MAT4110 – Oblig 2

Jonas Gahr Sturtzel Lunde (`jonassl`)

October 26, 2019

We know from lecture that any $m \times n$ matrix A can be factorized as

$$A = UDV^T$$

where

- U is a $m \times m$ unitary matrix. If A is real, so is D, making it orthogonal.
- D is a $m \times n$ diagonal matrix, containing the singular values of A.
- V is a $n \times n$ unitary matrix. If A is real, so is V, making it orthogonal.

We can use this decomposition to compress the matrix A. The singular values on the diagonal of D is conventionally stored in order, from high to low. This means that the first singular values contribute more to the contents of A than the later ones do. Picking some $r \leq (n, m)$, we introduce the $m \times r$ matrix U', containing the upper parts of U, the $r \times r$ matrix D', containing the upper left parts of D, and the $r \times n$ matrix V', containing the left parts of V. We then have the compressed matrix

$$A' = U'D'T'^T$$

The quality of the compression will depend on how quickly the singular values decay, and what r we choose.

Storing A required storing $n \cdot m$ values, while instead of storing A', we can store the values of U', D', and T', which mean storing $m \cdot r + r + r \cdot n$. We introduce the *compression ratio* of A, being

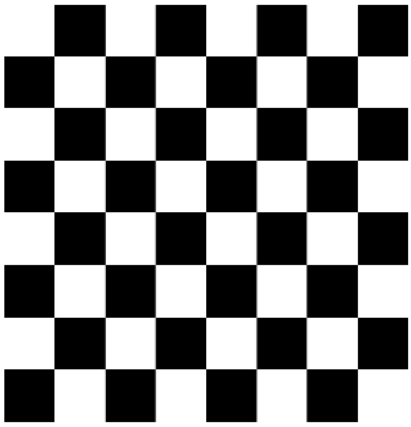
$$\text{compression ratio} = \frac{\text{uncompressed size}}{\text{compressed size}} = \frac{n \cdot m}{m \cdot r + r \cdot n + r}$$

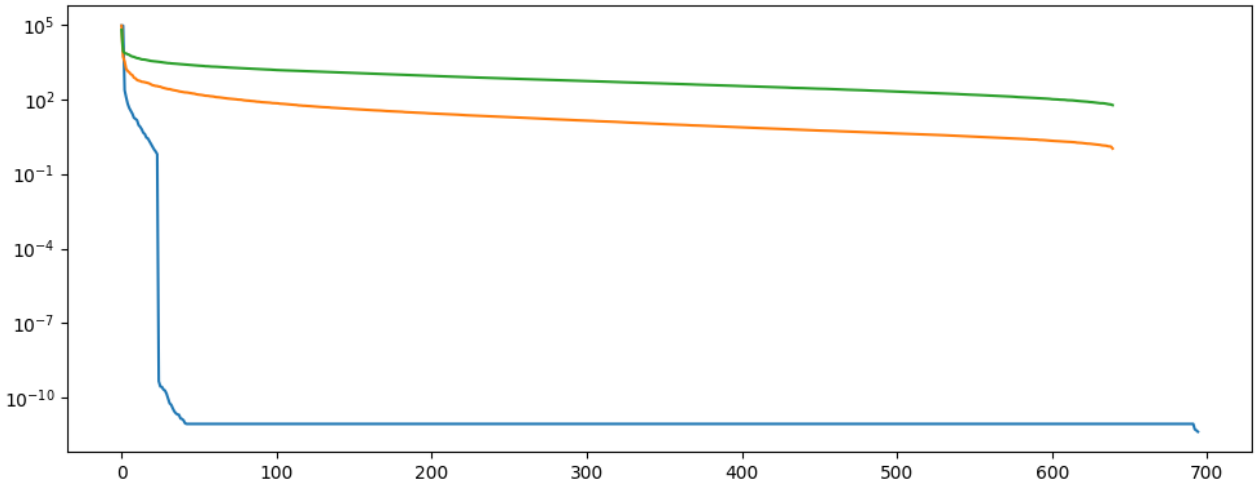
Most images are, more or less square, so this would approximately equal

$$\text{compression ratio} =\approx \frac{n \cdot n}{n \cdot r + r \cdot n + r} \approx \frac{n}{2r}$$

where we've also assumed that $r \ll 2n \cdot r$.

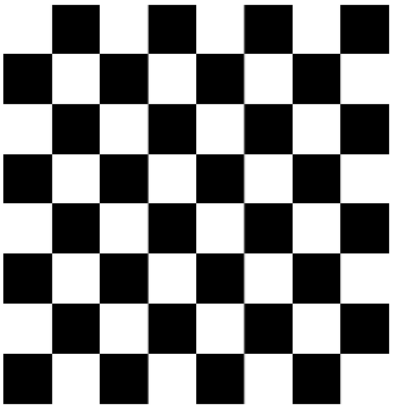
Consider now the three images below, respectively of sizes 695×720, 960×640, and 960×640. Their singular values, in order, are plotted below the pictures, in order blue, orange, green. We clearly see that the chessboard have very few singular values with much information, while the jellyfish and the new york picture contains a lot more information. The new york picture even more so than the jellyfish.





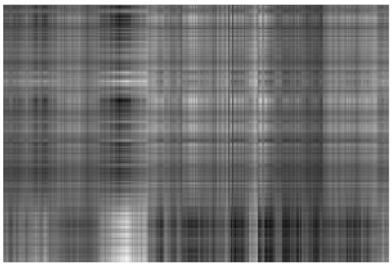
Below, the compressed versions of the images are plotted, for different number of singular values, r, and with calculated compression ratios. The chessboard looks indistiguishable already from $r = 2$, and we can compress the image an impressive factor of $50'000$ without losing information. The jellyfish starts looking good somewhere in the range $r = 16 - 32$, where we have a compression ratio somewhere around $300 - 1150$. For the new york image, we need around $r = 64 - 128$ singular values for a good looking image, giving a compression ratio of only $20 - 80$.

r = 2, Comp Ratio = 50040.0

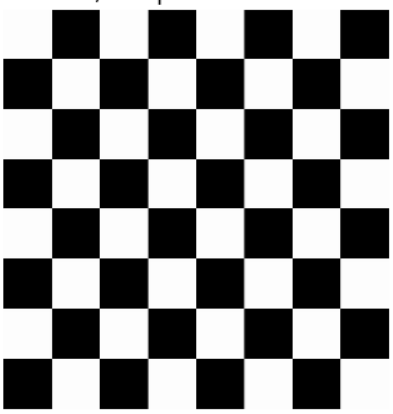


r = 2, Comp Ratio = 61440.0

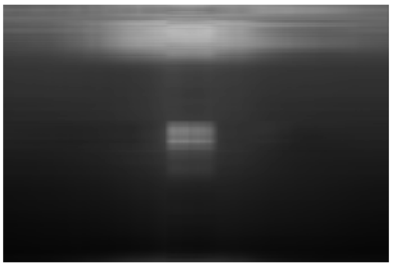
r = 2, Comp Ratio = 61440.0



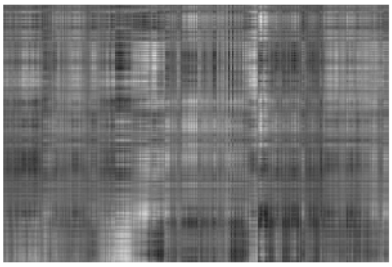
r = 4, Comp Ratio = 13900.0



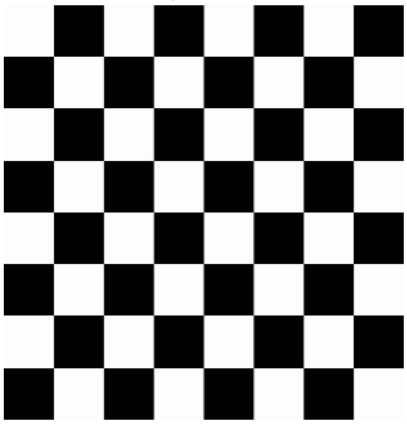
r = 4, Comp Ratio = 17066.7



r = 4, Comp Ratio = 17066.7

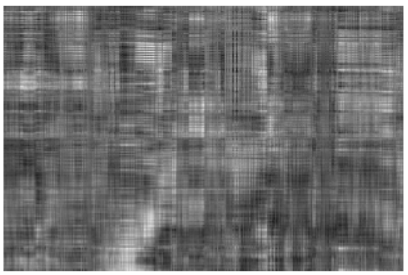


r = 8, Comp Ratio = 3679.4

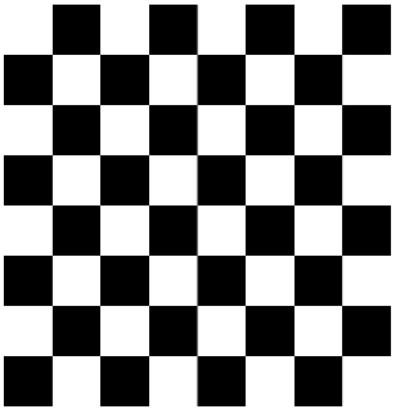


r = 8, Comp Ratio = 4517.6

r = 8, Comp Ratio = 4517.6

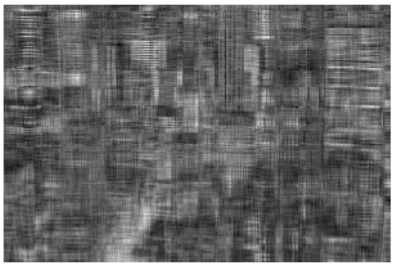


r = 16, Comp Ratio = 947.7

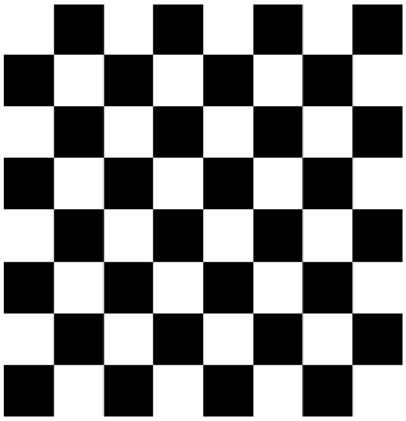


r = 16, Comp Ratio = 1163.6

r = 16, Comp Ratio = 1163.6

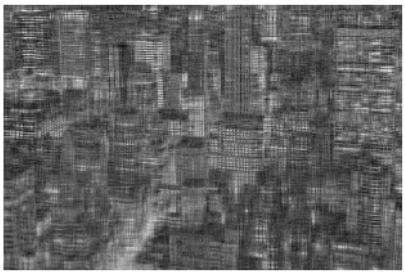


r = 32, Comp Ratio = 240.6

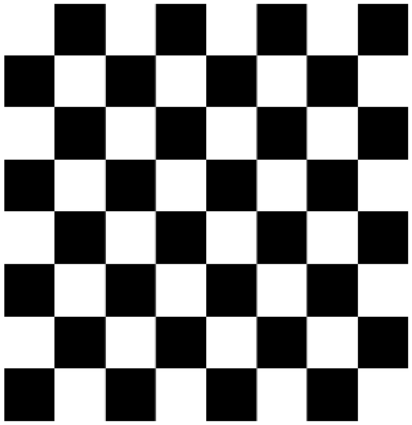


r = 32, Comp Ratio = 295.4

r = 32, Comp Ratio = 295.4



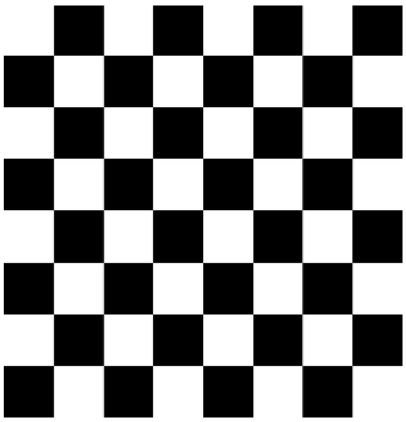
r = 64, Comp Ratio = 60.6



r = 64, Comp Ratio = 74.4

r = 64, Comp Ratio = 74.4

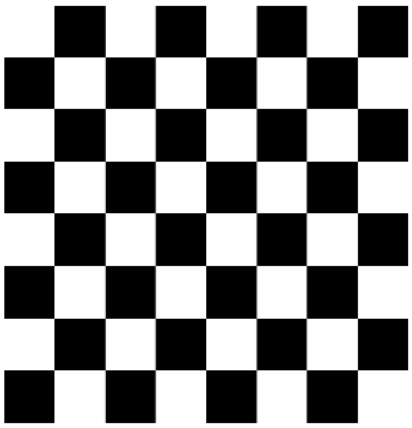
r = 128, Comp Ratio = 15.2



r = 128, Comp Ratio = 18.7

r = 128, Comp Ratio = 18.7

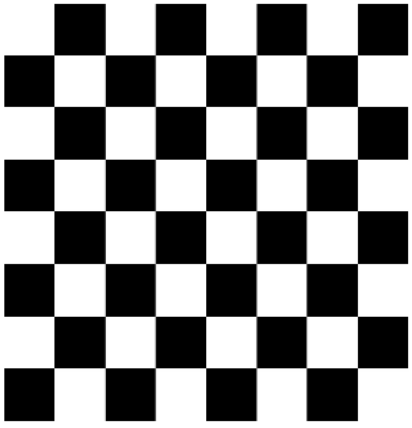
r = 256, Comp Ratio = 3.8



r = 256, Comp Ratio = 4.7

r = 256, Comp Ratio = 4.7

r = 512, Comp Ratio = 1.0



r = 512, Comp Ratio = 1.2

r = 512, Comp Ratio = 1.2

Appendix: Code

Listing 1: oblig2.py

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

fig, ax = plt.subplots(1, 3, figsize=(10, 4))
singular_values = []
for i, img_name in enumerate(["../board.png", "../jellyfish.jpg", "../new-york.jpg"]):
    img = Image.open(f"{img_name:s}").convert("L")
    img = np.array(img)
    ax[i].imshow(img, cmap="gray")
    ax[i].axis("off")

    U, D, V = np.linalg.svd(img)
    singular_values.append(D)

plt.tight_layout()
plt.savefig(f"../figs/original_img.png", bbox_layout="tight")
plt.clf()



N_list = 2**np.arange(1, 10)
M = len(N_list)
comp_ratio = np.zeros((M,3))
for j in range(M):
    fig, ax = plt.subplots(1, 3, figsize=(10, 4))
    N = N_list[j]
    for i, img_name in enumerate(["../board.png", "../jellyfish.jpg", "../new-york.jpg"]):
        img = Image.open(f"{img_name:s}").convert("L")
        img = np.array(img)

        U, D, V = np.linalg.svd(img)
        D = np.diag(D)

        U2 = U[:, :N]
        D2 = D[:N, :N]
        V2 = V[:N, :]
        img2 = U2@D2@V2

        comp_ratio[j, i] = (img.shape[0]*img.shape[1])/(2*N**2 + N)

        ax[i].imshow(img2, cmap="gray")
        ax[i].axis("off")
        ax[i].set_title(f"r = {N}, Comp Ratio = {comp_ratio[j, i]:.1f}")
    plt.tight_layout()
    plt.savefig(f"../figs/img{j}.png", bbox_layout="tight")
    plt.clf()

for i in range(3):
    plt.semilogy(singular_values[i])
plt.tight_layout()
plt.savefig(f"../figs/singular_values.png", bbox_layout="tight")

plt.plot()
```