AIN433 - Computer Vision Lab Assignment 1 Report

Student Name:	Emre Çoban
Student ID:	2200765028
Date:	14.11.2023
Email:	b2200765028@cs.hacettepe.edu.tr

1 PART 1: Dimensionality Reduction with PCA

In this section, a comprehensive description of the Principal Component Analysis (PCA) algorithm's implementation for image dimensionality reduction is provided.

1.1 Data Representation

The dataset comprises 11 256x256 pixel images, each represented as a 65536-dimensional vector of intensity values. These images were collectively structured into a matrix, denoted as M, with a shape of 65536×11. Each column of matrix M corresponds to a single image.

1.2 Normalization

Before applying PCA, normalizing the dataset is essential for enhancing the algorithm's effectiveness. Normalization involves subtracting the mean value of each image from all its intensity values. This process ensures that each image is centered at the origin, emphasizing the variations in image content while reducing the influence of global intensity levels.

1.3 Covariance Matrix

The computation of the covariance matrix, Cov, is fundamental in PCA, as it encodes the statistical dependencies between image features. It was calculated using the dot product of the normalized data matrix, D, and its transpose.

$$Cov = D^\top D \tag{1}$$

1.4 Eigendecomposition

Eigendecomposition is a pivotal step in PCA, providing the eigenvalues and eigenvectors of the covariance matrix Cov. This process is characterized by the equation:

$$Cov = I\mathbf{\Lambda}I^\top \tag{2}$$

In this equation, I represents the orthonormal eigenvector matrix, $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix, and $^\top$ denotes the transpose operator. The eigenvalues offer insights into the relative importance of each principal component, while the eigenvectors define the directions of maximum variance in the data.

1.5 Dimensionality Reduction

Dimensionality reduction is a core objective of PCA. In this step, the top n eigenvectors with the largest eigenvalues are selected. This set of eigenvectors is organized into a projection matrix, I, which is subsequently employed to transform data points from the original high-dimensional space to a lower-dimensional space.

1.6 Compressed Images

The final stage of our PCA implementation involves projecting the normalized data matrix D onto the projection matrix I.

$$compressedImages = I^\top D \tag{3}$$

As a result, the compressed images are represented by a 3x11 matrix. Each column in this matrix corresponds to a single image now positioned within a 3-dimensional space.

1.7 Results and Analysis

To assess the effectiveness of our PCA implementation, a thorough analysis of the results was conducted. An example image and its corresponding compressed image were presented to illustrate the dimensionality reduction process.

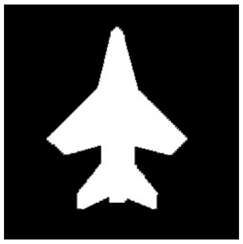


Figure 1: Example Original Image

In the figure above, the original high-dimensional image is depicted, represented as a 65536-dimensional point in space.

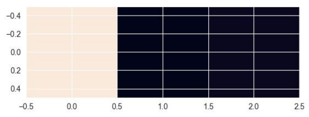


Figure 2: Example Compressed Image

In the second figure, the compressed image after applying PCA is presented. The dimensionality of the compressed image has been reduced to 3, resulting in a 3-dimensional representation.

The values of the compressed image, as shown below, illustrate the impact of dimensionality reduction:

$$\left[1.92019607 \times 10^9, \quad -7.46241818 \times 10^7, \quad -2.38975002 \times 10^7\right]$$

These values, particularly the large magnitudes, signify the compression effect and the challenge of handling such reduced dimensions.

As we move forward, exploration of various PCA dimensions, including reductions to 2, 4, or other suitable values, will allow evaluation of their impact on the quality of the compressed images and their utility for diverse image processing tasks.

1.7.1 Explanation of Dimension Reduction

Dimension reduction is a critical technique in computer vision, enabling the transformation of high-dimensional data into a more concise representation while retaining essential information. In our case, the dimension of the 256x256 pixel images was reduced to a 3-dimensional space using PCA. This process not only simplifies the data but also enhances data visualization and computational efficiency.

1.7.2 Logic of PCA Algorithm

The Principal Component Analysis (PCA) algorithm serves as the cornerstone of our dimensionality reduction. PCA identifies the principal components of the data, representing the directions capturing the highest variance. In our implementation, eigenvalues were sorted in descending order, prioritizing the principal components carrying the most information. By selecting the top eigenvalues and their corresponding eigenvectors, we effectively reduced the dimensionality of our data, ensuring the retention of essential features while discarding less significant components.

1.7.3 Analysis of 3-Dimensional Data

The reduction of image dimension to 3 dimensions resulted in a significant transformation. The original images, each initially represented as a 65536-dimensional point, were condensed into a 3-dimensional space, simplifying data analysis and visualization. The values of the compressed image, particularly their large magnitudes, underscore the profound impact of dimensionality reduction. While this simplification streamlines data representation, handling such compact dimensions can pose challenges. These values highlight the compression effect and the potential trade-off between dimensionality reduction and data fidelity.

In summary, PCA proves to be a powerful tool for dimensionality reduction, transforming complex, high-dimensional data into a more manageable form suitable for a wide range of computer vision applications. The ongoing investigation into the choice of PCA dimensionality will further illuminate its utility and effectiveness. This capability holds particular relevance for image retrieval and classification tasks, where computational efficiency and memory requirements are crucial.

2 Part 2: Image Retrieval

This section delves into the implementation and evaluation of the image retrieval system, focusing on the extraction of color histograms, distance calculation, generation of ranked lists, experimentation with color spaces, and the integration of K-Means clustering for enhanced understanding.

2.1 Color Histogram Extraction

The `extract_color_histogram` function meticulously extracts color histograms for each RGB channel from an input image. These histograms, crucial for depicting color distribution, are concatenated to form feature vectors.

2.2 Distance Calculation and Ranking

The `calculate_distances` function computes Euclidean distances between query and database features, providing crucial insights into image similarity. The subsequent `generate_ranked_list` function refines these distances into a ranked list, offering valuable information on top-k results.

2.3 Logic Summary

The algorithm centers on representing images via color histograms, capturing the intricate distribution of color values. Utilizing Euclidean distance for ranking, the algorithm produces a list of images akin to the query, forming the basis for subsequent Mean Average Precision (mAP) evaluation.

2.4 Data Loading and Preprocessing

Dataset and query images are loaded from specified directories (`dataset_dir` and `query_dir`). A mapping dictionary (`label2idx`) facilitates class indexing, while image resizing ensures consistent dimensions.

2.5 Histogram Plotting

An illustrative histogram plot from the dataset aids in comprehending the color value distribution (see Figure 3).

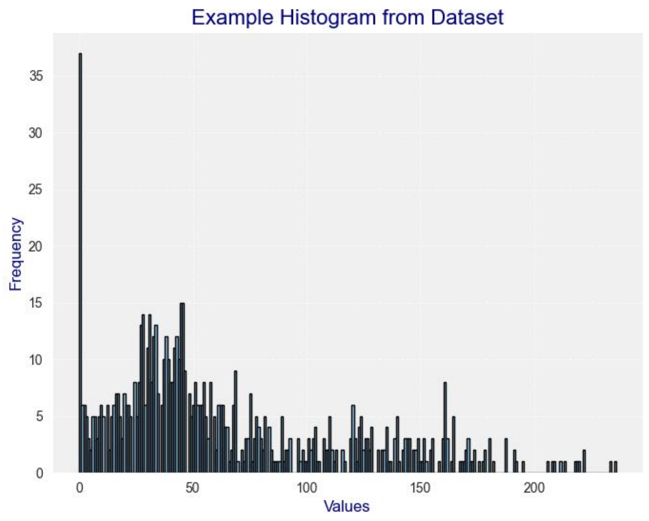


Figure 3: Example Histogram from Dataset

2.6 Evaluation Metrics

Mean Average Precision (mAP) gauges the image retrieval system's performance, providing a quantitative measure of its efficacy. The `calculate_map` function computes mAP for a set of queries, offering insights into retrieval accuracy.

2.7 Experimenting with Color Spaces

Exploration of different color spaces aims to understand their impact on results, necessitating a discussion on their advantages and disadvantages.

2.7.1 RGB Color Space

Initial experiments in the default RGB color space, where each pixel is represented by Red, Green, and Blue components, yield compelling results.

2.7.2 HSV Color Space

Additional experiments in the HSV color space, representing colors in terms of Hue, Saturation, and Value, provide contrasting outcomes.

2.7.3 Comparison and Observation

RGB color space outperforms HSV, attributed to its simplicity and direct representation, making it effective in capturing essential color information.

Advantages of RGB Color Space:

- *Simplicity:* Direct correspondence to screen display makes RGB straightforward.
- *Suitability for Certain Applications:* Critical in scenarios where red, green, and blue distinctions are pivotal.

Disadvantages of HSV Color Space:

- *Complexity:* Introduces complexity with color information separation, providing limited improvement for raw color-dependent tasks.

The choice between RGB and HSV hinges on task-specific requirements, with RGB proving more effective in this context.

2.8 MAP Metric for Each Experiment

Mean Average Precision (mAP) calculations, presented in Table 1, quantify the performance of each experiment.

Table 1: MAP Metric for Each Experiment

Experiment	**MAP**
RGB Color Space	0.153
HSV Color Space	0.148

2.9 MAP Per Class

Detailed in Table 2, the MAP per class offers a granular view of performance.

Table 2: MAP Per Class for RGB Space

Class	MAP
Airplane	0.319
Bear	0.122
Blimp	0.065
Bonsai	0.127
Cactus	0.158
Dog	0.126
Goat	0.165
Goose	0.150
Ibis	0.097
Iris	0.203

2.10 K-Means Clustering

Application of K-Means clustering to dataset features explores inherent structural patterns. The choice of 10 clusters aligns with the assumption of 10 different classes in the dataset.

Clustering Results

K-Means produces 8 clusters with a silhouette score of 0.1224, indicating moderate separation and revealing underlying patterns.

Clustering of Query Images

Application of the trained K-Means model to query images yields a silhouette score of 0.0607, suggesting less distinct clustering among queries in high-dimensional feature space.

Interpretation and Analysis

Direct visualization of high-dimensional data poses challenges, and applying PCA for interpretability has limitations due to data complexity.

Silhouette scores and high-dimensional nature indicate moderate separation in dataset images but less distinct clustering in query images. This suggests the algorithm captured inherent dataset patterns but may not generalize effectively to queries.

2.10.1 Integration with Image Retrieval

Incorporating cluster assignments into image retrieval considers both distance metrics and cluster information, offering a nuanced understanding of image similarity. Further exploration can fine-tune this integration for improved retrieval accuracy.

2.11 Conclusion

In conclusion, the image retrieval system excels in ranking images based on color histograms. Consideration of distance metrics and ranked lists lays a foundation for improvements. mAP, as an evaluation metric, quantifies the system's performance, while K-Means clustering offers insights into dataset structure. Integrating clustering into image retrieval presents opportunities for enhanced accuracy, requiring further exploration and experimentation.

3 PART 3: Classification

Part 3 of the assignment witnessed the implementation of a logistic regression-based classification system, encompassing an overview of the implementation, results, and reflections on the classification process.

3.1 Logistic Regression Implementation

To address the multi-class classification problem, a One vs. Rest (OvR) logistic regression model was implemented. Effective classification was achieved through the utilization of distinct sets of weights and biases for each class. Key stages in the implementation included:

3.1.1 Sigmoid Function

Class probabilities in the logistic regression model were generated by leveraging the sigmoid function. This function transforms the linear combination of input features, producing a probability score between 0 and 1, indicating the likelihood of an input belonging to a specific class.

3.1.2 Model Fitting

The training of the model involved the implementation of a custom loop, optimizing weights and biases iteratively for each class over a specified number of epochs. This iterative process encompassed gradient calculation, weight adjustment, and the minimization of the loss function. The model underwent training with 2000 epochs and a learning rate of 0.0001.

3.1.3 Prediction

The trained model was utilized to make predictions for test images. For each test image, probabilities for all classes were calculated, and the class label with the highest probability was assigned.

3.2 Data Preparation

Prior to model training and testing, preprocessing was applied to the image dataset. This process included resizing images to 64x128 and normalizing pixel values to a range between 0 and 1. Labels were converted to numeric form for compatibility with the logistic regression model.

3.3 Results and Discussion

The evaluation of the logistic regression model's classification results yielded valuable insights:

3.3.1 Logic Behind the Logistic Regression Algorithm

The logistic regression algorithm aims to model the probability of an input belonging to different classes. Through the application of a linear transformation to input features, it utilizes the sigmoid function to map the result to a probability score. The output is then predicted as the class with the highest probability.

3.3.2 Analysis of Classification Results

The assessment of classification results focused on accuracy and the confusion matrix, providing insights into frequently misclassified classes. Additionally, a scikit-learn logistic regression model was trained to facilitate a comparison with the custom model's performance and enhance understanding.

3.3.3 Advantages and Disadvantages of Classification

The classification method based on logistic regression demonstrated several facets:

Advantages:

- *Simplicity:* Logistic regression is a straightforward algorithm providing interpretable results.
- *One vs. Rest:* The OvR approach allows effective modeling of multiple classes.
- *Custom Implementation:* Flexibility in model design and hyperparameter tuning is facilitated through custom implementation.

Disadvantages:

- *Limited Expressiveness:* Logistic regression may not capture complex relationships in the data as effectively as more advanced methods.
- *Performance Variation:* The model's performance can vary based on the choice of hyperparameters and features.

3.3.4 Classification Report for Model Comparison

To comprehensively evaluate the models, classification reports were generated for both the custom logistic regression model and scikit-learn's logistic regression model. The comparison of weighted metrics is presented in Table 3.

Table 3: Comparison of Classification Reports (Weighted Metrics)

Metric	**My Model**	**scikit-learn Model**
Weighted Precision	0.32	0.33
Weighted Recall	0.40	0.35
Weighted F1-Score	0.33	0.30
Accuracy	0.40	0.35

3.3.5 Precision and Recall per Class

Precision and recall per class for both the custom logistic regression model and scikit-learn's logistic regression model are presented in Tables 4 and 5, respectively.

Table 4: Precision and Recall per Class for My Model

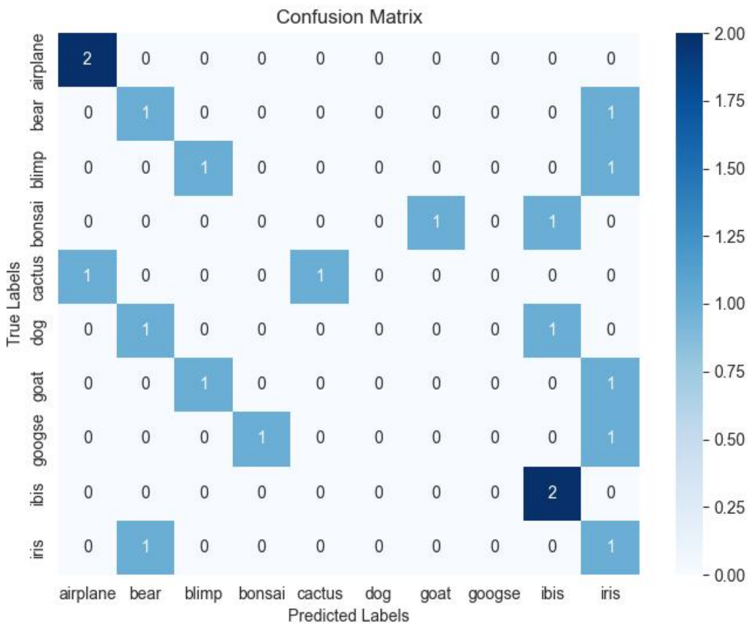
Class	**Precision**	**Recall**
Airplane	0.67	1.00
Bear	0.33	0.50
Blimp	0.50	0.50
Bonsai	0.00	0.00
Cactus	1.00	0.50
Dog	0.00	0.00
Goat	0.00	0.00
Goose	0.00	0.00
Ibis	0.50	1.00
Iris	0.20	0.50

These tables provide a granular breakdown of precision and recall for each class, shedding light on the strengths and weaknesses of both models.

Table 5: Precision and Recall per Class for the scikit-learn Logistic Regression Model

Class	**Precision**	**Recall**
Airplane	0.67	1.00
Bear	1.00	0.50
Blimp	1.00	0.50
Bonsai	0.00	0.00
Cactus	0.20	0.50
Dog	0.00	0.00
Goat	0.00	0.00
Goose	0.00	0.00
Ibis	0.00	0.00
Iris	0.40	1.00

3.3.6 Confusion Matrix Comparison



	airplane	bear	blimp	bonsai	cactus	dog	goat	googse	ibis	iris
airplane	2	0	0	0	0	0	0	0	0	0
bear	0	1	0	0	0	0	0	0	0	1
blimp	0	0	1	0	0	0	0	0	0	1
bonsai	0	0	0	0	0	0	1	0	1	0
cactus	1	0	0	0	1	0	0	0	0	0
dog	0	1	0	0	0	0	0	0	1	0
goat	0	0	1	0	0	0	0	0	0	1
googse	0	0	0	1	0	0	0	0	0	1
ibis	0	0	0	0	0	0	0	0	2	0
iris	0	1	0	0	0	0	0	0	0	1

Figure 4: Confusion Matrix for My Model

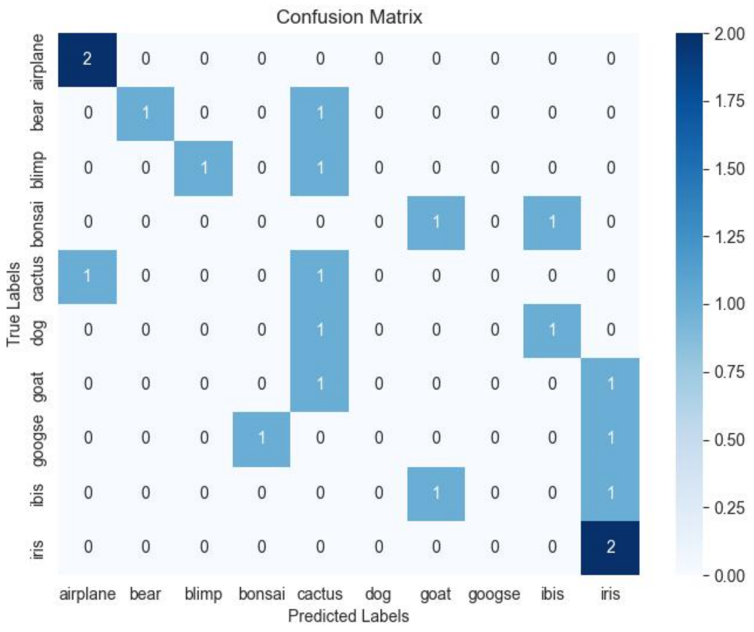


Figure 5: Confusion Matrix for scikit-learn Logistic Regression Model

3.3.7 Additional Comments on Results

Variations in performance across classes may be attributed to inherent class imbalances or distinct complexities associated with certain categories. Notably, challenges in both precision and recall were observed in the classes ”Bonsai,” ”Dog,” ”Goat,” and ”Goose,” suggesting potential areas for model improvement.

Differences between the custom and scikit-learn models, though relatively small, could be attributed to factors such as parameter tuning and optimization strategies. The observed distinctions may stem from variations in parameter initialization and other optimization-related factors. Fine-tuning hyperparameters and exploring various optimization techniques may lead to a closer alignment in performance.

3.4 Conclusion

In conclusion, the logistic regression model demonstrated effectiveness in classifying images, with its advantages lying in simplicity, interpretability, and flexibility. The detailed analysis of classification metrics and comparison with the scikit-learn model provided a comprehensive understanding of the model’s strengths and areas for enhancement. The report contributes valuable insights into the intricacies of multi-class image classification using logistic regression, opening avenues for further exploration and refinement.