K-means Clustering Algorithm for Color Image

Myoung-suk Kang

Abstract

In this project, we investigate the use of K-Means clustering for image compression and color palette extraction. We show how the algorithm can effectively decrease the number of unique colors in an image while preserving its visual appearance. Furthermore, we introduce a techni que for obtaining and displaying the color palette of the compressed image, which could be b eneficial in diverse areas such as design and computer vision.
As a student learning about K-Means clustering, you can apply this algorithm to explore ima ge compression and color palette extraction. The algorithm works by reducing the number of unique colors in an image, making it easier to store and process. By extracting and visualizin g the color palette, you can gain insights into the dominant colors in the image and apply th is knowledge in various creative and practical applications.

I. INTRODUCTION

Image compression is a crucial task in digital image processing, as it helps reduce the storage space required for images and allows for efficient transmission of images over networks. One of the techniques for image compression is color quantization, which involves reducing the number of unique colors in an image. In this paper, we use K-Means clustering to perform color quantization and extract a color palette from the compressed image. The color palette can be utilized for applications in design, computer vision, and other related fields.
K-Means clustering is an unsupervised learning algorithm that partitions a dataset into K clusters based on similarity. It aims to minimize the within-cluster sum of squares (WCSS), which is the sum of the squared distances between each data point and its corresponding cluster center. Mathematically, the objective function of K-Means clustering can be defined as:

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

where J(C) is the objective function, K is the number of clusters, C_k is the k-th cluster, x_i is a data point belonging to cluster C_k, μ_k is the centroid (mean) of cluster C_k, and $||x_i - \mu_k||^2$ is the squared Euclidean distance between x_i and μ_k.

The K-Means algorithm consists of the following steps:

1. Initialize K cluster centers randomly.
2. Assign each data point to the nearest cluster center.
3. Update the cluster centers by calculating the mean of all data points belonging to a cluster.
4. Repeat steps 2 and 3 until convergence or a specified number of iterations is reached.

The convergence of the algorithm is determined by checking whether the change in the objective function J(C) is smaller than a predefined threshold, or whether a maximum number of iterations has been reached.
K-Means has been widely used for various applications in image processing, such as image segmentation (Kanungo et al., 2002), feature extraction (Leung and Malik, 2001), and texture analysis (Haralick et al., 1973). In the context of image compression, K-Means has been employed for color quantization (Orchard and Bouman, 1991; Wu et al., 1992), which aims to reduce the number of unique colors in an image while preserving its visual quality.

II. METHOD

In this study, we employ the K-Means clustering algorithm, an unsupervised learning techniqu e, for partitioning data points into K distinct groups based on their similarity. The algorithm aims to minimize the sum of squared distances (in Euclidean space) between data points and their corresponding cluster centroids. The K-Means clustering algorithm can be broken down i nto the following steps:

1. **Initialization**: Randomly select K initial centroid positions.
2. **Assignment**: Assign each data point to the nearest centroid.
3. **Update**: Recalculate the centroid positions as the mean of all the data points belongin g to each centroid.
4. **Convergence**: Repeat steps 2 and 3 until the centroid positions do not change signific antly or a specified number of iterations is completed.

0.Load the two color image files: Airplane.jpg, Tiger.jpg

1. Image Preprocessing

The image preprocessing steps are crucial for preparing the image data for the K-Means clustering algorithm. The preprocessing includes:

- Loading the image using the **plt.imread()** function from the **matplotlib.pyplot** library. This function reads the image file and returns an array of shape (height, width, 3), where each element represents a pixel's RGB values.
- Normalizing the pixel values to be within the range [0, 1] by dividing them by 255. This step ensures that the algorithm operates on a standardized scale, which can improve its convergence and performance.

2. Image Reshaping and Clustering

To apply the K-Means clustering algorithm to the image data, the image is reshaped into a 2 D array of shape (width * height, 3), where each row represents the RGB values of a single pixel. We utilize the **KMeans** class from the **sklearn.cluster** module to fit the K-Means mod el on the reshaped image data. In this study, we chose 3, 5, 10, 20, 50 as the number of cl usters and set a random state for reproducibility. The random state ensures that the initial cen troid positions are consistent across multiple runs.

3. Calculating Cluster Frequencies

After fitting the K-Means model, the frequencies of each cluster label are calculated. This inf ormation is essential for determining the relative importance of each color within the image. The **np.unique()** function from the NumPy library is used with the **return_counts** parameter set to **True** to obtain unique cluster labels and their respective counts.

4. Visualization of Cluster Distribution and Colors

A custom bar plot is created using the **matplotlib.pyplot** library to visualize the distribution of cluster labels and their respective colors. The cluster centroids are used as colors for the b ars, providing a visual representation of the dominant colors in the image. The x-axis represe nts the cluster labels, and the y-axis indicates the frequency of each label.

5. Image Compression

The image compression process involves recreating the image using the K-Means cluster centr oids. This process effectively reduces the number of unique colors in the image while preserv ing its visual quality. The following steps are performed:

- For each pixel in the image, the K-Means label is used to assign the corresponding cl uster center color to that pixel.
- The image is reshaped back to its original dimensions (height, width, 3) to create the compressed image.

6. Color Palette Extraction

The color palette extraction process involves creating a visually appealing representation of th e dominant colors in the image. These colors are derived from the K-Means cluster centroids. The process includes:

- Initializing an empty list for storing the dominant colors.
- Creating a figure with a specified size using the **plt.figure()** function from the **matplo tlib.pyplot** library.
- Iterating through the K-Means cluster centroids and performing the following steps for each centroid:
    - Appending the centroid (a 3-element array representing the RGB values of the color) to the list of dominant colors.
    - Creating a color swatch (a small rectangular patch) filled with the current cent roid's color. This swatch is created by initializing an array of shape (100, 100, 3) with zeros, and then setting all elements in the array to the centroid's colo r.
    - Displaying the color swatch using the **plt.imshow()** function.
    - Calculating the percentage of pixels assigned to the current cluster label and d isplaying this percentage on the color swatch using the **plt.text()** function.

- Displaying the RGB values of the centroid below the color swatch using the **plt.xlabel()** function.

By following the above methods, we demonstrate the effectiveness of the K-Means clustering algorithm in image compression, color palette extraction, and visual analysis. The K-Means al gorithm is widely applicable for various image processing tasks, such as image segmentation, feature extraction, and texture analysis.

III. RESULTS AND DISCUSSION

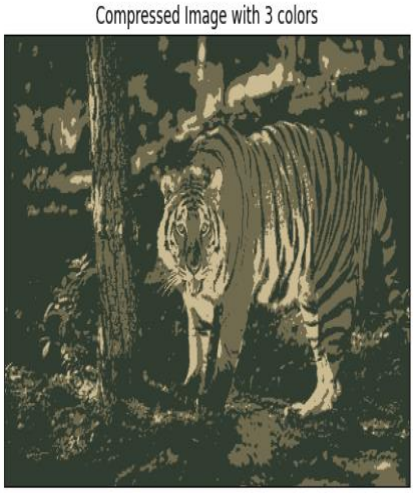
A. Result

In this study, we applied the K-Means clustering algorithm to achieve image compression and color palette extraction. The results are presented below:

1. Image Compression

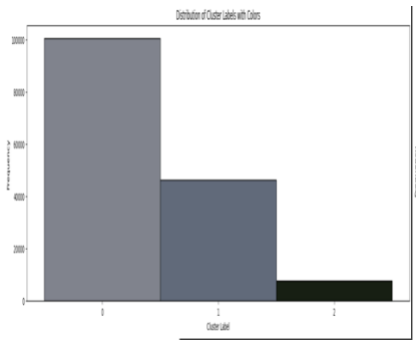
Airplane

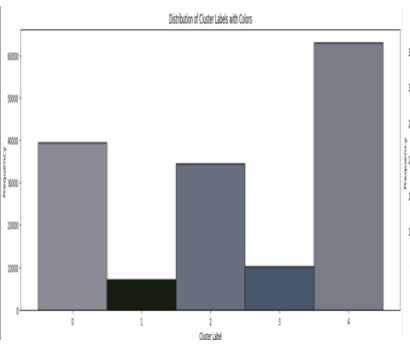
Tiger

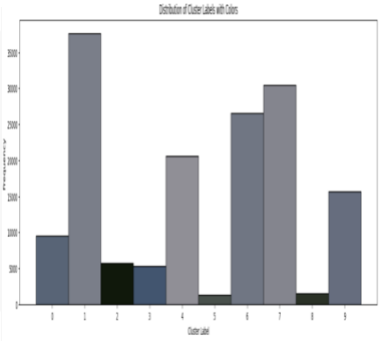


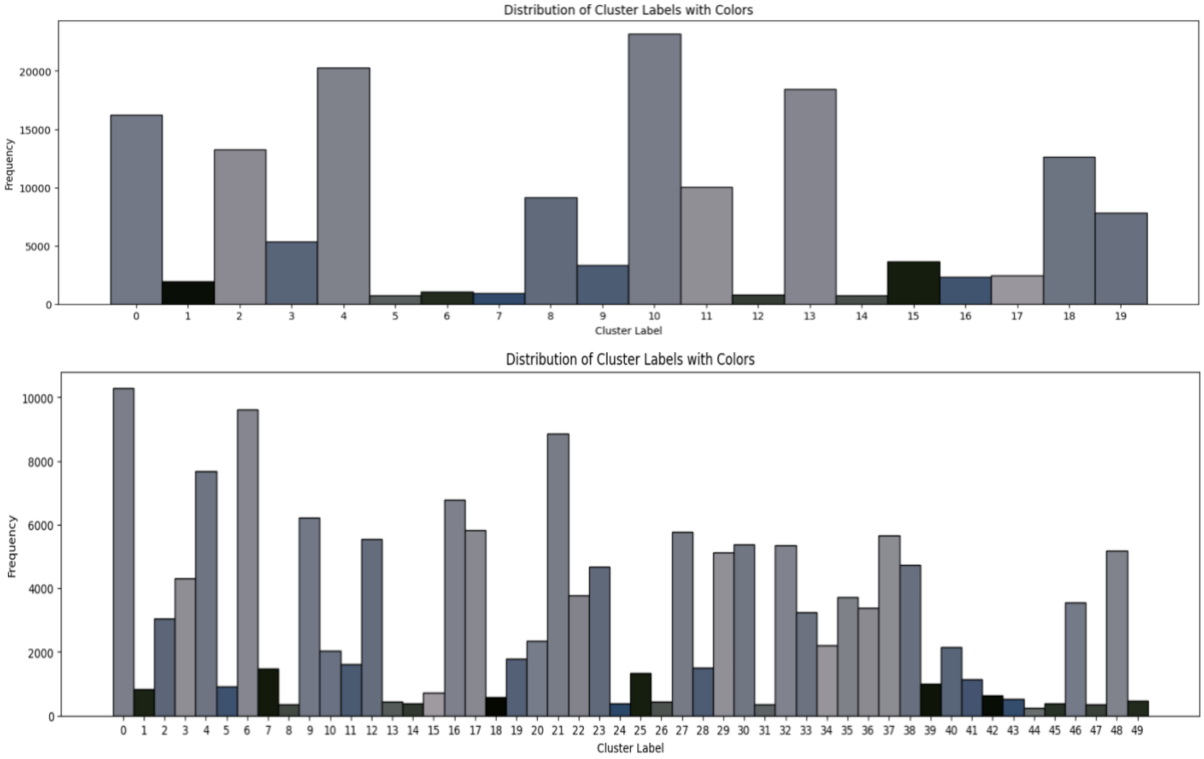
2. Visualization of Cluster Distribution and Colors

Airplane

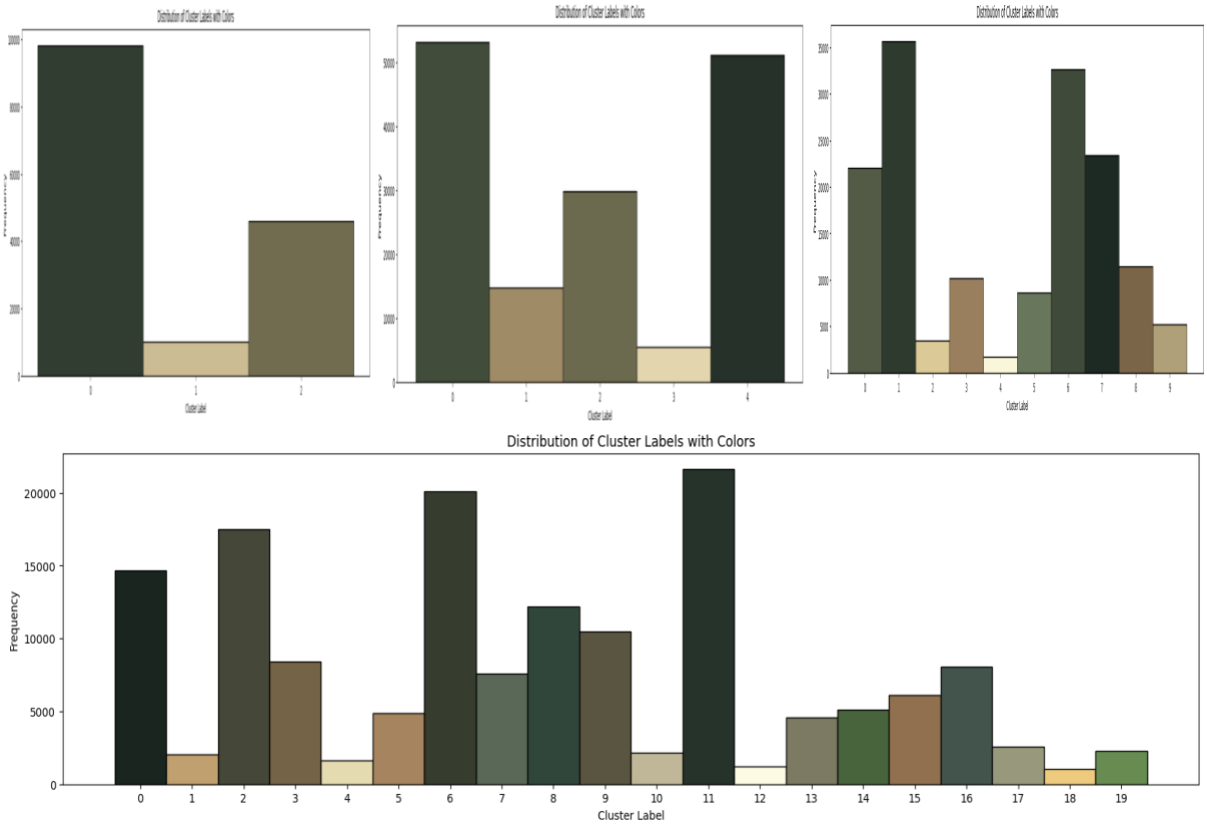


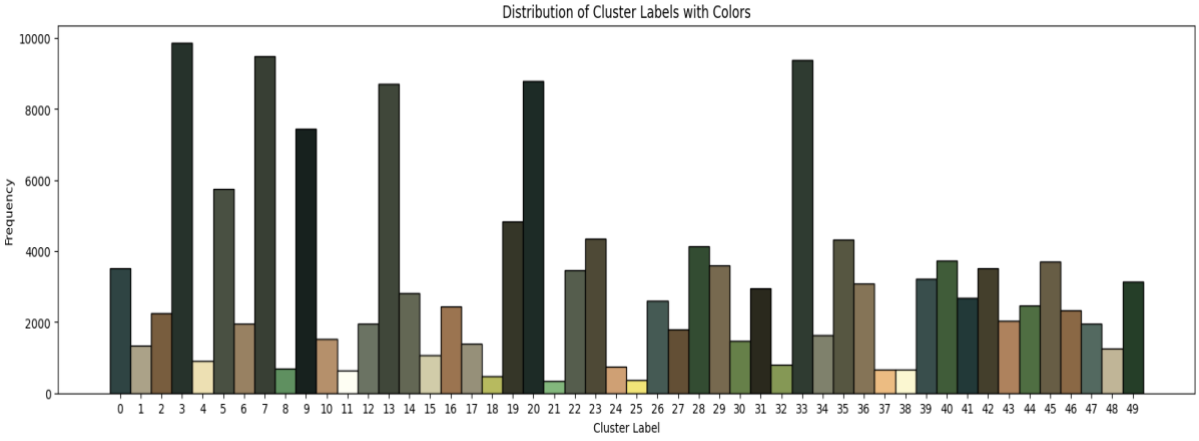





Distribution of Cluster Labels with Colors
Frequency
Cluster Label
Distribution of Cluster Labels with Colors
Frequency
Cluster Label

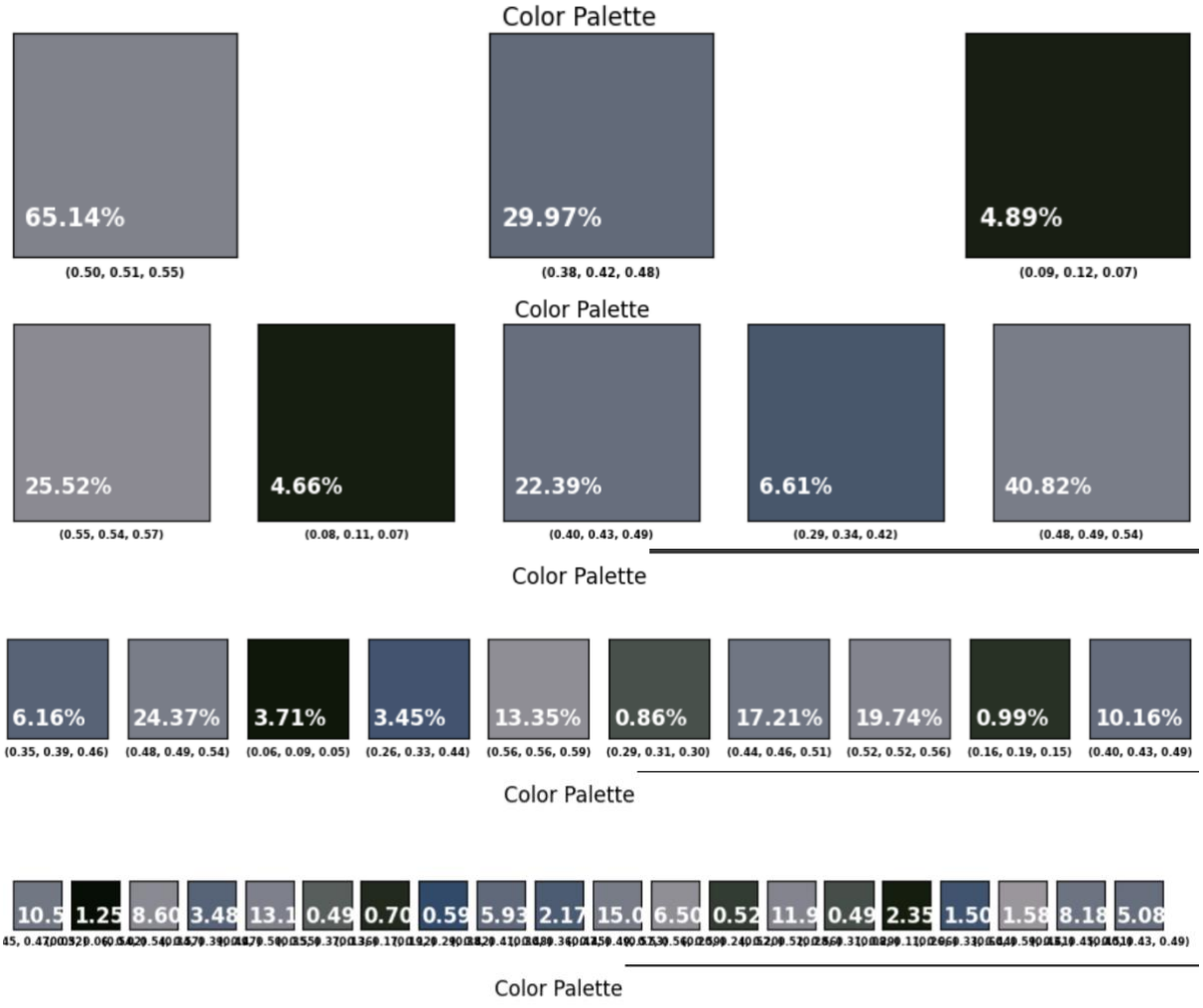
Tiger


Distribution of Cluster Labels with Colors
Frequency
Cluster Label

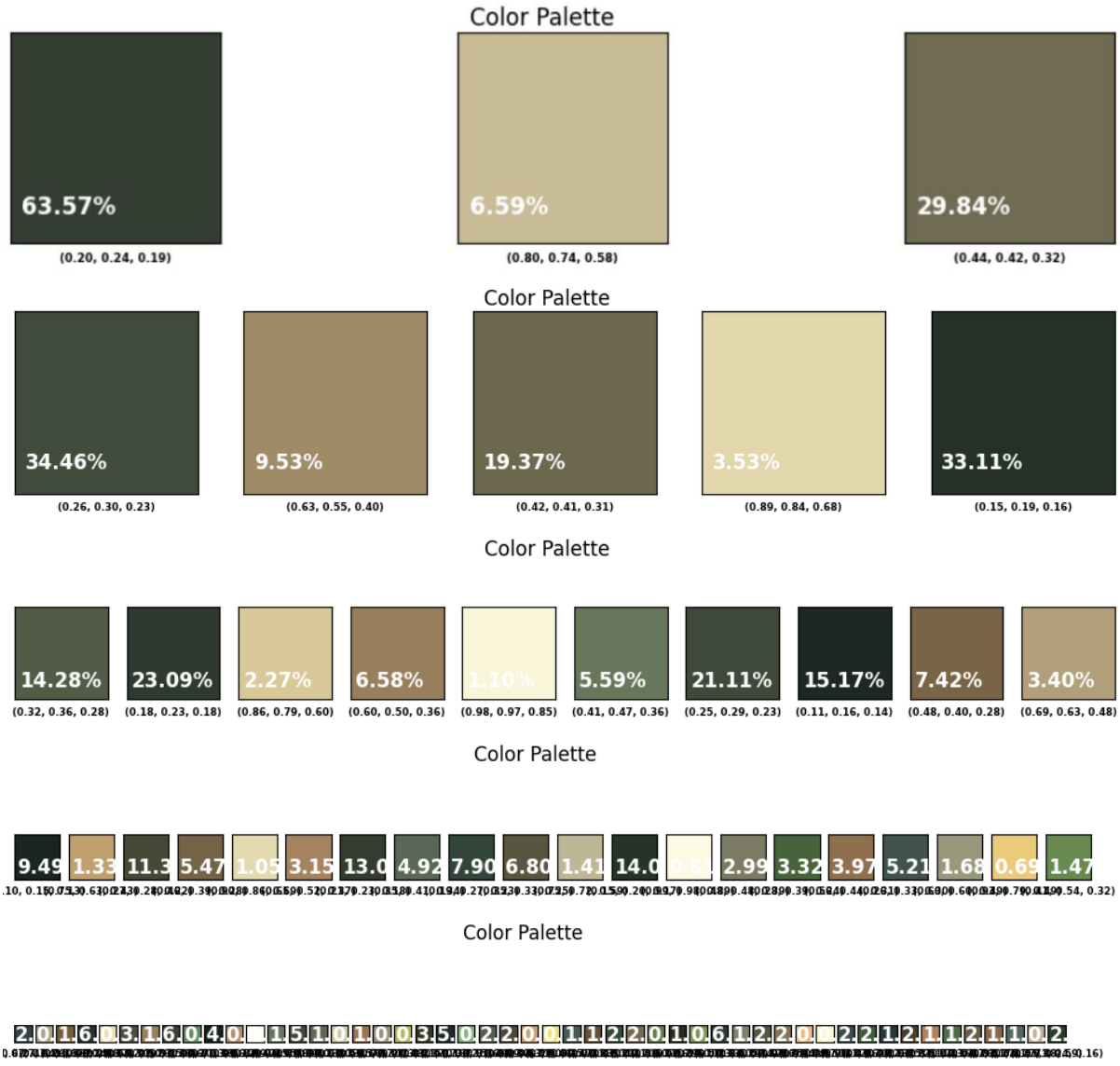


3. Color Palette Extraction

Airplane



Tiger



B. Discussion

In this study, we applied the K-Means clustering algorithm with varying numbers of clusters (3, 5, 10, 20, and 50) for image compression and color palette extraction. Through this proces s, we encountered several challenges, limitations, and lessons learned, which are discussed bel ow.

1. Selection of the optimal number of clusters: One of the main challenges in using the

K-Means algorithm is determining the appropriate number of clusters for a given image. While increasing the number of clusters can result in a more accurate representation of the original image, it may also lead to overfitting and increased computational complexity. On the other hand, using too few clusters can oversimplify the image and reduce its visual quality. We learned that selecting the optimal number of clusters is a trade-off between image fidelity and computational efficiency.

2. Initialization sensitivity: The K-Means algorithm is sensitive to the initial placement of cluster centroids, which can impact the final clustering results. In our experiments, we set a fixed random state for reproducibility, but this may not always lead to the best possible clustering. We learned that using techniques like K-Means++ for smart initialization or running the algorithm multiple times with different initializations can help overcome this limitation.
3. Cluster shape and distribution: K-Means assumes that clusters are spherical and have similar sizes. This assumption may not always hold true for image data, leading to suboptimal clustering results. In our experiments, we observed that some cluster shapes and distributions may not be well-represented by the K-Means algorithm. In such cases, alternative clustering techniques like DBSCAN or Gaussian Mixture Models could provide better results.
4. Evaluation of clustering quality: Quantitatively assessing the quality of the clustering results is a challenging task. We learned that using metrics such as the silhouette score, Calinski-Harabasz index, or Davies-Bouldin index can help in evaluating the clustering performance and comparing different clustering configurations.
5. Computational complexity: As the number of clusters increases, the computational complexity of the K-Means algorithm also grows. We observed that the algorithm took longer to converge when using a higher number of clusters, which could be a concern for large-scale or real-time applications. We learned that parallelization, dimensionality reduction techniques, or approximated algorithms like MiniBatch K-Means can help address this issue.

In conclusion, our experiments with various K-Means clustering configurations offered invaluable insights into the challenges, limitations, and lessons learned when applying the algorithm to image compression and color palette extraction tasks. The exploration of different cluster numbers, such as 3, 5, 10, 20, and 50, demonstrated the impact of these configurations on the algorithm's performance and results. These experiments have been instrumental in deepening our understanding of the K-Means algorithm and the intricacies of clustering. This knowledge will not only benefit future research in refining and optimizing the K-Means algorithm but also contribute to the development of more efficient and robust solutions for image processing tasks across various domains.

References:

- Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: analysis and implementation. IEEE transactions on pattern analysis and machine intelligence.
- Leung, T., & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. International journal of computer vision.
- Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. IEEE Transactions on systems, man, and cybernetics.

- Orchard, M. T., & Bouman, C. A. (1991). Color quantization of images. IEEE Transa ctions on Signal Processing.
- Wu, X., Zhang, N., & Steinbach, M. (1992). A new class of suboptimal clustering alg orithms: Sequential optimization and relaxation. Journal of Classification.