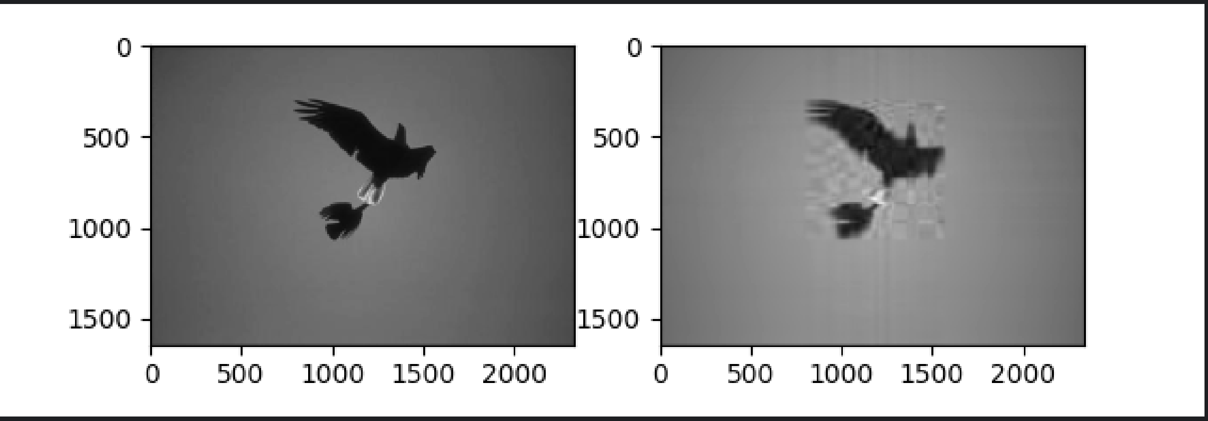
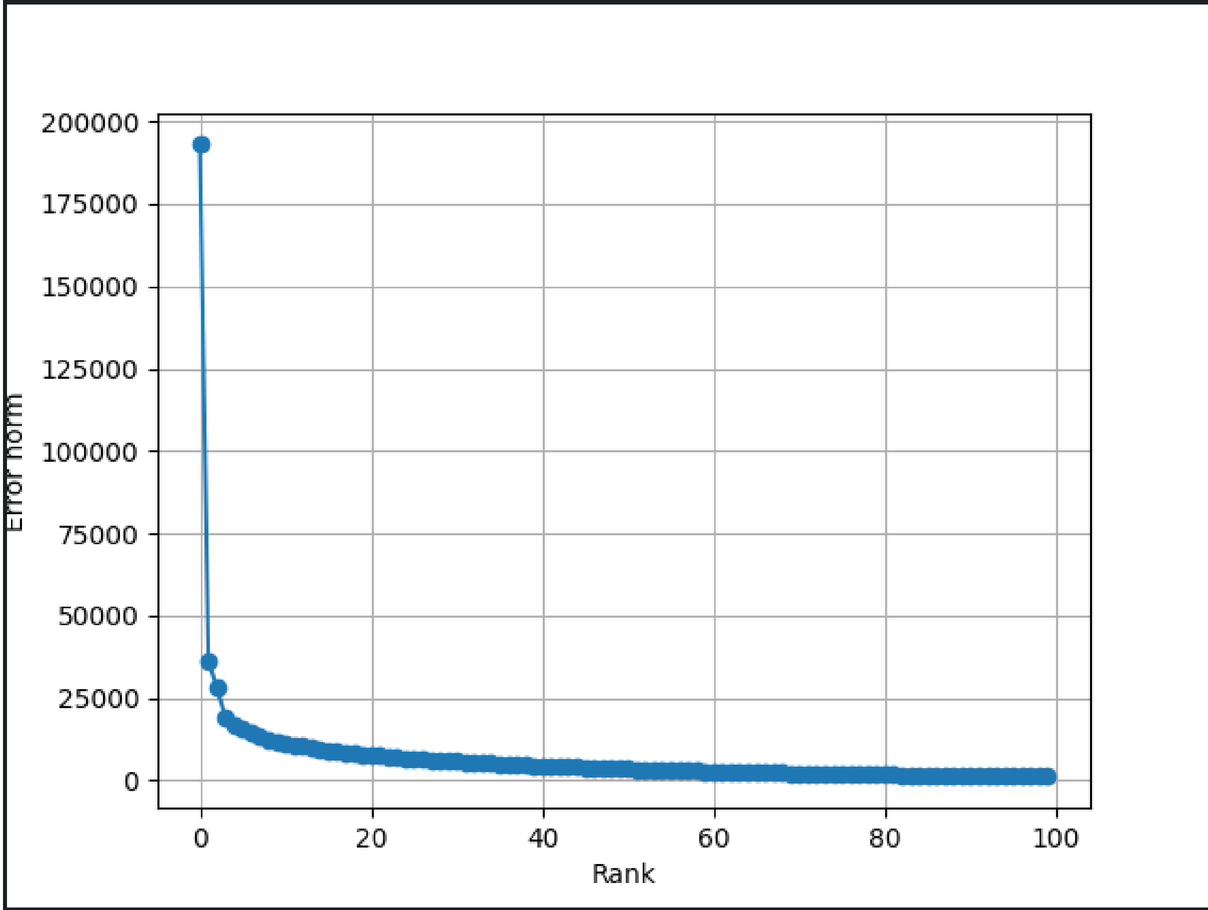
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

matrix = cv2.imread(r"C:\Users\alepa\Desktop\eagle.png", cv2.IMREAD_GRAYSCALE)

A = np.array(matrix)

# I compute and store the SVD of A as matrices U, s and V transpose,
# and pick the 10 biggest singular values through list manipulation
U, Σ, Vt = np.linalg.svd(A, full_matrices=False)

largest_sigma = np.argsort(Σ)[-10:]

# the definition of the interpolation matrix (given rank-10) is the following
P = np.dot(U[:, :10], np.dot(np.diag(Σ[:10]), Vt[:10, :]))


# we keep only the first 10 singular values to make a rank-10 approximation
U_approx = U[:, :10]
Σ_approx = np.diag(Σ[:10])
Vt_approx = Vt[:10, :]

# and so the resulting matrix is the dot product
A_approx = np.dot(U_approx, np.dot(Σ_approx, Vt_approx))

# plotting
plt.subplot(1, 2, 1)
plt.imshow(A, cmap='gray')

plt.subplot(1, 2, 2)
plt.imshow(A_approx, cmap='gray')

plt.show()
```




```python
# store errors in a list, then loop over the matrices for plotting
errors = []

for k in range(0, 100): # loop over the first 100 columns
    U_k = U[:, :k] # truncated U
    Σ_k = np.diag(Σ[:k]) # truncated Σ
    Vt_k = Vt[:k, :] # truncated V transpose
    A_k = np.dot(U_k, np.dot(Σ_k, Vt_k))
    e = np.linalg.norm(A - A_k, 'fro')
    errors.append(e)



plt.plot(range(0, 100), errors, marker='o')
```

```python
plt.xlabel('Rank')
plt.ylabel('Error norm')
plt.grid(True)
plt.show()
```



If we can approximate an image (matrix) with a rank much lower than its initial dimension, then the matrix is low rank. The plot shows that after about rank 10 the error curve stops improving significantly and flattens out almost completely at around rank 50. The intuition is that adding more singular values beyond this point does not improve much our approximation.