Comparative Analysis of K-means and Spectral Clustering for Color-based Image Segmentation

Sai Surya Teja Bhogaraju
u7143732

Michele Meziu
u7622160

Abstract

A central task in computer vision is to perform image segmentation in an unsupervised way. In this project, we implemented a customized version of K-means and spectral clustering to perform color-based image segmentation on horse, deer and airplane images from the CIFAR-10 dataset. With thorough experimentation, our code accurately segmented images in most of the scenarios and matched the performance of the reference implementation. Notably, our implementation also prioritized computational efficiency.

1 Introduction

In this work, we aim to address the need for unsupervised learning methods in image segmentation, considering the limitations of supervised approaches that rely on large amounts of labeled training data. Image segmentation plays a crucial role in various domains such as object recognition, tracking, medical imaging, and autonomous driving. While supervised methods, particularly those based on neural networks like U-nets [6], have achieved impressive results, their reliance on extensive training data can be cost-prohibitive. On the other hand, unsupervised methods such as K-Means clustering [4], SLIC [1], and spectral clustering [7], [5] are less accurate compared to deep learning-based models. Therefore, our research focuses on investigating unsupervised learning methods to overcome this deficiency. Specifically, we analyze and discuss the performance of two prominent algorithms, namely K-Means and spectral clustering, on the widely-used CIFAR-10 dataset [3].

The structure of our work is as follows: In Section 2, we provide a clear definition of the problem we aim to tackle. Section 3 presents a detailed explanation of our implementations of K-means and spectral clustering algorithms, including the mathematical details of our experiment metrics. The segmentations produced by our algorithms are presented and compared in Section 4. Finally, in Section 5, we draw conclusions based on the experimental results, summarizing our findings and discussing the implications of the performance of the evaluated algorithms. By conducting this research, we aim to contribute to the understanding of unsupervised learning methods for image segmentation and provide insights into their suitability and effectiveness for practical applications.

2 Problem Definition

The problem can be formulated as follows:

- Implement an unsupervised model using the K-means++ algorithm to accurately segment the input image X of size $n \times n$ pixels.
- Implement an efficient graph-based image segmentation method using spectral clustering.
- Qualitatively assess the segmentation results of both methods and compare them against each other and with a reference implementation on a variety of images.
- Perform a quantitative analysis using metrics like IoU to determine how similar the images are, and silhouette score to determine the quality of clustering.

3 Methodology

3.1 K-means clustering for image segmentation.

K-means clustering is one of the most popular unsupervised machine-learning algorithms used for performing image segmentation. A cluster is defined as a neighborhood of points (pixels in our case) that are similar to each other according to some measure (color in our case). In other words, image segmentation is performed by splitting the whole image into K clusters where the pixels corresponding to each cluster are similar in color.

Let X be an image of size $n \times n$ with a total of N pixels that we want to partition in K clusters. Each of the N pixels is an RGB vector. The main aim of the K-means clustering algorithm is to minimize the objective function:

$$F = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|(x_i - \mu_k)^2\| \tag{1}$$

where μ_k is the mean of all the points belonging to cluster k and the parameter r_{ik} is called responsibility. The value of r_{ik} is 1 if the point x_i belongs to the cluster k, and 0 otherwise. The pseudo-code below explains the steps that are taken to perform K means image segmentation. We introduce the parameter $iterations$ to determine the number of iterations required for the execution of the K-means algorithm. The time complexity of the algorithm is $\mathcal{O}(n^2 * K * iterations)$ and the space complexity is $\mathcal{O}(n^2)$.

Algorithm 1 K-means Image Segmentation

procedure K-MEANSSEGMENTATION(X, $K, iterations$)
  Initialize k centroids $\mu_1, \mu_2, \ldots, \mu_K$ from the image X
  while $iteration < iterations$ **do**
    for $pixel$ in X **do**
      $nearest\_centroid \leftarrow \arg\min_\mu \|pixel - \mu\|^2$ ▷ Find nearest centroid using Euclidean distance
      Add $pixel$ to $clusters[nearest\_centroid]$
    for $centroid$ in $centroids$ **do**
      $new\_centroid \leftarrow \frac{1}{|\text{clusters[centroid]}|} \sum_{pixel \in \text{clusters[centroid]}} pixel$
    $iteration \leftarrow iteration + 1$
  Assign each pixel in image to the nearest centroid
  return Segmented image

In our implementation, we initialize centroids using K-means++. We choose the first centroid randomly among our data points, i.e. the pixels. Successively, we choose centroids randomly one after the other: the probability of a point being picked as a centroid is proportional to its distance to the closest centroid. This procedure makes sure that centroids are well distributed since points that are far from the current centroids are likely to be picked as the next centroids.

3.2 Spectral clustering for image segmentation

Spectral clustering is a clustering algorithm that takes a graph-based view of the data. A weighted graph is a tuple $G = (V, E, w)$, where V is called the *vertex* set, $E \subseteq V \times V$ is called the *edge* set, and $w : E \mapsto \mathbb{R}$ is a function mapping each edge to its weight. We indicate $w((u, v))$ with w_{uv} as a shorthand. We will consider only undirected graphs, that is, (u, v) and (v, e) are the same edge, and therefore $w_{uv} = w_{vu}$.

In weighted graphs, we can define *cuts*, which are partitions of V into two sets to which we can assign a cost. The cost of a cut $(A, V \backslash A)$ is given by the formula below.

$$c(A, V \backslash A) = \sum_{\substack{(u,v) \\ u \in A, v \in V \backslash A}} w_{uv} \tag{2}$$

A more useful concept in the context of clustering is the one of a *normalized cut*. Unlike cuts, it takes into account the size and the number of edges of the two partitions between which it is computed. To define normalized cuts we need to introduce another concept. The volume of a subset of nodes $A \subseteq V$ is defined as

$$v(A) = \sum_{u \in A} \sum_{v \in V} w_{uv}, \tag{3}$$

where we consider the weight of edges that are not present in the graph to be zero. Volumes can be used to add a weight factor to cut costs to penalize cuts that isolate small groups of nodes, giving normalized cuts

$$n(A, V \backslash A) = c(A, V \backslash A) \cdot \left(\frac{1}{v(A)} + \frac{1}{v(V \backslash A)} \right). \tag{4}$$

One can generalize this score to a K-partition of V just by summing all the pairwise normalized-cut costs.

Spectral clustering constructs a *neighborhood similarity graph* from the data, which is nothing but a weighted graph on the data points, and uses tools from spectral graph theory to find an optimal K-way normalized cut of the graph, returning a cluster for each connected component in the graph.

There are two main approaches for computing the K-way normalized cut in the data graph: one can compute 2-way cuts recursively as in [2], or one can find them simultaneously as in [7]. We compute the cuts simultaneously since this is a more efficient and accurate method, in addition to being more established in current literature.

The following pseudocode describes the application of the algorithm to image segmentation on a high level.

Algorithm 2 Spectral Clustering Image Segmentation

1: **procedure** SPECTRALSEGMENTATION($\mathbf{X}, K$)
2: Compute weight matrix W
3: $D \leftarrow \text{diag}\left(\sum_j W_{ij}\right)$ ▷ Compute the degree matrix
4: $L \leftarrow D^{-1/2}(D - W)D^{-1/2}$ ▷ Compute normalized Laplacian matrix
5: Find the top K eigenvectors $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{n \times K}$ of L and normalize the rows such that $\sum_i U(i) = 1$
6: Perform K-means clustering on each row of the top K eigenvectors to obtain the clusters $C_1, C_2, \ldots, C_k$
7: **return** Segmented image

We use two different versions of W in our code. The first implementation (which we call **Shi-Malik's spectral clustering**, i.e. column (d) of every figure) is using the weight matrix from Shi-Malik [7], which is given by

$$w(i,j) = e^{\left\| \frac{F(i)-F(j)}{\sigma_I} \right\|_2} \cdot \begin{cases} e^{\left\| \frac{X(i)-X(j)}{\sigma_X} \right\|_2} & \text{if } \|X[i] - X[j]\|_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $F[i] = [v, v \cdot s \cdot \sin(h), v \cdot h \cdot \cos(h)][i]$ and h, s, v are the HSV values of pixel i. In our implementation, we used RGB values instead, since it gave better segmentations. $X[i]$ and $X[j]$ represent the spacial locations of pixels i and j in the image. The weight matrix entry between two pixels i and j is calculated using the above equation 5 if the distance between those two pixels is less than a parameter r. The values we use in our code for σ_I, σ_X and r are the ones proposed in [7] for Fig. (9a), namely $\sigma_I = 0.01, \sigma_X = 4$ and $r = 5$.

Our second implementation (which we call **Ng spectral clustering**, i.e. column (c) of every figure) uses the weight matrix from [5], which is the RBF Gram matrix computed over the RGB values of the pixels.

In [5] they make use of a different matrix to extract the eigenvectors for segmentation, using W instead of $D - W$ in step 4 of Algorithm 2. However, we stick with $D - W$ as in the Shi-Malik paper.

We came up with these two method implementations after performing extensive trial and error with different combinations of weight and Laplacian matrices on the dataset. These approaches are chosen to optimize segmentation results and improve the quality of the outcomes.

Regarding the complexity of Algorithm 2, we observe that we are working on the $n \times n$ matrix L, where n is the number of pixels in the image. As far as time complexity goes, the bottleneck of the computation is the eigenvector extraction in line 5, which runs in $\mathcal{O}(n^3)$. Regarding space complexity, we notice that we are storing a constant number of $n \times n$ matrices, giving a $\mathcal{O}(n^2)$ complexity.

3.3 Metrics

We used the silhouette score and Intersection over Union (IoU) metrics to determine the quality of segmentations. The silhouette score is a metric to determine how close a pixel is associated with its cluster. IoU, on the other hand, is used to determine the overlap between two clusters. The metrics are expressed mathematically as follows:

$$\text{silhouette Score} : s(i) \begin{cases} \frac{b(k)-a(k)}{\max\{b(k),a(k)\}}, & |\text{cluster}[i]| > 1 \\ 0, & |\text{cluster}[i]| = 1 \end{cases} \tag{6}$$

where $a(i)$ and $b(i)$ are the mean inter- and intra-cluster distances of point i. If the size of the cluster where i belongs is one, then we assign it a silhouette score of zero. The silhouette score for a segmented image is just the average silhouette score of the pixels, with distances computed with RGB vecors.

$$\text{Intersection over Union} : IoU(seg1, seg2) = \frac{|seg1 \cap seg2|}{|seg1 \cup seg2|} \tag{7}$$

where seg1 and seg2 are corresponding segments obtained from two different segmentation algorithms. In our experiments we compute the mean IoU to compare two segmentations: we sort the segments of each of the two segmentations by size; we match the largest segment of one segmentation to the largest segment of the other segmentation, and so on; we compute the average IoU for the corresponding segments. This is based on the assumption that, if the background is dominant in the image, it will be the largest segment in both segmentations. Sometimes matching segments correspond to different parts of the image, making the mean IoU score occasionally unreliable.

4 Experiments

We performed our experiments on the airplane, deer, and horse class images from the CIFAR-10 dataset [3]. Each image from the dataset is of size 32x32, and it has 3 color channels. We run our code on macOS Monterey on a 2021 iMac with an AppleM1 chip and 16GB of RAM. On average, our K-means runs in 1.2s, Ng spectral clustering runs in 1.7s, and Shi-Malik spectral clustering runs in 2.7s. In fig. 1, fig. 2, and fig. 3 we present the results of the segmentation algorithms on airplane, deer, and horse images respectively. Specifically, for each image, we show the results for $K = 4$ of our K-means++ image segmentation, Ng spectral clustering, Shi-Malik spectral clustering, and the reference sklearn spectral clustering implementation. These figures are used to visually compare the results obtained from different approaches against the reference implementation.

For image (1b) in fig. 1 we observe that our implementation of K-means++ is outperforming the other algorithms, since it is preserving the left edge of the plane, and the shape and color of the blue stripe on the front. This is because K-means++ segments the image only based on the colors. Our spectral clustering approach (1c) yields a suboptimal segmentation on the left side of the plane since it is merging the airplane with the background. We speculate that this is due to the number of clusters not matching the eigenvalue gap. In the image (1d), the application of spectral clustering with the Shi-Malik weight matrix fails to provide a meaningful segmentation. This outcome is due to the weight matrix considering both color differences and distances between neighboring pixels, leading to the creation of patches with the same color in specific regions of the image.

Images (1b) and (1c) have an IoU score of $I_{bc} = 0.267$, which indicates that the segments computed for both images agree on a large part of the pixels. The IoU scores $I_{be} = 0.065$ and $I_{ce} = 0.004$ are close to zero. This could be due to a mismatch of the segments in the IoU computation since these

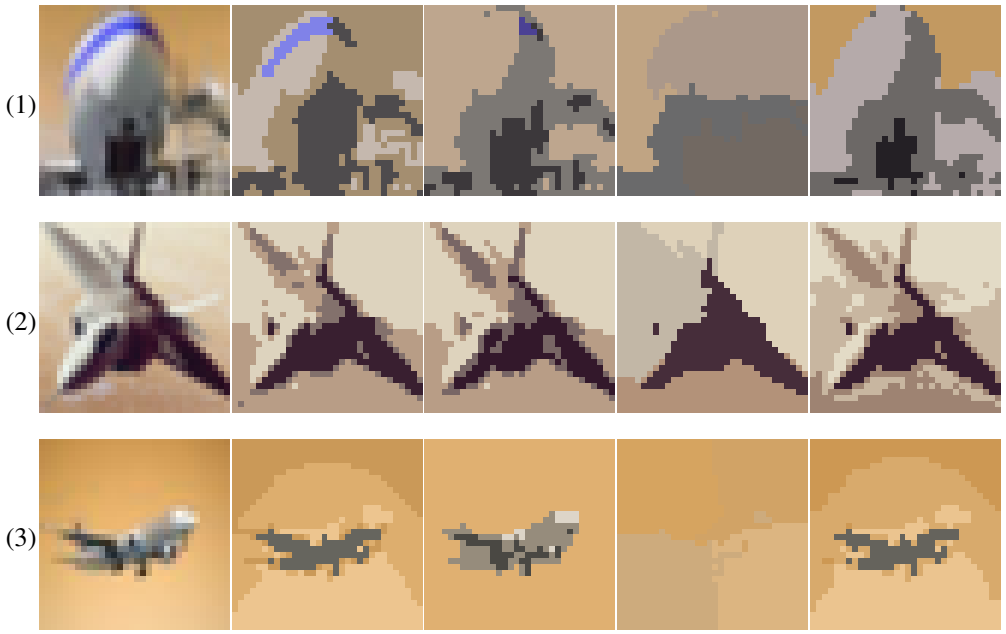

Figure 1: **Airplane Segmentation results:** a) Original image. b) Our K-means++. c) Ng spectral clustering. d) Shi-Malik spectral clustering. e) Sklearn Spectral Clustering.

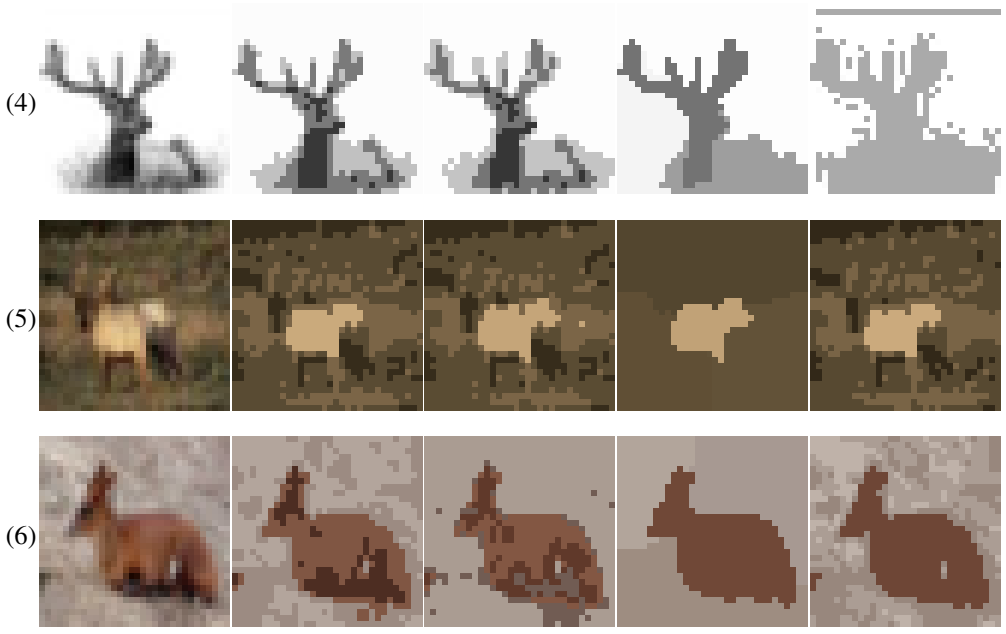

Figure 2: **Deer Segmentation results:** a) Original image. b) Our K-means++. c) Ng spectral clustering. d) Shi-Malik spectral clustering. e) Sklearn spectral clustering.

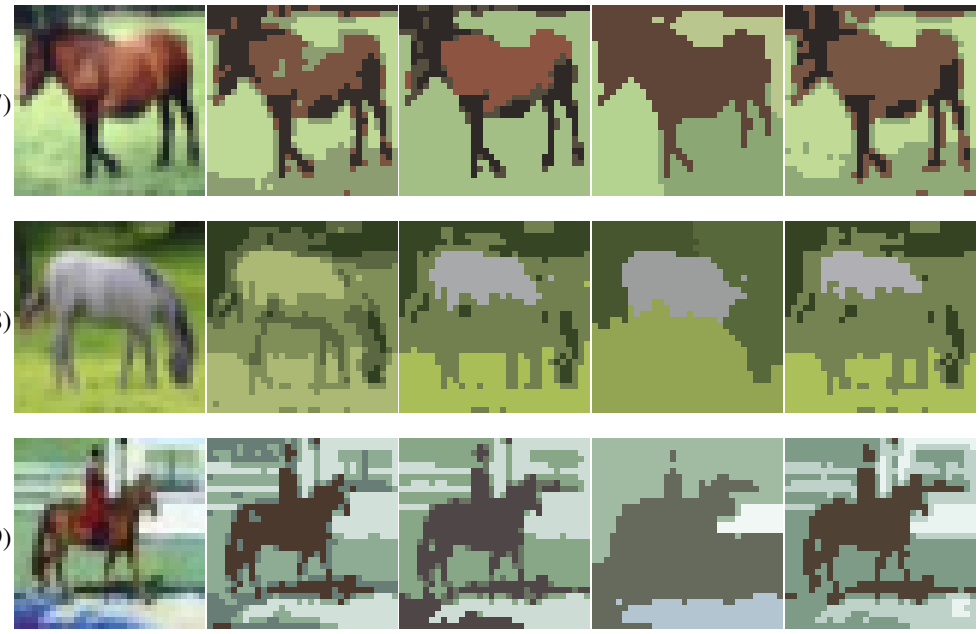

Figure 3: **Horse Segmentation results:**, a) Original image. b) Our K-means++. c) Ng spectral clustering. d) Shi-Malik spectral clustering. e) Sklearn spectral clustering.

segmentations visually do not look that different. The silhouette scores for (1b), (1c), and (1e) are $s_b = 0.409$, $s_c = 0.388$. $s_e = 0.414$. We notice that the scores for (1b) and (1e) are close, although the segmentation given by K-means++ looks more similar to the original image.

For image (2) in fig. 1, we notice that none of the algorithms is able to separate the shape of the plane from the background. We think that this is due to the fact that the left side of the plane shares the same color as the background. This situation highlights the challenge of segmenting objects that have a color resemblance to the background.

Regarding image (3) in fig. 1, we see that only spectral clustering with the Shi-Malik weight matrix (3d) does not separate the plane from its background. Even though the other methods are able to do that, only Ng spectral clustering (3c) assigns the background to one single cluster and distinguishes between the different parts of the plane like wings and fuselage. In (3b) and (3e), on the other hand, the background is split into three different clusters despite being of uniform color and texture. It is evident that in this example our custom implementation outperforms the other methods that we presented.

To test our algorithm on a grayscale image, we considered image (4) from the deer class in CIFAR-10. We notice that all five algorithms are able to roughly separate the deer from the background. The sklearn implementation of spectral clustering (4e), however, assigns parts of the background and some isolated pixels to the same cluster as the foreground. Moreover, it assigns pixels to only two clusters, although the provided parameter that defines the number of clusters is four. It is possible that this is due to a lack of convergence of K-means in the sklearn implementation. This supposition is corroborated by the fact that in the sklearn source code the maximum number of iterations `max_iter` is set to 300, whereas we always perform 10000 iterations whenever we use K-means in our work. Experimenting with the `max_iter` and the tolerance parameters in the sklearn source code is left to future research.

The IoU scores we have computed agree with our qualitative observations. $I_{bc} = 0.456$ is in fact larger than $I_{ce} = 0.326$ and $I_{be} = 0.292$, which means that images (4b) and (4c) are more similar to each other than they are to (4e). The silhouette scores for (4b), (4c), and (4e) are respectively $s_b = 0.784$, $s_c = 0.759$, and $s_e = 0.5$, which confirms that in (4b) and (4c) colors are better clustered than in (4e).

Analyzing image (5) of fig. 2 we can clearly see that all the algorithms are failing to extract the shape of the deer from the background. It is because there is no clear distinction between the foreground and background pixels as the entire deer image is majorly covered with a single pixel color. However, if we look at image (6) of fig. 2, there is a clear distinction between the foreground deer object and the background which makes all the algorithms' performance better. For instance, even though Shi-Malik spectral clustering segments the image into only three clusters, we can clearly recover the shape of the object.

In the image (8) of fig. 3, we observe that, due to lighting conditions, the back of the horse is brighter than the rest of its body. This causes almost all of the segmentation algorithms to cluster the region of the horse's back separately. We deduce that color-based segmentation algorithms are sensitive to lighting conditions. It is interesting to note that our K-means++ implementation assigns the back of the horse to the same segment as the grass, although K-means++ chooses centroids as far apart as possible, and therefore regions that are of very different colors initially should be mapped to different clusters.

5 Conclusion

In this project, we successfully implemented K-means++ and two versions of spectral clustering with different weight matrices. We evaluated the segmentation results of these algorithms on CIFAR-10 [3] images and compared them with the reference sklearn spectral clustering. Additionally, we conducted comparisons using IoU and silhouette scores.

For images where the foreground closely resembles the background or under challenging lighting conditions, K-means++ struggles to accurately separate the main object from the background, despite its overall satisfactory performance.

In terms of our custom spectral clustering implementation, we found that it generally performs well across different images, except for those with high noise levels like (5a) or poor lighting conditions like (8a). Specifically, spectral clustering with Shi-Malik's weight matrix yields better segmentation results when there is a distinct separation between the foreground and background, as evident in images (6d) and (7d).

Overall, our algorithms consistently match or even surpass the performance of the reference implementation in most cases. Furthermore, our algorithms demonstrate computational efficiency.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[3] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[4] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[5] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015.

[7] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.