Computer Vision Project Assignment 1: Dimensionality Reduction, Image Retrieval, and Classification

Name: Sare Naz
Surname: Bayraktutan
ID:21992957

Dimension Reduction

Dimension reduction techniques, including Principal Component Analysis (PCA), are deemed to play a crucial role in the management of high-dimensional data by being transformed into a lower-dimensional space. The primary objective of these techniques is the extraction and retention of essential information while diminishing the number of features or dimensions. PCA, as a mathematical technique employed for dimensionality reduction, is characterized by the identification and computation of principal components—linear combinations of the original features. These principal components, capturing the maximum variance in the data, enable a more compact representation.

Importance of Dimension Reduction

- **Computational Efficiency:** Increased computational complexity is often observed with high-dimensional data. The reduction of dimensions contributes to a decrease in the computational cost of algorithms, enabling more efficient processing.
- **Visualization:** Visualizing data in two or three dimensions is feasible, but beyond that, challenges arise. Dimension reduction is employed to facilitate the creation of meaningful visualizations, contributing to data exploration and interpretation.
- **Noise Reduction:** High-dimensional data may contain noise or irrelevant features. The filtering out of noise and retention of only the most significant information is facilitated by dimension reduction.
- **Overfitting Mitigation:** In machine learning models, overfitting can occur when the model learns noise in the training data. Mitigation of overfitting is facilitated by dimension reduction, which focuses on the most relevant features.
- **Improved Generalization:** Models trained on lower-dimensional data often generalize better to unseen data. The creation of more robust and generalizable models is facilitated by dimension reduction.
- **Collinearity Management:** Collinearity, where features are highly correlated, can impact model interpretability. The mitigation of collinearity issues is achieved by dimension reduction, which captures the underlying structure of the data.

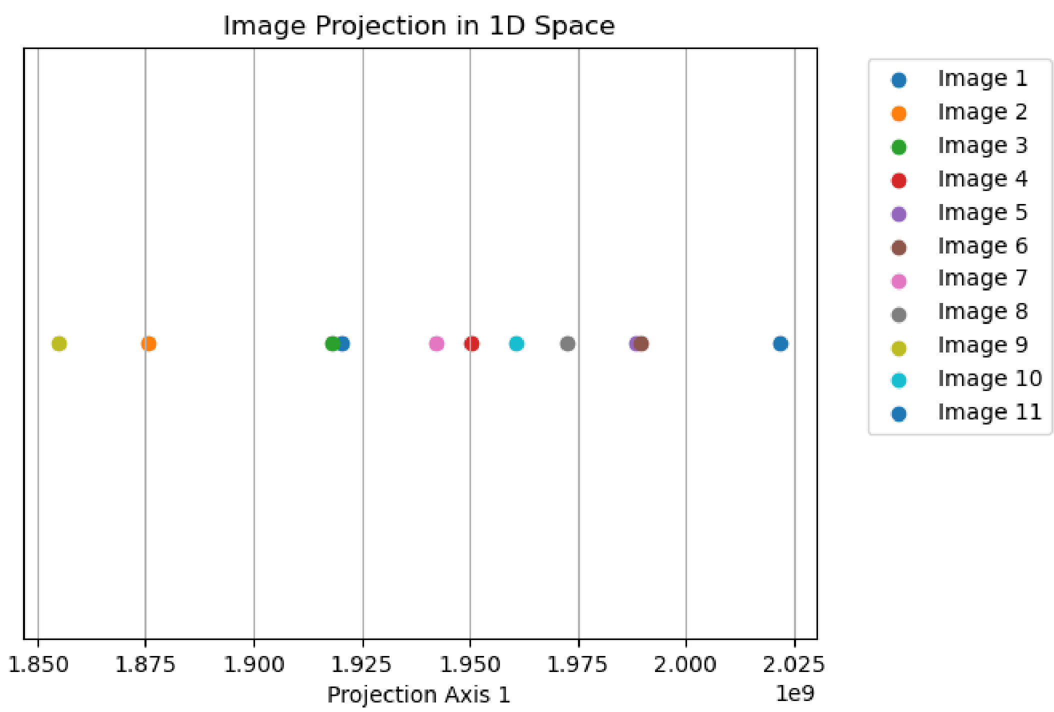
Principal Component Analysis (PCA) Algorithm

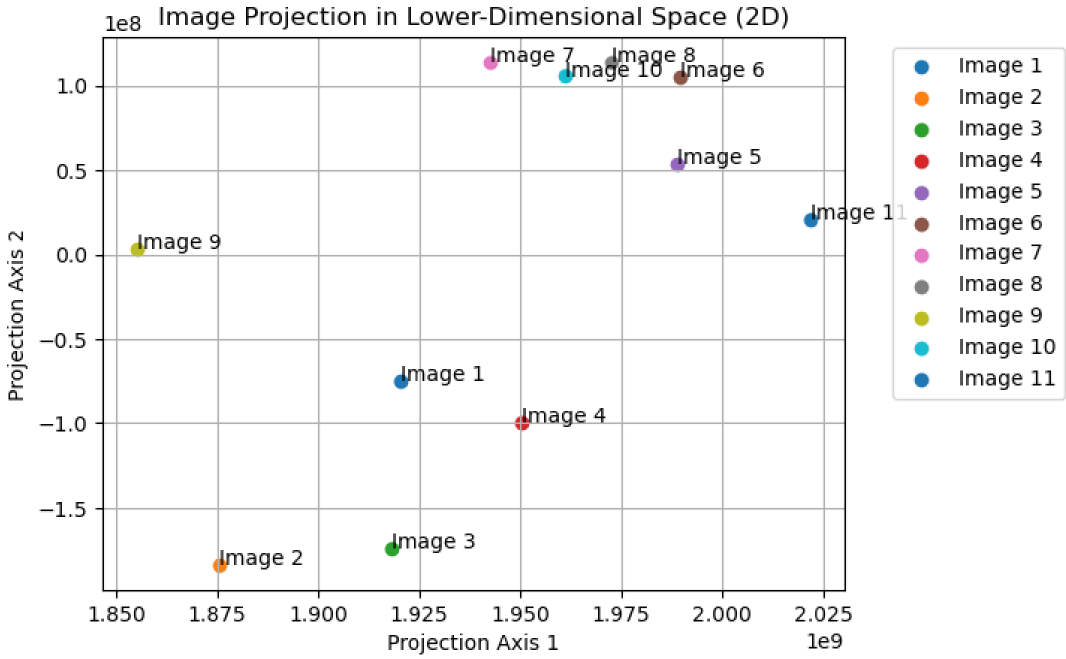
- **Data Standardization:** The standardization of the data, ensuring that all features have the same scale, marks the initiation of PCA. This is achieved by subtracting the mean and dividing by the standard deviation for each feature.

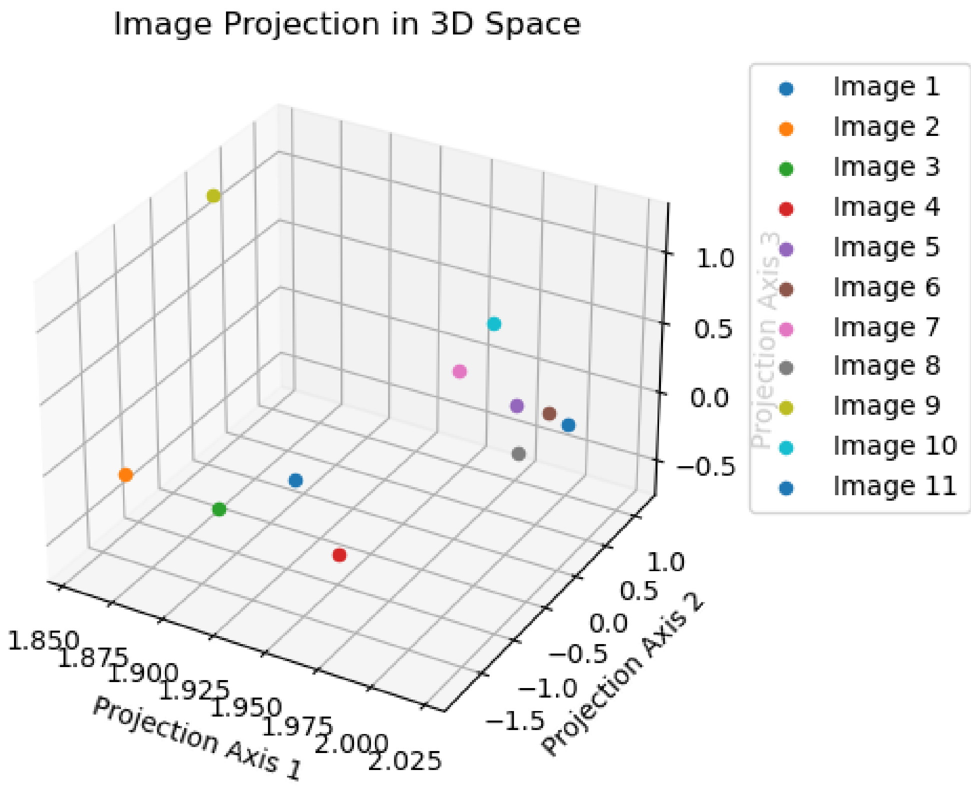
- **Covariance Matrix Calculation:** The computation of the covariance matrix is based on the standardized data. Insights into the relationships between different features are provided by the covariance matrix, indicating whether they tend to increase or decrease together.
- **Eigenvalue and Eigenvector Computation:** The next step involves finding the eigenvalues and eigenvectors of the covariance matrix. The variance along the principal components is represented by the eigenvalues, and the directions of maximum variance are represented by the corresponding eigenvectors.
- **Sorting Eigenvalues:** The eigenvalues are sorted in descending order. This sorting process is crucial as it helps identify the principal components in order of significance. The higher the eigenvalue, the more variance is captured by the corresponding eigenvector. Sorting eigenvalues allows prioritization of the principal components that contribute the most to the dataset's overall variance.
- **Selecting Principal Components:** After sorting, the top k eigenvectors (where k is the desired dimensionality) are chosen as the principal components. These eigenvectors form a new basis for the data, capturing the essential information in a reduced-dimensional space. The selection of eigenvectors corresponds to the sorted eigenvalues, prioritizing the directions that contribute the most to the overall variance.

Dimensionality reduction is facilitated by these steps while preserving the essential structure and variability of the data.

Data Plotting and Analysis







Distribution of Data Points

The distribution of data points can be observed, and it can be easily seen that some of the points are closer, forming a cluster like images 6, 7, 8, and 10. These pictures will be seen in the following part, and it will be realized that they are similar. It will be proven that our PCA worked.

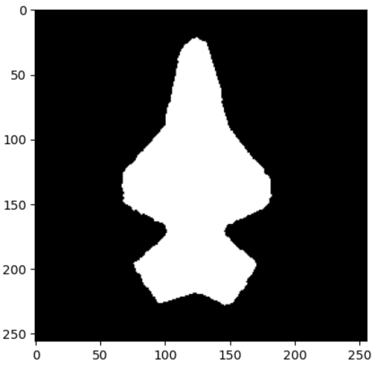


Image 6

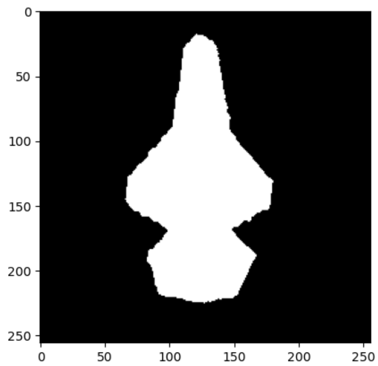


Image 7

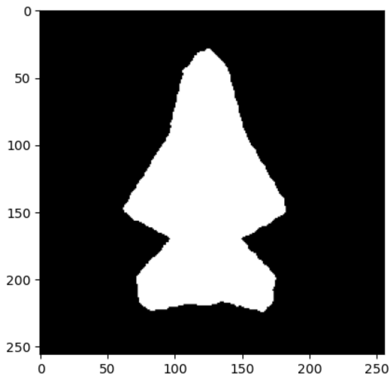


Image8

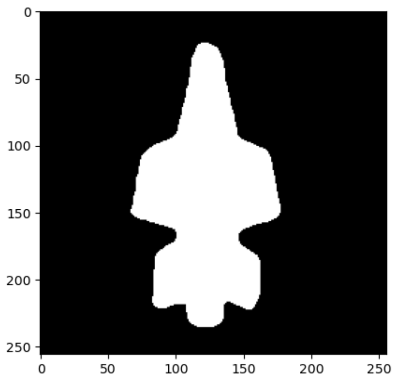


Image 10

Variance Captured by Axes

```
PC1: 85.89%
PC2: 4.82%
PC3: 2.07%
PC4: 1.61%
PC5: 1.32%
PC6: 0.39%
PC7: 1.06%
PC8: 0.91%
PC9: 0.72%
PC10: 0.57%
PC11: 0.64%
```

The proportion of variance explained by each principal component was examined. In the analysis of the sorted eigenvalues and corresponding eigenvectors, it was observed that the first principal component (PC1) played a pivotal role, capturing a substantial variance of 85.89%. This indicated its significance in representing a primary trend or pattern within the data. Subsequent principal components, notably PC2 (4.82%) and PC3 (2.07%), continued to contribute to the overall understanding of the dataset, albeit with diminishing individual impacts. The descending order of percentages highlighted a gradual decrease in the explanatory power of each successive principal component, emphasizing the diminishing role of components beyond PC3 in capturing essential information about the data's structure. The cumulative percentages of variance explained by these components provided insights into the potential for dimensionality reduction while retaining critical information in the dataset.

Data Separability

After PCA, the formation of a cluster comprising aircraft with shorter wings resembling more of an ellipse (clusters 6, 7, 8, 10) and another cluster consisting of aircraft with longer wings (clusters 2, 3) is observed. This signifies that PCA has been deemed successful, and accurate inferences about similarities have been drawn.

Outliers

Image 9 and Image 11 can be considered outliers as they do not appear to be close to any cluster.

Dimensionality Reduction Effectiveness

The effectiveness of PCA in reducing the dimensionality of the data while retaining essential information was evaluated. The 3D plot visually conveyed a comprehensive view of the dataset, showcasing the reduced complexity achieved through PCA. This evaluation aimed to assess the overall performance of PCA in simplifying the data structure while preserving critical information, providing insights into the method's utility for dimensionality reduction in this specific context.

Histogram and PCA Components

A histogram is presented as a graphical representation of the distribution of data, where the x-axis denotes the range of pixel values, and the y-axis indicates the frequency of pixels with each intensity value. In image processing, the distribution of pixel intensities in an image is depicted through histograms, allowing for the analysis of overall brightness and contrast.

Principal Component Analysis (PCA) is utilized for the transformation of high-dimensional data into a set of linearly uncorrelated variables known as principal components. These components, capturing the most significant sources of variation in the data, are arranged in such a way that the first principal component explains the maximum variance, followed by subsequent components. Widely employed for feature extraction and reduction, PCA facilitates the simplification of complex datasets while retaining essential information.

Image Retrieval Algorithm Logic

The image retrieval algorithm implemented involves the extraction of relevant features from images, the calculation of similarity metrics, and the subsequent retrieval of images based on their similarity to a given query image. The key steps in image retrieval are as follows:

- **Feature Extraction:** Features representing the visual characteristics of images are extracted.
- **Similarity Calculation:** Similarity between the features of the query image and other images in the dataset is computed using metrics such as Euclidean distance or cosine similarity.
- **Ranking:** Images are ranked according to their similarity to the query image, with higher similarity scores indicating greater relevance.
- **Retrieval:** The top-ranked images are retrieved as the results of the query.

In this algorithm, the following features are employed:

- **Color Histograms**: The distribution of colors in an image is quantized into bins, and the frequency of each color is represented.

```
def calculate_histogram(image):
    histogram = np.zeros((num_bins, 3), dtype=int)

    for channel in range(3):  # Loop over each channel (R, G, B)
        values = image[:, :, channel].flatten()
        histogram[:, channel] = [np.sum(values == i) for i in range(num_bins)]

    return histogram.flatten()
```

- **Principal Component Analysis (PCA):** PCA is utilized to reduce the dimensionality of the feature space, identifying principal components that capture significant variance. (This algorithm was not employed due to its cancellation.)
- **Euclidean Distance:** Similarity between the query image's features and those of other images is quantified using Euclidean distance.

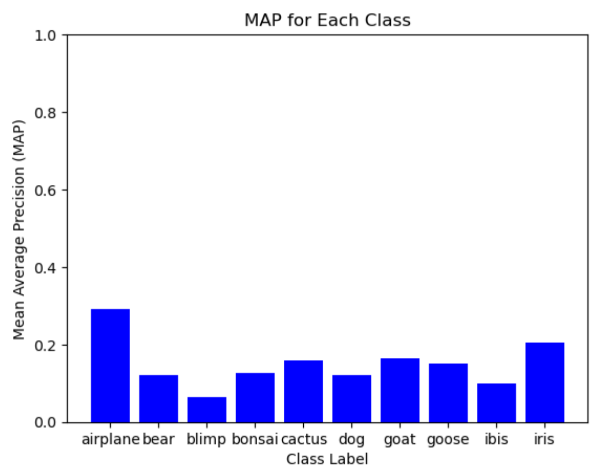
```
def calculate_euclidean_distance(vector1, vector2):
    squared_diff = np.square(vector1 - vector2)
    sum_squared_diff = np.sum(squared_diff)

    # Check for non-negative value before taking the square root
    if sum_squared_diff >= 0:
        euclidean_distance = np.sqrt(sum_squared_diff)
    else:
        # If sum_squared_diff is negative (due to numerical issues), set distance to a large positive value
        euclidean_distance = np.inf

    return euclidean_distance
```

These features are chosen due to their capability to capture distinctive aspects of images, with color histograms providing information about color distribution, PCA aiding in dimensionality reduction, and Euclidean distance serving as a metric for measuring feature vector similarity.

Results and Feature Analysis

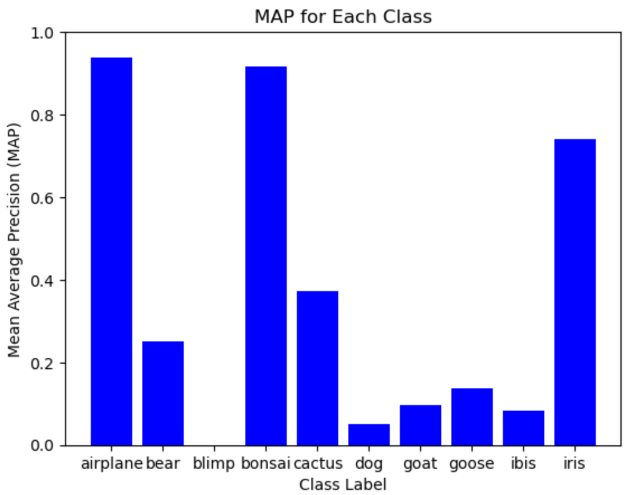
The results of the image retrieval algorithm were analyzed in terms of Class-wise Mean Average Precision (MAP) and the overall MAP across all classes. The MAP values for each class were computed as follows:

```
Class-wise Mean Average Precision:
airplane: 0.29122986038563947
bear: 0.12198380477368591
blimp: 0.06548257745749796
bonsai: 0.1273644988417558
cactus: 0.15786600443225174
dog: 0.12101000471629428
goat: 0.16555007742665268
goose: 0.1505029322969063
ibis: 0.09866691746392318
iris: 0.20431868552796761

Mean Average Precision (MAP) across
0.15039753633225747
```



These results were calculated using 300 dataset2 images. The subsequent results, however, were computed using only the top 10 most relevant images in the ranked list, done solely for experimental purposes.

```
Class-wise Mean Average Precision:
airplane: 0.9380208333333333
bear: 0.25
blimp: 0.0
bonsai: 0.9166666666666666
cactus: 0.3714285714285714
dog: 0.05
goat: 0.09722222222222221
goose: 0.13680555555555557
ibis: 0.08333333333333333
iris: 0.7416666666666667

Mean Average Precision (MAP) acros:
0.3585143849206349
```



Higher results are obtained because the first calculations incorporate irrelevant images. However, it is noted that the first calculation yields more consistent results compared to the second one.

The Mean Average Precision across all classes was determined to be 0.15039753633225747. The performance analysis suggests that the retrieval algorithm demonstrates varying levels of effectiveness across different classes. The algorithm performed relatively well for the "Airplane" and "Iris" classes, as indicated by higher MAP values, while other classes exhibited lower precision. The choice of features influenced the algorithm's capability to capture class-specific characteristics, contributing to the observed results.

Advantages/Disadvantages of Image Retrieval

The retrieved images were observed and evaluated to discern the advantages and disadvantages of the employed algorithm and representation methods. Several key observations can be highlighted:

Advantages:

Distinctive Features Captured: The algorithm, leveraging color histograms demonstrated the ability to capture distinctive features relevant to the queried classes. This is particularly evident in classes such as "Airplane" and "Iris," where the retrieved images exhibit visual similarities to the query.

Disadvantages:

Limited Discriminative Power: In certain classes, the algorithm struggled to precisely discriminate among visually similar objects. For instance, in classes like "Blimp" and "Bonsai," where objects may share common visual elements, the algorithm's precision was comparatively lower.

Sensitivity to Image Quality: The algorithm's performance may be influenced by variations in image quality, lighting conditions, and occlusions. Noisy or poorly illuminated images may lead to less accurate retrievals, impacting the overall effectiveness of the algorithm.

Dependency on Feature Selection: The success of the algorithm relies heavily on the choice of features. If the selected features fail to adequately capture class-specific characteristics, the algorithm may struggle to distinguish between classes, affecting the quality of retrieved images.

Color Space Variation for Color Histogram

The experiments with different color spaces for color histograms, specifically in HSV and LAB color spaces, have resulted in varying outcomes in terms of image retrieval performance. The Class-wise Mean Average Precision (MAP) and the overall MAP across all classes have been provided for each color space:

```
Class-wise Mean Average Precision:
airplane: 0.26747214933210334
bear: 0.1557153544100377
blimp: 0.08276212842198308
bonsai: 0.10379952577981805
cactus: 0.12408575352371382
dog: 0.0946801797316357
goat: 0.1741307673110537
goose: 0.15711546244855704
ibis: 0.1020376033669553
iris: 0.18912189174658822

Mean Average Precision (MAP) acros
0.14509208160724457
```

HSV Results

```
Class-wise Mean Average Precision:
airplane: 0.19174096592571607
bear: 0.16617152877107916
blimp: 0.10575073978663717
bonsai: 0.10175332640820053
cactus: 0.14046024929808715
dog: 0.10075663298846449
goat: 0.14480729417737817
goose: 0.11352548732865633
ibis: 0.10433020948200172
iris: 0.3330418010059729

Mean Average Precision (MAP) across
0.15023382351721937
```

LAB Results

Key Observations:

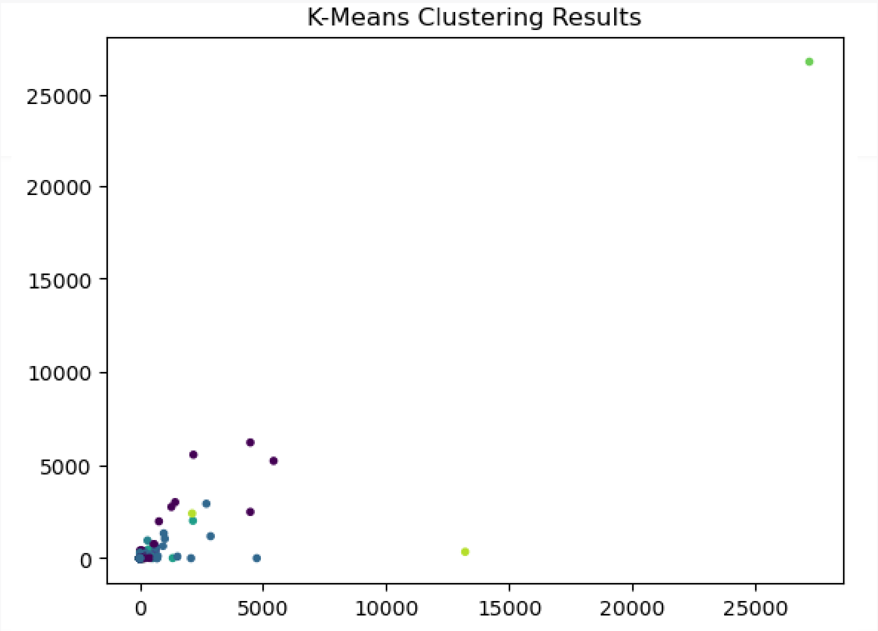
- Image retrieval performance is notably influenced by the selected color space.
- Higher MAP values across all classes are observed in the LAB color space compared to the HSV color space.
- Different color spaces capture distinct facets of color information, impacting the retrieval of relevant images.
- Consideration of color space as a parameter in image feature extraction is emphasized for tailoring approaches to specific image retrieval tasks.
- These observations underscore the significance of accounting for color space selection as a crucial factor in image feature extraction for tasks like image retrieval.

KMEANS

For the K-means algorithm, both dataset2 and query images were combined, and clusters were formed with both datasets. Each cluster was then displayed separately for analysis.

```
Cluster 1:
Query Images in Cluster:
  goose
  ibis
  iris
Dataset2 Images in Cluster:
  bear
  bear
  blimp
  blimp
  blimp
  blimp
  bonsai
  bonsai
  bonsai
  bonsai
```

```
Cluster 2:
Query Images in Cluster:
Dataset2 Images in Cluster:
  blimp
  iris

Cluster 3:
Query Images in Cluster:
  dog
Dataset2 Images in Cluster:
  blimp
  blimp
  blimp
  blimp
  blimp
  bonsai
  bonsai
```




```
Cluster 1:
Query Images in Cluster: [15 17 18]
Dataset2 Images in Cluster: [ 34  35  67  74  75  83 104 105 106 107 108 114 116 131 140 147 148 157
 161 163 170 199 200 203 233 243 247 253 260 264 266 274 277 286 288 290
 292 294 299]

Cluster 2:
Query Images in Cluster: []
Dataset2 Images in Cluster: [ 87 283]

Cluster 3:
Query Images in Cluster: [10]
Dataset2 Images in Cluster: [ 66  68  71  82  84 102 111]

Cluster 4:
Query Images in Cluster: [ 1  2  4  5  7  8 11 12 13 14 16 19]
Dataset2 Images in Cluster: [  0   7   8  10  21  27  31  33  36  37  38  41  42  44  46  47  50  52
  53  55  56  58  65  69  77  79  86  89  90  93  95  99 100 101 112 113
 119 121 124 125 126 127 128 129 130 132 133 135 136 137 138 141 142 143
 145 146 149 150 152 153 154 156 162 164 165 166 167 168 171 173 174 175
 176 179 180 182 183 184 185 187 189 190 191 192 193 195 196 197 198 202
 204 205 207 208 209 210 212 213 214 215 217 219 220 222 224 225 226 229
 231 232 234 238 239 240 241 242 245 246 248 249 250 251 252 255 256 261
 262 265 268 270 273 275 276 278 279 280 281 282 284 285 287 289 293 295
 296 297 298]
```

The clustering results indicate that the images have been grouped into 10 clusters using the k-means algorithm. Each cluster consists of both query and dataset2 images, and they are displayed along with their respective names or folder names and indexes.

- **Cluster Quality:** The quality of the clusters can be assessed by examining the similarity of image contents within each cluster. It seems that some clusters have a clear theme, with similar images grouped (Cluster 4 has various animals). However, other clusters may show mixed content, and this could be due to the nature of the feature vectors or the complexity of the dataset.
- **Interpretability:** The interpretability of the clusters depends on the chosen features for clustering. If the features effectively capture the visual characteristics of the images, the resulting clusters are likely to be more meaningful. It's essential to consider the domain-specific interpretation of the clusters to evaluate their practical usefulness.
- **Visual Inspection:** The scatter plot visualizes the clustering results in a 2D space, and while it provides a high-level overview, it may not capture the full complexity of the feature space. Consider visualizing in a higher-dimensional space or exploring other visualization techniques for a more comprehensive understanding.

MAP Metric Calculation and Analysis

The Mean Average Precision (MAP) metric was calculated for different color spaces, exploring their impact on image retrieval performance. The results and observations are summarized as follows:

- Original Color Space (RGB):

```
Class-wise Mean Average Precision:
airplane: 0.29122986038563947
bear: 0.12198380477368591
blimp: 0.06548257745749796
bonsai: 0.1273644988417558
cactus: 0.15786600443225174
dog: 0.12101000471629428
goat: 0.16555007742665268
goose: 0.1505029322969063
ibis: 0.09866691746392318
iris: 0.20431868552796761

Mean Average Precision (MAP) across
0.15039753633225747
```

The original RGB color space provided baseline performance. RGB, a widely used color space, may not capture certain color nuances effectively.

- HSV Color Space:

```
Class-wise Mean Average Precision:
airplane: 0.26747214933210334
bear: 0.1557153544100377
blimp: 0.08276212842198308
bonsai: 0.10379952577981805
cactus: 0.12408575352371382
dog: 0.0946801797316357
goat: 0.1741307673110537
goose: 0.15711546244855704
ibis: 0.1020376033669553
iris: 0.18912189174658822

Mean Average Precision (MAP) across
0.14509208160724457
```

Performance in HSV color space was slightly lower than RGB. HSV separates intensity, potentially impacting color-based retrieval.

- LAB Color Space:

```
Class-wise Mean Average Precision:
airplane: 0.19174096592571607
bear: 0.16617152877107916
blimp: 0.10575073978663717
bonsai: 0.10175332640820053
cactus: 0.14046024929808715
dog: 0.10075663298846449
goat: 0.14480729417737817
goose: 0.11352548732865633
ibis: 0.10433020948200172
iris: 0.3330418010059729

Mean Average Precision (MAP) across
0.15023382351721937
```

LAB color space showed comparable performance to RGB.
LAB, designed for perceptual uniformity, offers consistent color representation.

- Additional Experiment

In the additional experiment, emphasis was placed on utilizing the top 10 in the ranked list for image retrieval evaluation. The purpose was to assess the performance when considering a more focused subset of retrieved images and evaluate the precision of the retrieval algorithm in identifying relevant images within a more limited subset.

```
Class-wise Mean Average Precision:
airplane: 0.9380208333333333
bear: 0.25
blimp: 0.0
bonsai: 0.9166666666666666
cactus: 0.3714285714285714
dog: 0.05
goat: 0.09722222222222221
goose: 0.13680555555555557
ibis: 0.08333333333333333
iris: 0.7416666666666667

Mean Average Precision (MAP) acros:
0.3585143849206349
```

This experiment contributes to a nuanced understanding of how the algorithm performs when considering a more concentrated set of results.

Overall Observations:

- The choice of color space significantly influences image retrieval performance.
- HSV, separating intensity, may affect performance, especially if intensity information is crucial.
- LAB, designed to be perceptually uniform, offers comparable performance with RGB.
- Consider specific image characteristics and retrieval requirements when choosing a color space.

In conclusion, the evaluations suggest that the LAB color space performs comparably to RGB, while HSV shows slightly lower performance. The choice of color space should be tailored to image characteristics and retrieval task requirements.

Below are presented the results of all experiments in table format.

Precision Values:

Class	RGB	HSV	LAB	Experiment 4
airplane	0.29122986038563947	0.26747214933210334	0.19174096592571607	0.9380208333333333
bear	0.12198380477368591	0.1557153544100377	0.16617152877107916	0.25
blimp	0.06548257745749796	0.08276212842198308	0.10575073978663717	0.0
bonsai	0.1273644988417558	0.10379952577981805	0.10175332640820053	0.9166666666666666
cactus	0.15786600443225174	0.12408575352371382	0.14046024929808715	0.3714285714285714
dog	0.12101000471629428	0.0946801797316357	0.10075663298846449	0.05
goat	0.16555007742665268	0.1741307673110537	0.14480729417737817	0.09722222222222221
goose	0.1505029322969063	0.15711546244855704	0.11352548732865633	0.13680555555555557
ibis	0.09866691746392318	0.1020376033669553	0.10433020948200172	0.08333333333333333
iris	0.20431868552796761	0.18912189174658822	0.3330418010059729	0.7416666666666667

MAP Values:

Class	RGB	HSV	LAB	Experiment 4
MAP	0.15039753633225747	0.14509208160724457	0.15023382351721937	0.3585143849206349

10. Logistic Regression Algorithm Logic:

For this section, the classes chosen were ibis and airplane.

The logistic regression algorithm is based on the following key components:

- Sigmoid Function: The sigmoid function, denoted $\sigma(z) = \frac{1}{1+e^{-z}}$ plays a pivotal role in logistic regression. It transforms the input z into values between 0 and 1, representing probabilities.

```
# Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

- Logistic Regression Hypothesis: The hypothesis function defines the relationship between input features (X) and the output (y). It utilizes the sigmoid function to model the probability that y is 1 given X and the parameter vector θ.

```
# Logistic regression hypothesis
def hypothesis(theta, X):
    return sigmoid(np.dot(X, theta))
```

- Cost Function: The cost function assesses the performance of the hypothesis by quantifying the difference between predicted and actual values. It calculates the logistic loss, penalizing deviations from the true labels y.

```
def cost_function(theta, X, y, epsilon=1e-15):
    m = len(y)
    h = hypothesis(theta, X)
    h = np.clip(h, epsilon, 1 - epsilon)  # Clip values to avoid log(0) or log(1)
    return (-1 / m) * np.sum(y * np.log(h) + (1 - y) * np.log(1 - h))
```

- Gradient Descent:The optimization process involves minimizing the cost function through gradient descent. This iterative algorithm updates the parameter vector θ by computing the gradient of the cost function concerning θ. Adjustments are made in the direction that reduces the cost.

```
# Gradient descent
def gradient_descent(X, y, theta, learning_rate, num_epochs):
    m = len(y)
    cost_history = []

    for epoch in range(num_epochs):
        h = hypothesis(theta, X)
        gradient = np.dot(X.T, (h - y)) / m
        theta -= learning_rate * gradient
        cost = cost_function(theta, X, y)
        cost_history.append(cost)

    return theta, cost_history
```

- Predict Function: The predict function uses the learned parameters to predict binary outcomes based on a specified threshold. If the sigmoid output is greater than or equal to the threshold (typically 0.5), the prediction is set to 1; otherwise, it is 0.

```
# Predict function
def predict(theta, X, threshold=0.5):
    return (hypothesis(theta, X) >= threshold).astype(int)
```

- Training Logistic Regression Model: The model is trained using gradient descent on the training dataset. The learning rate and number of epochs govern the optimization process. The parameters (θ) are adjusted to minimize the logistic loss.

```
# Train logistic regression model
learning_rate = 0.01
num_epochs = 1000
theta, cost_history = gradient_descent(Train_x_normalized, Train_y_binary, theta, learning_rate, num_epochs)
```

- Prediction and Conversion: The trained model is applied to the test dataset to generate predictions. Numerical predictions are then converted back to class labels using a predefined mapping of numerical values to class labels.

```
# Predict using the learned parameters
predictions = predict(theta, Test_x_normalized)
```

```
# Convert numerical predictions back to class labels
predicted_labels = [unique_labels[i] for i in predictions]
```

- Normalization and Initialization: Feature normalization is performed to standardize the input data. The parameters (θ) are initialized to zeros before the training process.

```
# Normalize features
Train_x_normalized = (np.array(Train_x) - np.mean(Train_x)) / np.std(Train_x)
Train_x_normalized = np.c_[np.ones(Train_x_normalized.shape[0]), Train_x_normalized]
```

- Result Analysis: Learned parameters and predicted labels are printed to provide insights into the logistic regression model's performance.

```
# Print the predicted labels for the test set
print("\nPredicted Labels:")
print(predicted_labels)
```

11. Classification Results Analysis:

```
[61]: Test_y
```

```
[61]: ['airplane', 'airplane', 'ibis', 'ibis']
```

```
[62]: # Calculate accuracy
      correct_predictions = sum(1 for true_label, predicted_l
      accuracy = correct_predictions / len(Test_y) * 100

      # Print the predicted labels and accuracy for the test
      print("\nPredicted Labels:")
      print(predicted_labels)
      print("\nAccuracy: {:.2f}%".format(accuracy))
```

```
Predicted Labels:
['airplane', 'airplane', 'ibis', 'ibis']

Accuracy: 100.00%
```

The observed accuracy of 100.00% in the classification results is notable, but it's essential to acknowledge that the dataset's limited size might contribute to inflated accuracy. In scenarios with a minimal number of test cases, achieving perfect accuracy becomes relatively easier due to fewer instances of potential misclassifications. Therefore, while the current result is promising, it's important to interpret it within the context of the dataset's size and recognize that similar performance might not generalize seamlessly to larger and more diverse datasets.

12. Advantages/Disadvantages of Classification:

The advantages of the classification method lie in its ability to achieve a high accuracy of 100.00%, showcasing its proficiency in distinguishing between different classes. This accuracy, however, needs to be interpreted cautiously, considering the limited size of the dataset. The method demonstrates efficacy in the specific context of the available test cases.

One potential disadvantage is the risk of overfitting, especially with a small dataset. Overfitting may lead to a model that is too tailored to the training instances, making it less robust in handling unseen data. Additionally, the method's performance might not generalize well to more extensive and diverse datasets. Further experimentation with larger datasets is necessary to comprehensively assess the method's strengths and limitations.

REFERENCES

1-) https://piazza.com/class_profile/get_resource/ln8iuoi1cwv7dn/lo4l7yg8y2h3rf
2-) https://iq.opengenus.org/algorithm-principal-component-analysis-pca/
3-) https://www.coursera.org/lecture/pca-machine-learning/steps-of-pca-RaNDs?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi&campaignid=20665163467&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemodel=&adposition=&creativeid=&hide_mobile_promo&gclid=CjwKCAiA0syqBhBxEiwAeNx9NxyMtEnCbIikHFRGZBvSbMai7zlgOSMo1vbF_hFKxrgVNU9W4O3xuBoCcjMQAvD_BwE
4-) https://www.pinecone.io/learn/series/image-search/color-histograms/
5-) https://github.com/AdityaShaha/CBIR-Using-CDH
6-) https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
7-) https://realpython.com/logistic-regression-python/