Project 2a - Image Compression

Olha Morozova

Introduction

In this project, we compress and reconstruct grayscale image *Elaine* using PCA learned with the GHA algorithm. The images are divided into small blocks (8×8), flattened into vectors, and treated as input data.

The model does the following:

- Normalizes the input data to have zero mean and unit variance.
- Initializes a weight matrix with small random values.
- Trains the weights using GHA, updating them using a learning rate schedule.
- Stores the learned components and provides methods to encode (compress), decode (reconstruct), and get components.

1 Principal Component Analysis

In this part, we analyze how well GHA learns the principal components (PCs) of an image dataset. **Figure 1** shows the first 8 PCs reshaped into 8×8 blocks. We can see that the first few components capture smooth variations, while later ones contain finer texture patterns. This matches the idea that PCA puts the most important components (with the most variance) first.

1.1 Eigenvalues and Variance Explained

Figure 2 shows the eigenvalues of the correlation matrix. From the first few eigenvalues we see, that most of the data variance is concentrated in the first components. This

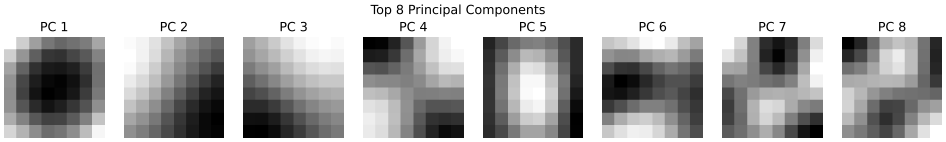

Figure 1: Top 8 principal components learned.

suggests using a relatively small number of PCs for compression.

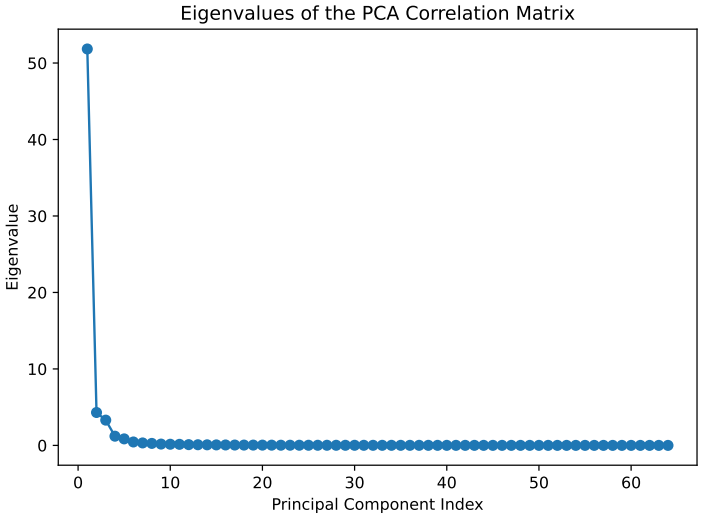

Figure 2: Eigenvalues of the PCA correlation matrix.

Figure 3 presents the cumulative variance explained. We can see that around 5 components are enough to capture more than 95% of the total variance.

1.2 Coefficient Analysis

In **Figure 4**, each image shows how strongly each block in the image responds to a certain component. The first PC captures rough texture, while higher PCs highlight finer patterns.

2 Image Reconstruction

After training the model on the image *Elaine*, we evaluated how well the model can reconstruct the original image using different numbers of components k, and how well they generalize to other similar images.

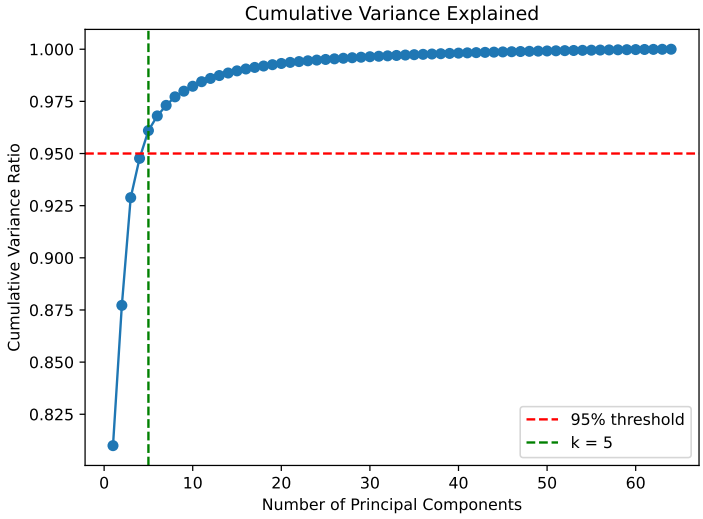

Figure 3: Cumulative variance explained. Red line is 95% threshold, green line shows optimal $k = 5$.

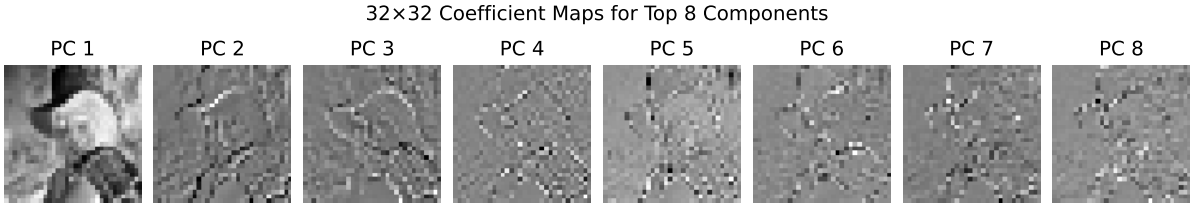

Figure 4: Coefficient maps for the top 8 PCs.

We reconstructed the original image using $k = 1, 2, 4, ..., 16$ components (**Figure 5**). As k increases, the reconstructed image becomes clearer and more similar to the original. When only a few components are used, the image lacks detail. Around $k = 8$ to $k = 16$, the image is already visually close to the original.

To better compare different reconstructions, **Figure 6** shows results for $k = 8$, $k = 16$, and $k = 32$. Visually, the difference between $k = 16$ and $k = 32$ is small, meaning that using $k = 16$ already preserves most of the information. This confirms what we saw in the cumulative variance plot (Figure 3).

We computed Mean Squared Error (MSE) for each k. **Figure 7** shows that the error drops quickly up to $k = 8$, and then stabilizes.

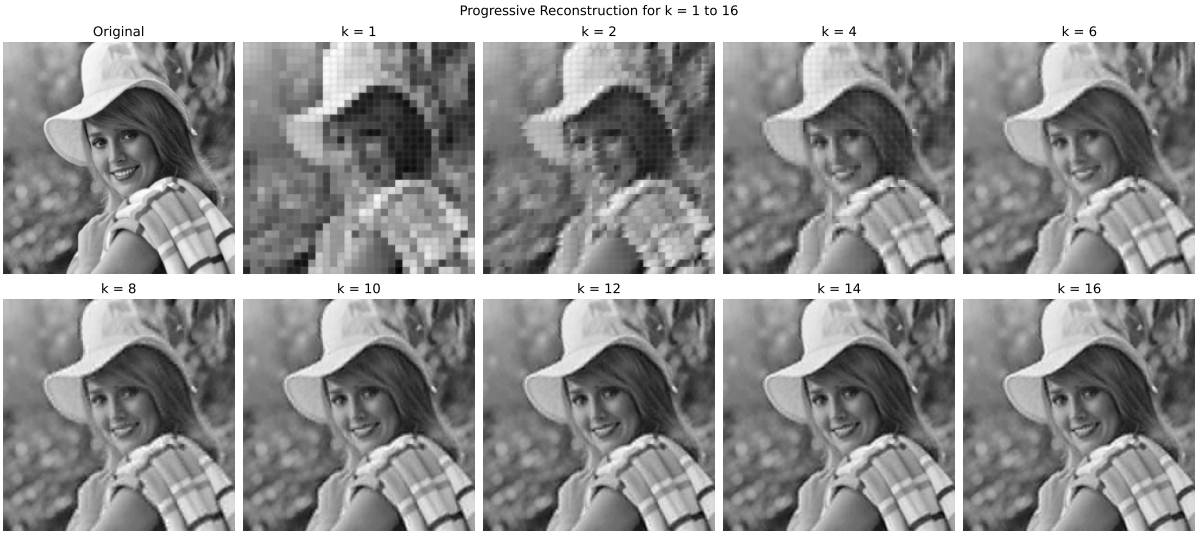

Figure 5: Progressive image reconstruction using $k = 1$ to 16 components.

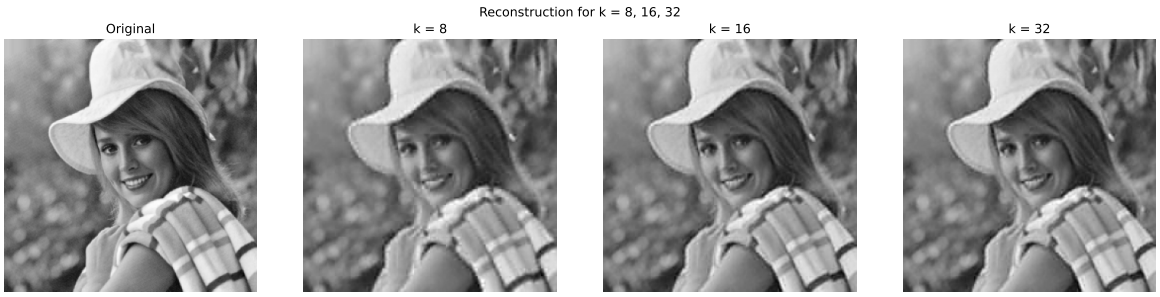

Figure 6: Reconstruction results for $k = 8$, 16, and 32.

2.1 Testing Generalization on New Images

To test generalization, we used two new grayscale portraits: *Lena* and *Woman2*. These images are visually similar to the training image, so we expect the same principal components to work for them. Both were resized and processed into blocks just like the training image. **Figure 8** shows the reconstruction results with $k = 16$. The reconstructions are visually good. The *Lena* image was reconstructed with $MSE = 44.94$, while the *Woman2* image had $MSE = 16.05$. These results show that the our model can generalize to new but visually similar portraits.

2.2 Component Similarity

We also measured cosine similarity between the components learned from *Elaine* and those from *Lena*. **Figure 9** shows the similarity matrix. The diagonal pattern suggests

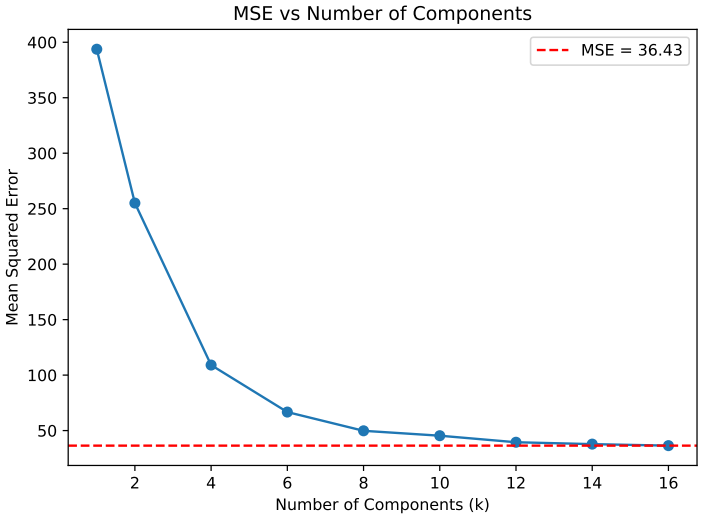

Figure 7: MSE between original and reconstructed image as a function of k.

Figure 8: Generalization test: original vs. reconstructed images using $k = 16$ PCs.

that the same types of features (edges, gradients) are captured in a similar order.

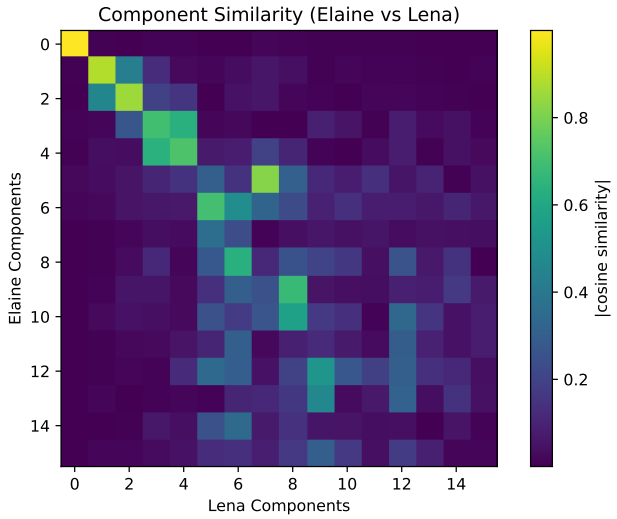

Figure 9: Cosine similarity between components from Elaine (rows) and Lena (columns). Brighter cells show more similar vectors.

3 Bonus: Linear Autoencoders

Here we compare our PCA results from the GHA algorithm with two types of linear autoencoders: one using separate weights and one using shared weights.

Separate Weights Autoencoder uses two independent matrices: encoder matrix W_{enc} and decoder matrix W_{dec}. **Shared Weights Autoencoder** assumes the decoder is the transpose of the encoder: $\hat{\mathbf{x}} = W^T W \mathbf{x}$. This model is closer to PCA, where projection and reconstruction use the same PCs. Both models were trained using gradient descent to minimize the MSE.

3.1 Training

We trained both models on *Elaine* image. As seen in **Figure 10**, the shared weights model learns faster and reaches lower error.

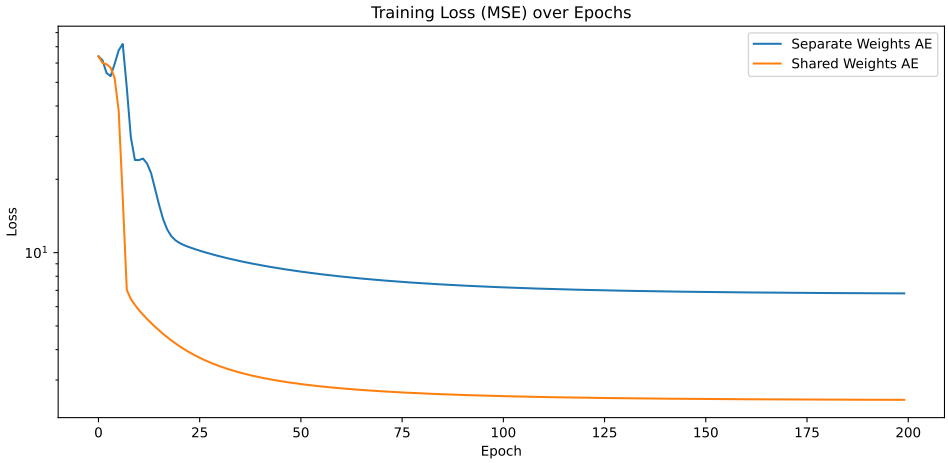

Figure 10: Training loss (MSE) of both models.

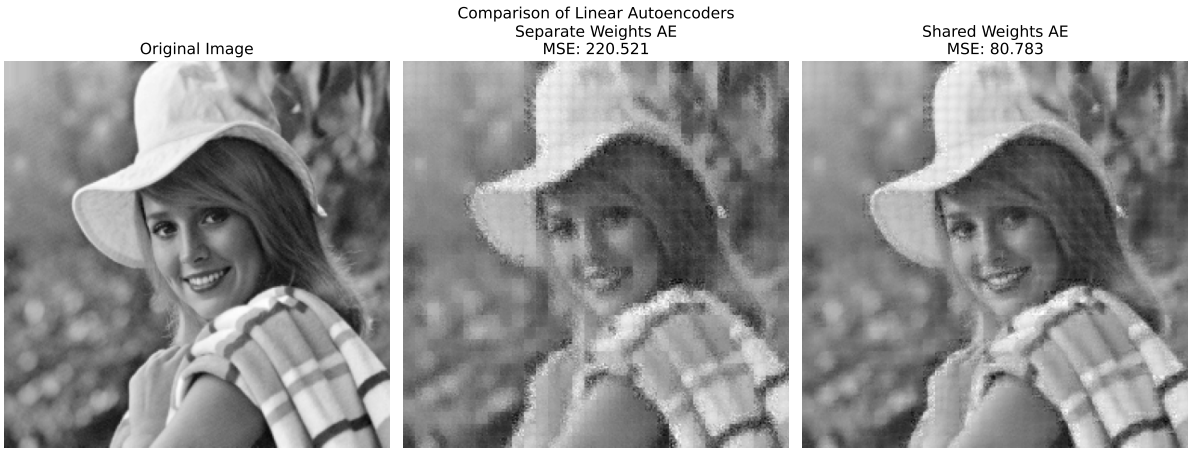

Figure 11: Reconstruction comparison on Elaine image.

3.2 Reconstruction

In **Figure 11**, we compare reconstructions using 32 hidden units. The separate weights model produces blurry output and has MSE over 220. The shared weights model gives clearer reconstruction with MSE around 80.

3.3 Generalization Test on New Images

We tested both models on two new grayscale portrait images, similar to previous testing. **Figure 12** shows, that the shared model again reconstructs both images better than the separate one. The MSE is lower for both test cases.

Figure 12: Generalization results on Lena and Woman2 images.

4 Summary

Our original PCA model worked well and produced good results. Testing on new images showed that the learned PCs could generalize to similar portraits. Small random weight initialization and a decreasing learning rate helped the model converge and stay stable.

The shared-weights autoencoder gave better results than the separate-weights version, which produced blurrier reconstructions. Compared to GHA-PCA, both autoencoders needed more components to reach similar quality.