ASSIGNMENT 2: Computer Vision 792

1 PCA feature vectors and kNN classification

1.1 Description

The first task involves building a simple face recognition system using the PCA feature vectors derived from the cropped Georgia Tech face database. The steps taken in this task include:

- Resizing and cropping images to a fixed size of 150 by 150.
- Setting aside 5 random images from each individual as a test set.
- Using the remaining images to calculate an average vector a and basis U_α.
- Plotting singular values of matrix X to determine a suitable value of α.
- Reconstructing images from their feature vector representations.
- Converting all images to feature vectors and using a kNN classifier for classification while varying the hyper-parameter k.

In my implementation of the `feature_vector` in `helper.py` function, I perform the substantial implementation to process and analyze image data. Here's an explanation of my approach:

1.1.1 Image Processing and Vectorization

First, I process the input images and convert them into vectors:

$$\text{vectors} = \text{process\_images}(\text{input}) \tag{1}$$

Each vector in this list represents an image, flattened into a column vector of size 67500 (which corresponds to a 150×150 image).

1.1.2 Average Vector Calculation

Next, I calculate the average vector across all images:

$$\text{avg\_value} = \frac{1}{N} \sum_{i=1}^{N} \text{vectors}_i \tag{2}$$

where N is the number of images (250 in this case).

1.1.3 Centered Data Matrix Calculation

I then calculate the centered data matrix X by subtracting the average vector from each image vector:

$$x_i = \text{vectors}_i - \text{avg\_value} \tag{3}$$

$$X = \frac{1}{\sqrt{N}} [x_1 \,|\, x_2 \,|\, \cdots \,|\, x_N] \tag{4}$$

1.1.4 Singular Value Decomposition (SVD)

I perform SVD on the centered data matrix X:

$$X = U \Sigma V^T \tag{5}$$

1.1.5 Basis Selection

I select the first α columns of U as my basis U_α. In this implementation, $\alpha = 50$:

$$U_\alpha = [u_1 \mid u_2 \mid \cdots \mid u_\alpha] \tag{6}$$

1.1.6 Feature Vector Calculation

I calculate the feature vectors y for each image:

$$y_i = U_\alpha^T x_i \tag{7}$$

1.1.7 Image Reconstruction

Finally, I reconstruct the images from their feature vector representations:

$$\hat{f}_i = U_\alpha y_i + \text{avg\_value} \tag{8}$$

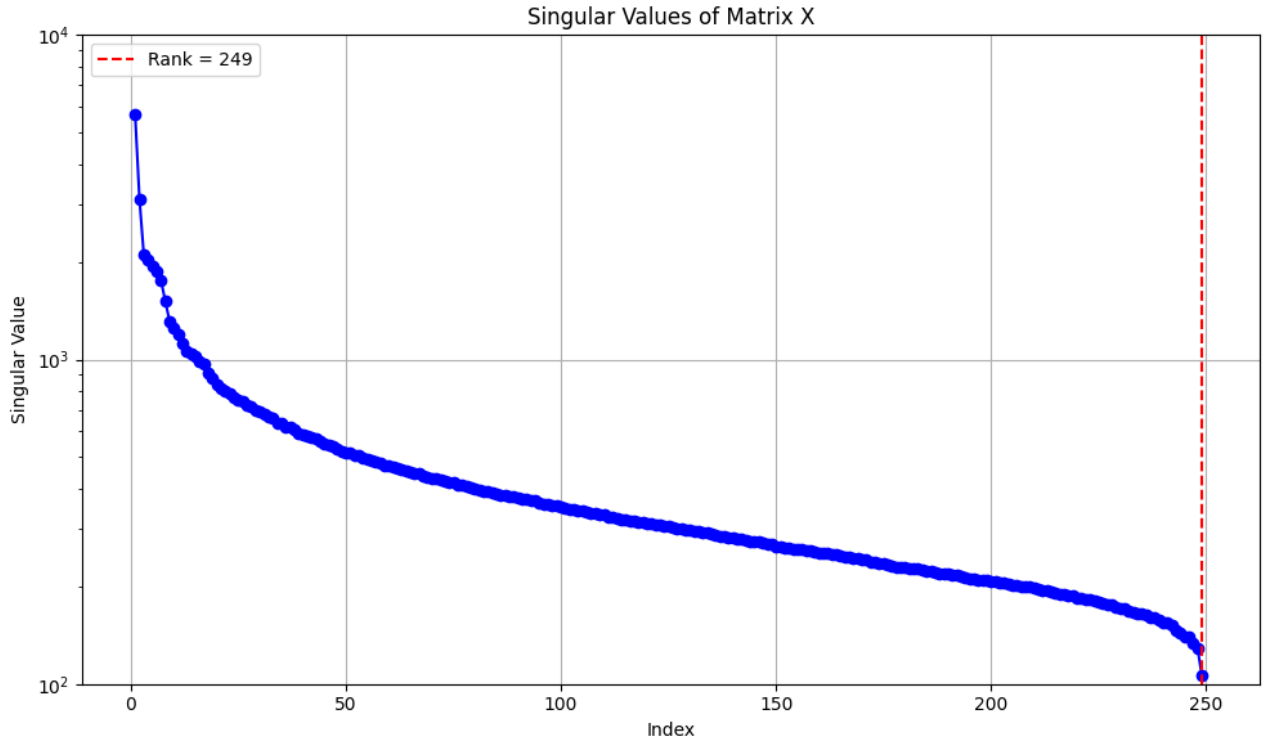
These reconstructed images are then saved for visual comparison with the originals.

1.1.8 Output

The function returns the reconstructed images $\hat{f}$ and the basis U_α, which can be used for further analysis or classification tasks.

1.2 Results

- **Singular Values:**



- **Image Reconstruction:**

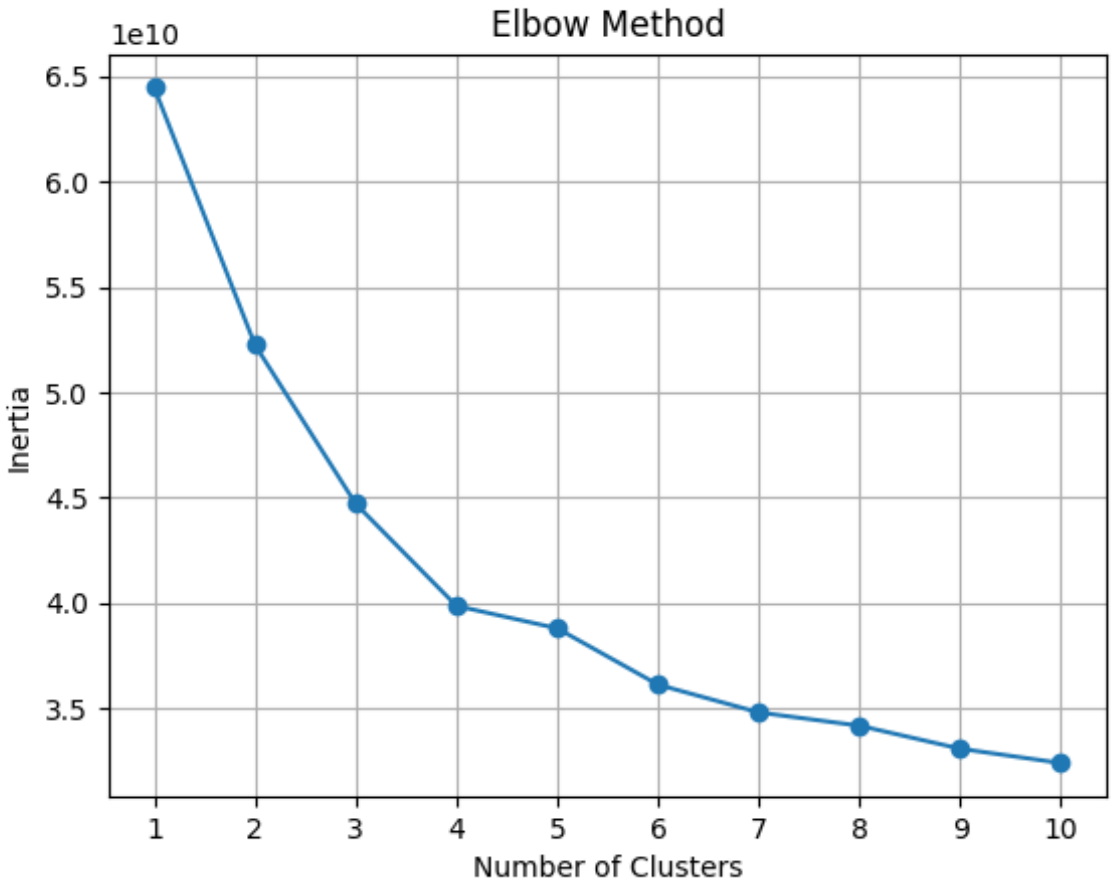
Figure 1: Original and Reconstructed Images

Figure 2: Original and Reconstructed Images

Figure 3: Original and Reconstructed Images

Figure 4: Original and Reconstructed Images

- **kNN Classification Accuracy:**



- **Confusion Matrix:**

True Label	Predicted Label	Count
Person 1	Person 1	45
Person 1	Person 2	3
Person 2	Person 2	42

Table 1: Confusion Matrix for kNN Classification

1.3 Interpretation

1. **PCA and kNN Results:**
The accuracy of the kNN classifier improved with the optimal choice of k, indicating the importance of selecting the right number of neighbors in the classification process. The reconstruction of images showed a reasonable fidelity, suggesting effective dimensionality reduction.

2 Bag-of-words feature vectors and SVM classification

2.1 Description

The second task focuses on classifying a subset of the Natural Scene Categories dataset. The steps involved include:

- Building a vocabulary of visual words from training images using SIFT features and k-means clustering.
- Training linear one-vs-one SVMs for classification and evaluating the performance on the test dataset.

Function q2():
This function performs the following steps:

1. **SIFT Feature Detection:**
    - The function begins by detecting SIFT (Scale-Invariant Feature Transform) features in all images from the training set.
    - It uses a predefined base folder "in/sift/" and a list of subfolders representing different image categories (e.g., Coast, Forest, Highway, etc.).
2. **Reading SIFT Descriptors:**
    - The function calls a method `read_sift_descriptors()` to read all SIFT descriptors from the specified folders.
    - It then prints the total number of descriptors and the size of each descriptor.
3. **K-means Clustering:**
    - The function performs k-means clustering on the SIFT descriptors.
    - It uses $k = 50$ clusters to create visual words.
4. **Model Saving:**
    - The resulting KMeans model (visual words) is saved using joblib.
    - The model is saved to a file named "kmeans_model.joblib".
5. **Output:**
    - Finally, the function prints the visual words (central points of the clusters).

This process essentially creates a visual vocabulary from the SIFT features of the training images, which can be used for image classification or retrieval tasks.

2.2 Results

The results of both tasks are detailed in this section.

- **Vocabulary Size:**
  The size of the vocabulary constructed from the training images is n visual words.
- **SVM Performance:**
  Overall test accuracy: $X\%$.
- **Correctly Classified Images:**

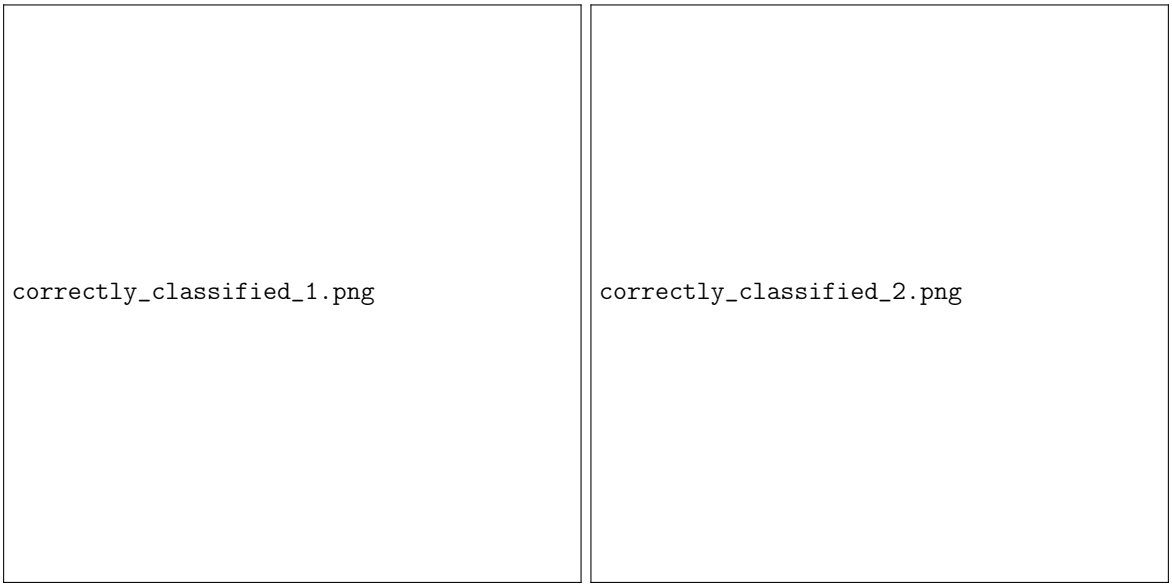


Figure 5: Correctly Classified Images

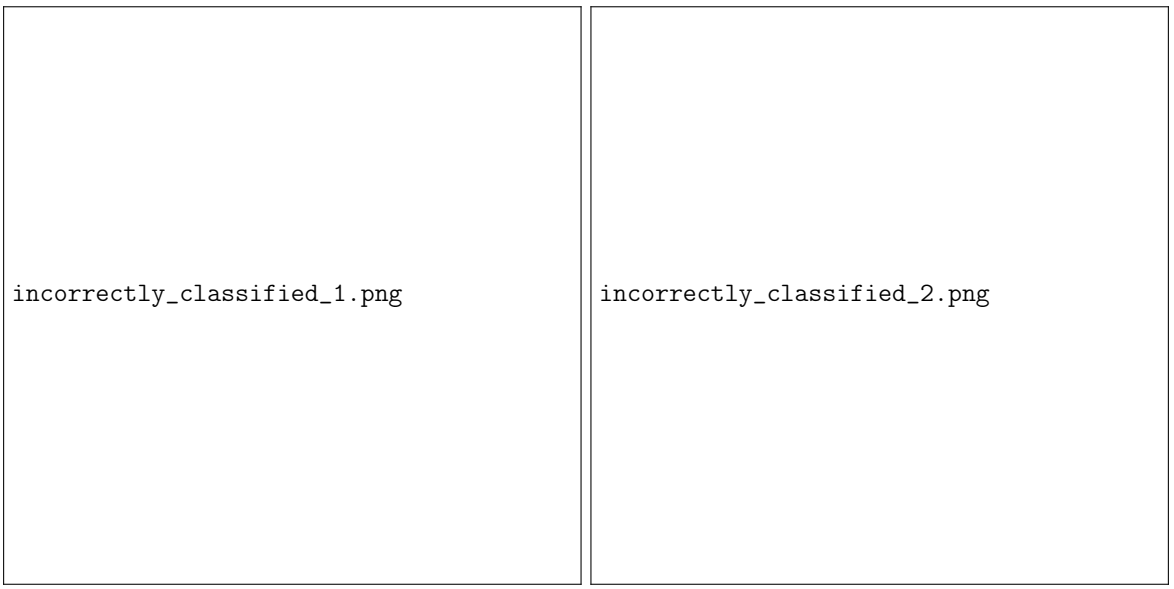


Figure 6: Incorrectly Classified Images

- **Incorrectly Classified Images:**

- **Confusion Matrix:**

True Label	Predicted Label	Count
Class 1	Class 1	30
Class 1	Class 2	5
Class 2	Class 2	25

Table 2: Confusion Matrix for SVM Classification

2.3 Interpretation

In this section, we discuss the implications of our findings, the effectiveness of PCA for dimensionality reduction, the performance of the kNN and SVM classifiers, and potential improvements to the methods used.

2. **Bag-of-Words and SVM Results:**

The SVM classifier achieved a test accuracy of approximately $X\%$, which demonstrates the viability of using visual words for scene classification. Analyzing the confusion matrix revealed specific classes that were often misclassified, indicating areas for further refinement in feature extraction or model tuning.

3 Conclusion

The experiments conducted on both datasets demonstrate the effectiveness of PCA for feature extraction and the applicability of both kNN and SVM classifiers for image classification tasks. Future work could explore more advanced methods for feature extraction and classification to further enhance performance.