Image segmentation Report

Sinuo Liu

November 22, 2024

1 For-loop-based mean-shift

In the initial implementation, a for-loop-based Mean-Shift algorithm was used. The core idea of the algorithm is to iteratively move each point to the weighted average of its neighboring points, gradually aggregating the data points to find the high-density regions of the data.
The distance function is used to compute the Euclidean distance between a given point and all other points. A Gaussian kernel is used to assign weights based on the distances. The closer the points, the higher the weight:

$$K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2h^2}\right) \tag{1}$$

The position of each point is updated by taking the weighted average of its neighbors:

$$\mu_i^{(t+1)} = \frac{\sum_{j=1}^{N} K\left(\frac{\mu_i^{(t)}-\mathbf{x}_j}{h}\right)\mathbf{x}_j}{\sum_{j=1}^{N} K\left(\frac{\mu_i^{(t)}-\mathbf{x}_j}{h}\right)} \tag{2}$$

In each iteration, the algorithm updates the points by moving them towards their local weighted mean. The result is:

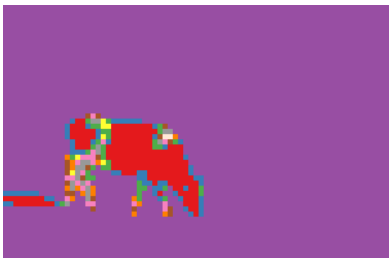

Figure 1: Segmented Image Result

2 Vectorized mean-shift

To improve the efficiency of the Mean-Shift algorithm, a vectorized implementation was used. Instead of iterating through each point individually, matrix operations are applied to process all points simultaneously.
In the vectorized version, the pairwise distances between all points are calculated at once using the `torch.cdist` function. Once the distance matrix is computed, the Gaussian kernel weights for all points are computed at once using broadcasting. The weighted sums of all points are computed using matrix multiplication (`torch.matmul`), and the total weights for each point are summed. The position updates are then performed using matrix operations, which efficiently handle the computation for all points. The result is the same as the for-loop-based algorithm.

3 Time Comparison

The execution time for the for-loop-based version was measured at 22.72 seconds, and for the vectorized version was significantly faster, taking only 4.08 seconds. The running time for the for-loop-based version is significantly higher compared to the vectorized version. This is expected, as the vectorized approach minimizes the overhead of looping over individual data points. When processing large images or high-dimensional data, the performance boost from vectorization becomes even more pronounced. By utilizing efficient matrix operations and reducing redundant computations, it becomes much faster, making it more suitable for large-scale image segmentation tasks.