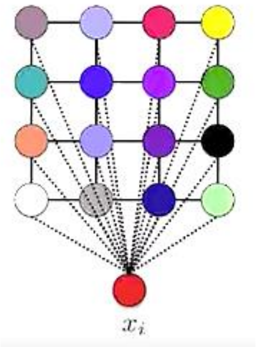
Step 2.a

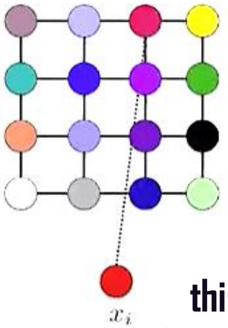




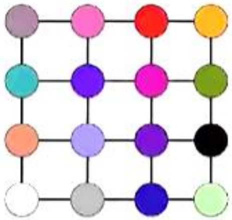
Update the weights

Step 2.b



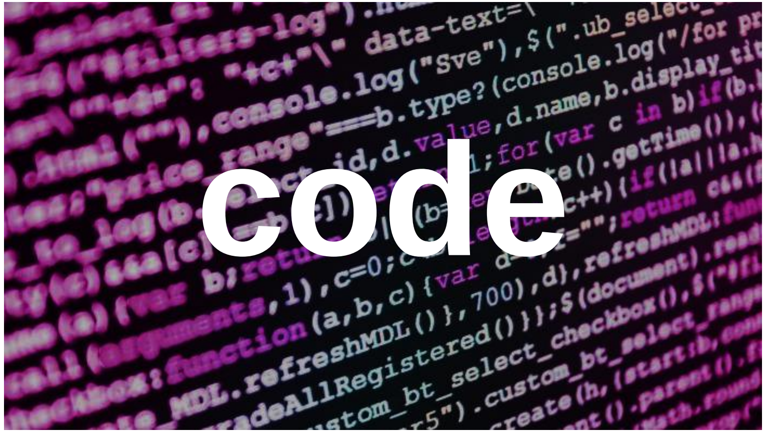


Repeat this process



Step 2.c

03



code

The libraries imported in the code

1

NumPy used for numerical computations, array manipulation, and mathematical operations.

2

PyQt is used to create graphical user interfaces (GUIs) and desktop applications with a native look.

3

Pillow (PIL) the Python Imaging Library (PIL), is used for opening, manipulating, and saving various image file formats.

4

matplotlib.pyplot is used for creating visualizations, including displaying images and plotting the performance_measurement.

5

OS Module

os provides a way of using operating system-dependent functionality and used to get the size of the images files.

6

time provides utilities for time-related tasks, including measuring time, delaying program execution, and formatting time values.

The component functions of the code

train

Trains SOM using data for iterations, updating weights based on BMU and neighbors.

find_bmu

Finds BMU for a given data point by calculating Euclidean distances.

update_weights

Updates weights based on BMU, learning rate, and neighborhood radius.

compress_image

compresses an image using a trained SOM. It loads the image, trains the SOM, and assigns each pixel to the closest neuron's weight, saving the result.

compare_images

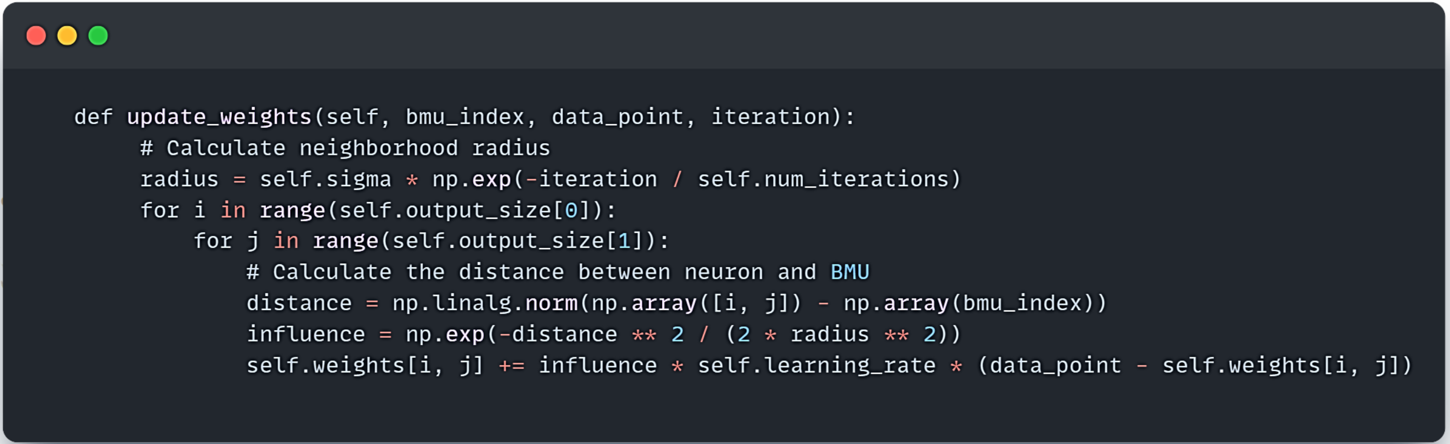
Compares original/compressed images side by side.

performance_measurement

Measures compression ratio and quantization error, displays results.

update_weights fuction

1. **Calculate Neighborhood Radius**:
    1. The neighborhood radius determines the extent to which neighboring neurons will be updated. It decreases over time (iterations).
    2. It is calculated based on the sigma value (initially specified) and decreases exponentially over the course of the iterations.
2. **Update Weights of BMU and Neighbors**:
    1. For each neuron in the SOM grid:
        1. Calculate the Euclidean distance between the current neuron and the Best Matching Unit (BMU).
        2. The influence of the neuron on the BMU is calculated using a Gaussian function. **wj = e^(-dj^2/2r^2)**
        3. The weights of the neuron are updated based on the influence, learning rate, and the difference between the input data point and the current weights of the neuron. **Nj (new) = Nj (old) + η wj (xi - Nj)**



```python
def update_weights(self, bmu_index, data_point, iteration):
    # Calculate neighborhood radius
    radius = self.sigma * np.exp(-iteration / self.num_iterations)
    for i in range(self.output_size[0]):
        for j in range(self.output_size[1]):
            # Calculate the distance between neuron and BMU
            distance = np.linalg.norm(np.array([i, j]) - np.array(bmu_index))
            influence = np.exp(-distance ** 2 / (2 * radius ** 2))
            self.weights[i, j] += influence * self.learning_rate * (data_point - self.weights[i, j])
```

Performance measurement

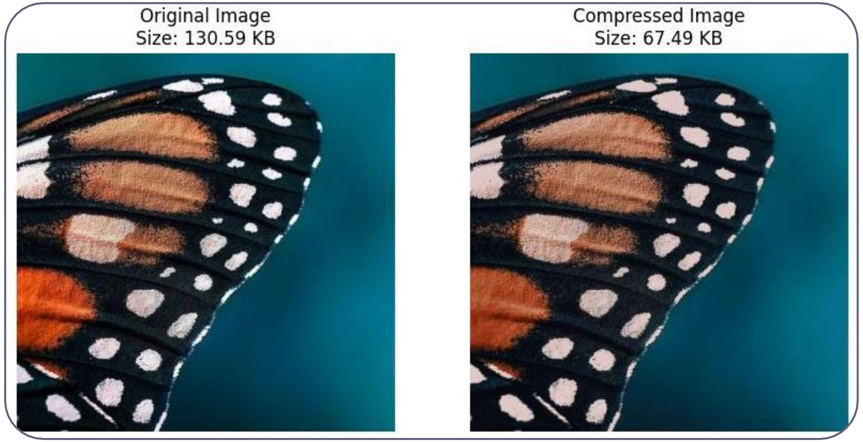
1. Compression Ratio (CR) Calculation:

- **Definition**: Compression Ratio (CR) measures the degree of compression achieved by an algorithm. It is the ratio of the size of the original data to the size of the compressed data.

- **Formula**: CR=Original Size / Compressed Size

where:

- **Original SizeOriginal Size:** Size of the original dataset.
- **Compressed SizeCompressed Size:** Size of the compressed dataset after applying the algorithm.



Performance measurement

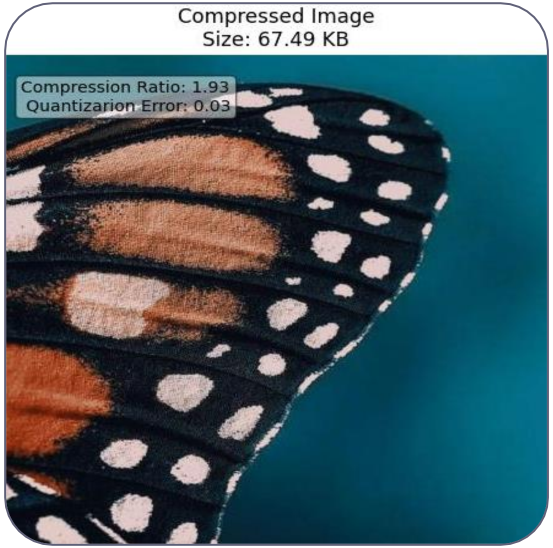
2. Quantization Error (QE) Calculation:

- **Definition**: Quantization Error (QE) measures the average difference between the original data points and their corresponding best matching units (BMUs) in the trained Self-Organizing Map (SOM).

- **Formula**: QE= ∑i=1-> N || Xi – BMU_Weight$i$ || / N

where:

- N : Total number of data points.
- Xi : ith data point in the dataset.
- BMU_Weight$i$: the weight of Best Matching Unit (BMU) for the i th data point in the trained SOM.
- ||·|| : Euclidean distance function.



04

Project application area

Image compression using SOM
has many applications in real life scenarios .

where large amounts of image data need to be stored or transmitted over limited bandwidth networks.

Here are some examples of its applications:

- **Digital cameras:** Compressing images to save storage space.
- **Online image sharing:** Compressing images to make them easier and faster to upload and download.
- **Medical imaging:** Compresses medical images like X-rays and CT scans to reduce storage space and network bandwidth required.
- **Satellite imaging:** Compressing images to reduce the amount of data that needs to be transmitted back to Earth.
- **Video streaming:** Compressing individual frames of videos to reduce file size and improve streaming performance.
- **Autonomous vehicles:** Compressing visual data to reduce the amount of data that needs to be processed in real time.

Majestic

Prepared By :

Ahmed Diaa ELShahat

Raafat Mohamed El-menayar

Wesam Ahmed El-Sharawy

Ibrahim Mohamed El-Morsi

Ahmed Abdel Raouf Abdel Shafi

Under the supervision of :

Dr.Mona Elbedwehy