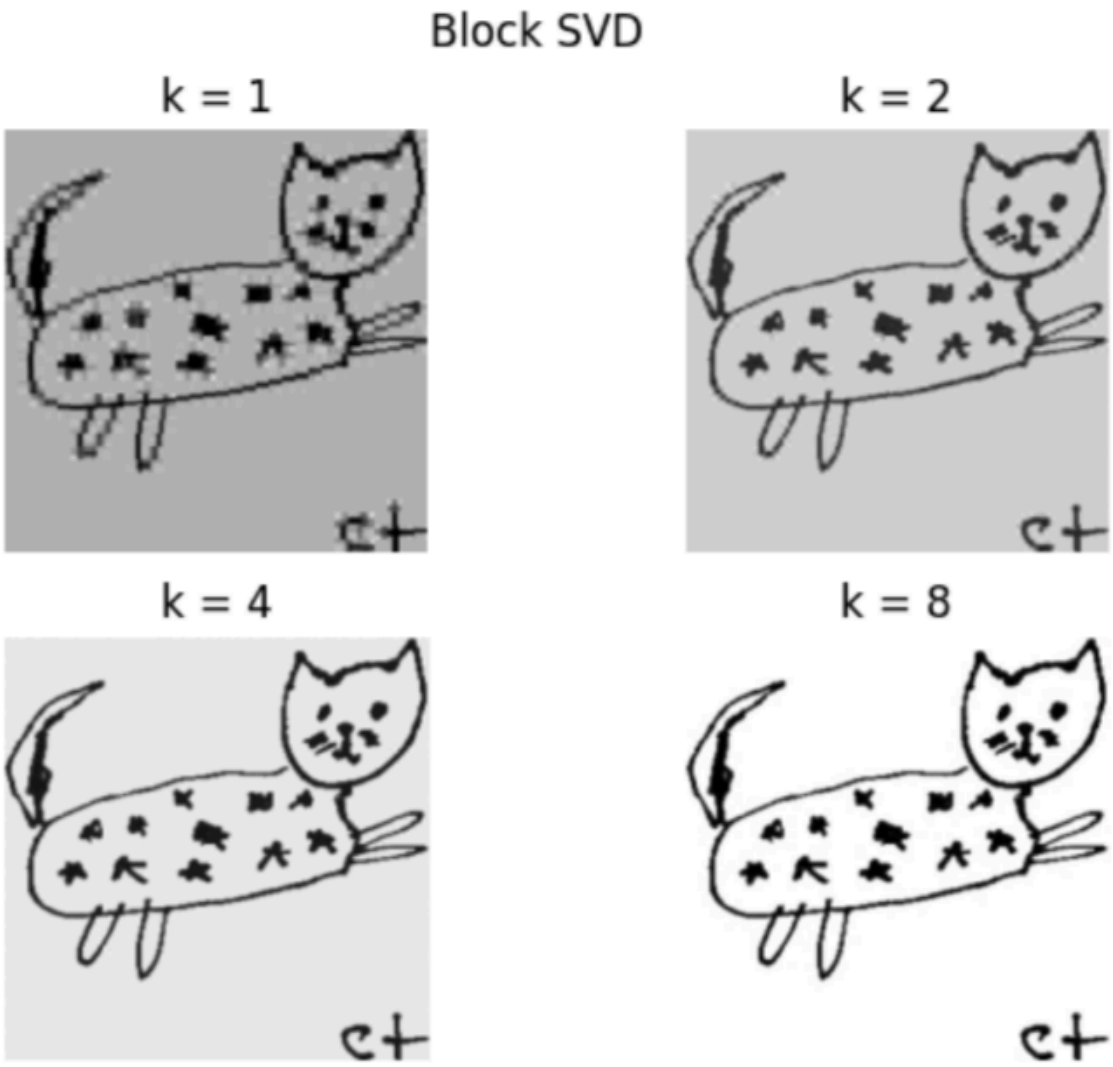
Koris Lucienne Haq
4/15/2025
MATH-485
Bo Shen

Assignment 5: SVD

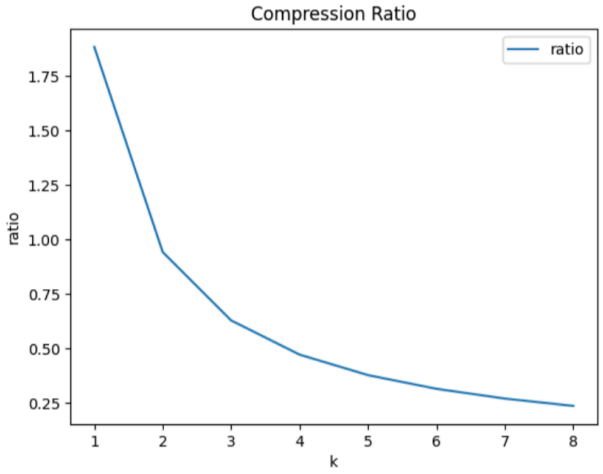
To tackle compression via block-wise SVD, I needed to write a number of functions that would simplify the process of repeated tests on an image. First I wrote a decompose function that returned a list of 8x8 blocks of values from the original image. Next I needed a function that could stitch them back together which simply took the list of 8x8 blocks and converted them back into a numpy array. Next, I needed a function that I could pass a block into and perform SVD on it. Lastly I wrote a macro to easily utilize the aforementioned functions.

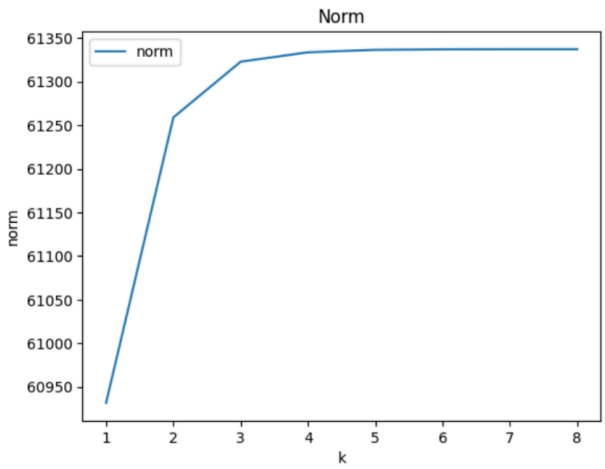
```
def decompose_image(A: np.array):
    blocks = []
    for i in range(0, 32):
        blocks.append([])
        for j in range(0, 32):
            (x, y) = (i, j)
            (x, y) = (x * 8, y * 8)
            blocks[i].append(A[x:x+8, y:y+8])
    return blocks

def combine_blocks(blocks) -> np.array:
    combined = np.zeros((256, 256))
    for i in range(0, 32):
        for j in range(0, 32):
            combined[i*8:i*8+8, j*8:j*8+8] = blocks[i][j]
    return combined

def compress_block(block, k: int) -> np.array:
    U, S, VT = np.linalg.svd(block, full_matrices=False)
    block_k = U[:, :k] @ np.diag(S[:k]) @ VT[:k, :]
    return block_k

def process_image(A: np.array, k: int) -> np.array:
    blocks = decompose_image(A)
    for i in range(0, 32):
        for j in range(0, 32):
            blocks[i][j] = compress_block(blocks[i][j], k)
    combined = combine_blocks(blocks)
    return combined
```

Next, I wanted to compare the results of block-wise SVD for different values of k, so I plotted k = 1, 2, 4, 8 as such:

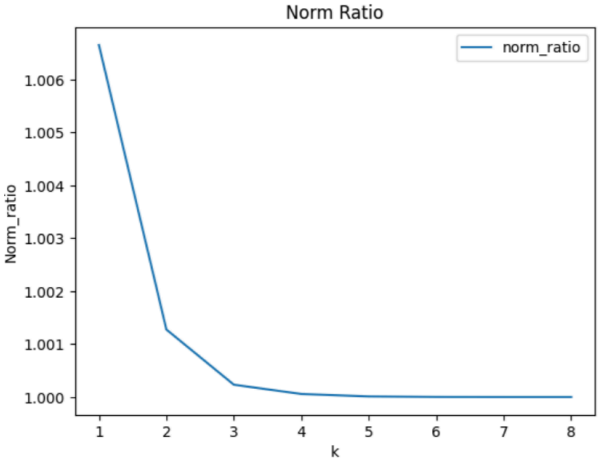


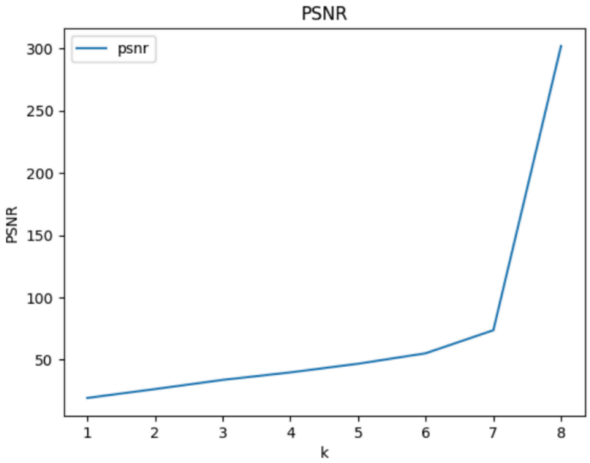
One thing to note is that the background white-level is darker for lower values of k, show less accurate grayscale values. The image for k = 1 is also very choppy and blocky. As k increases, the sharpness of the image as well as the gray-scale accuracy improve. The image at k = 8 appears identical to the original image. To better understand these results, I plotted a variety of metrics for the image as the value of k increase (1, 2, 5, … 8).





I started with the compression ratio and the frobenius norm of the new image. The compression ratio (32 / (k * (8 + 8 + 1)) gets smaller as the value of k increases for the amount of data in reconstruction, which corresponds to the image increasing in quality with more data. The frobenius norm shows that the amount of quality improvements between values of k falls off and stops notably increasing (which makes sense as there is only a limited amount of visual information to restore). Next, I plotted norm ratio and peak to signal noise ratio.





For the norm ratio in particular, I was comparing the frobenius norm of the original image to the new image. We can see that as the images get closer in appearance, the norm ratio approaches 1. We see an inverse trend in PSNR where the PSNR value gets much larger as the k value increases, further reflecting how a higher value of k leads to an image result much closer to the original.

Resources

- https://numpy.org/devdocs/reference/generated/numpy.linalg.norm.html
- https://www.mathworks.com/help/vision/ref/psnr.html