MAT185 – Linear Algebra

Assignment 2

Instructions:

Please read the **MAT185 Assignment Policies & FAQ** document for details on submission policies, collaboration rules and academic integrity, and general instructions.

1. **Submissions are only accepted by** Gradescope. Do not send anything by email. Late submissions are not accepted under any circumstance. Remember you can resubmit anytime before the deadline.
2. **Submit solutions using only this template pdf**. Your submission should be a single pdf with your full written solutions for each question. If your solution is not written using this template pdf (scanned print or digital) then your submission will not be assessed. Organize your work neatly in the space provided. Do not submit rough work.
3. **Show your work and justify your steps** on every question but do not include extraneous information. Put your final answer in the box provided, if necessary. We recommend you write draft solutions on separate pages and afterwards write your polished solutions here on this template.
4. **You must fill out and sign the academic integrity statement below**; otherwise, you will receive zero for this assignment.

Academic Integrity Statement:

Full Name: CHEUNG, Hei Shing (Hayson) ____

Student number: 1010907823 ____

Full Name: SIU, Nelson ____

Student number: 1010940608 ____

I confirm that:

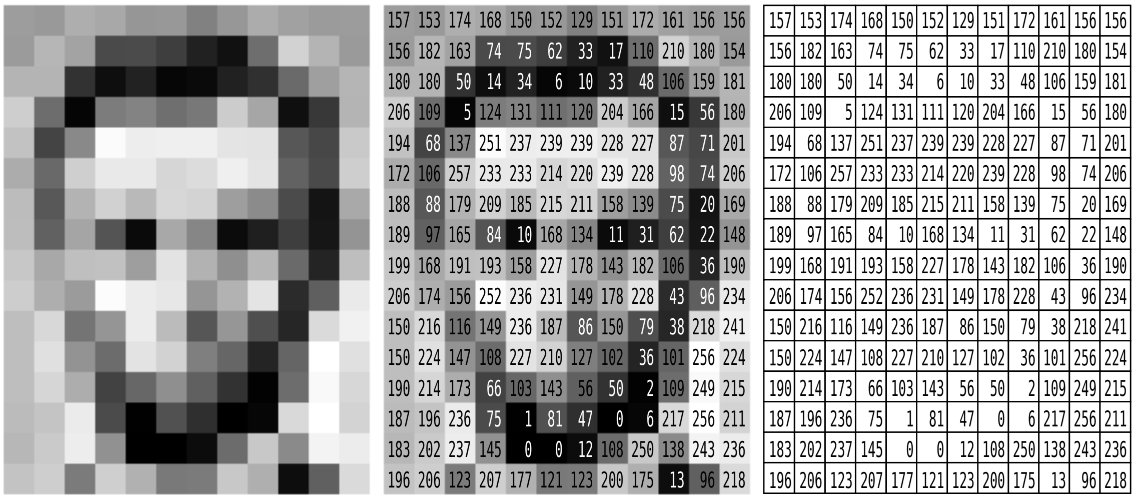
- I have read and followed the policies described in the document **MAT185 Assignment Policies & FAQ**.
- In particular, I have read and understand the rules for collaboration, and permitted resources on assignments as described in subsection II of the the aforementioned document. I have not violated these rules while completing and writing this assignment.
- I have not used generative AI in writing this assignment.
- I understand the consequences of violating the University's academic integrity policies as outlined in the Code of Behaviour on Academic Matters. I have not violated them while completing and writing this assignment.

By submitting this assignment to Gradescope, I agree that the statements above are true.

Preamble:

Image processing is the manipulation of digital images by applying mathematical tools and algorithms. A wide range of applications based on digital image processing are, for example, medical imaging, image optimization in consumer cameras, computer vision, and satellite imagery. Linear algebra, and the techniques you will learn during this course, plays a crucial role in that field by providing a mathematical foundation.

A typical digital image can be considered as a 2-dimensional matrix of *pixels* (abbreviation for *picture element*). Methods of digital image processing are manipulating this 2D-matrix. A pixel is the smallest element of a digitally acquired raster image, and can be considered as a colour sample at each point of an image. Typically, each pixel is represented by 3 positive integer values from 0 to 255 for each colour red, green, and blue: 0 is representing black and 255 ($= 2^8 - 1$) either red, green, or blue. In that case, 24 Bits are used to code 16,777,216 distinct colours for each pixel. This is called a 24 bpp (24 Bits-per-pixel) colour depth. If a grey-scale image is sampled, each pixel samples the light intensity. In that case, only 8 Bits (single integers from 0 to 255) are typically necessary, as shown in Figure 1, to store grey-scale images.



157	153	174	168	150	152	129	151	172	161	156	156
156	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	257	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
206	174	156	252	236	231	149	178	228	43	96	234
150	216	116	149	236	187	86	150	79	38	218	241
150	224	147	108	227	210	127	102	36	101	256	224
190	214	173	66	103	143	56	50	2	109	249	215
187	196	236	75	1	81	47	0	6	217	256	211
183	202	237	145	0	0	12	108	250	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Figure 1: Grey-scale image represented by 8 bpp (Bits-per-pixel). [*source: stanford.edu*]

One application of linear algebra in the context of digital image processing is for a linear transformation of the images / 2D-matrices. We will discuss linear transformations in more detail later this term. Common transformations include scaling, rotation, and translation of the image. All of these linear transformations can be represented by a matrix multiplication.

Another important application of linear algebra in the context of digital image processing is filtering, which will be explored in Question 1 of this assignment. Filtering includes, for example, methods for noise reduction, blurring/sharpening, edge detection, white balancing, colour correction, and many more.

Lastly, images have to be stored in efficient ways. For a colour image of size $n \times m$ with a 24 bpp colour depth, $m \times n \times 24$ Bits of memory are necessary. An image of 8000×6000 Pixels (4:3 aspect ratio and 48MP resolution of modern smartphone cameras) with 24 Bit colour depth will lead to an uncompressed image filesize of 144 MB. To reduce the image filesize, lossy (permanently removing *unnecessary* information of the original image) and lossless compression methods are applied. For lossy compression algorithms, it is vital to identify unnecessary information of the image, which are not perceived by the viewer and can be removed. One approach for a lossy compression is based on the Singular Value Decomposition (SVD), which will be explored in more detail in Question 2.

Question 1:

For simplicity, we assume that each pixel of a grey-scale image is considered as a real value in Question 1(a) to (e). Let $G \in {}^n\mathbb{R}^m$ be a grey-scale image of $n \times m$ pixels. A collection of image filters $F_1, F_2, \ldots, F_k \in {}^n\mathbb{R}^m$ can be used to process this image, resulting in filtered images A_l represented as:

$$A_l = F_l \circ G, \quad l = 1, 2, \ldots, k, \tag{1}$$

where $\circ$ denotes the entry-wise product. The entry-wise product (also called *Hadamard* product) of two matrices of the same size is defined as the product, where each entry of the resulting matrix is the product of the corresponding entries of the original matrices. For example, if $P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$ and $Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}$ are 2×2 matrices, the Hadamard product is defined as

$$P \circ Q = \begin{bmatrix} p_{11}q_{11} & p_{12}q_{12} \\ p_{21}q_{21} & p_{22}q_{22} \end{bmatrix}. \tag{2}$$

(a) Can the filter F_1 in Equation (1) be used to blur or smooth an image A_1? Unsupported answers will not receive full credit.

YES. Define *blurring* as an operation so that the average gradient between pixels decreases. Apply a blurring algorithm to the image $G = \{g_{ij}\}$ to get the filter $F_1 = \{f_{ij}\}$. One naive way to do this is by averaging the each pixel values in a 3×3 window, and devide by the pixel value to get the filter F_1', hence $F_1' \circ G$ is *blurred.* You specified for the same filter F_1 to blur the filterd image $A_1 = F_1 \circ G$ when applied as a Hadamard Product. We can modify the same said averaging filter to blur both A_1 and G. We can do it by taking the root of the filter values, which allows us to gradually reuse the same filter as Haddamard product: $A_2 = F_1 \circ A_1 = F_1 \circ F_1 \circ G$. Further explanation and results: https://github.com/HaysonC/svdAssignment

$$f_{ij} = \sqrt[3]{\frac{1}{9g_{ij}} \sum_{0 \le s,t \le 2} g_{i+s-1,j+t-1}} = \sqrt[3]{\frac{\text{conv}(\{1/9\}_{3\times3}, G)_{ij}}{g_{ij}}}$$

(b) Name at least **three** applications in the context of digital image processing for a filter as defined in Equation (1). Additionally, give details how F_i would look like for your applications.

1. **Adusting white balance/color temperature** The filter F_i can be used to adjust the white balance of an image. By setting a filter to enhance the red or blue channel, we can adjust the color temperature of the image by mulitplying the filter to the specific channel. We assume that a colored image is defined as $\{R, G, B\}$, then $\{F_i \circ R, G, B\}$ will be the new image with adjusted white balance. In this case F_i would be a uniform matrix with entries greater than one if we want to enhance the color, and less than one if we want to decrease the color.
2. **Cinematic Effects** Certain filters can be used to create cinematic effects. For example, a filter that dims the edges of image (vignette) can be applied by multiplying the filter to the image. The filter will have a ones in the center and approaches zero as it goes to the edges.
3. **Artifcialy Increasing Exposure** A filter can be used to artificially increase the exposure of an image. By multiplying the filter to the image, the pixel values will be increased by a certain factor. The filter will have values greater than one, otherwise, we can use the filter to decrease the exposure. The increased exposure image will be $\{F_i \circ G\}$, and it would look like a brighter version of the original image.

(c) Assume that the entries of the image G are all nonzero. Prove that the set of filtered images $\{F_1 \circ G, F_2 \circ G, \ldots, F_k \circ G\}$ is linearly independent if and only if the set of filters $\{F_1, F_2, \ldots, F_k\}$ is linearly independent.

("$\Rightarrow$") Assume, for the sake of contrapositive, that the set of filters $\{F_1, F_2, \ldots, F_k\}$ is linearly dependent. Then, we proof that exists a non-trivial linear combination of the filtered images that equals the zero vector. We first consider the linear combination of the filtered images, by the Distributive Property, as:

$$\lambda_1(F_1 \circ G) + \lambda_2(F_2 \circ G) + \cdots + \lambda_k(F_k \circ G)$$

Since multiplicati on in the Hadamard product is commutative, we can consider $\lambda_j(F_j \circ G) = G \circ (\lambda_j F_j)$. We can then rewrite the linear combination as:

$$G \circ (\lambda_1 F_1 + \lambda_2 F_2 + \cdots + \lambda_k F_k)$$

Since there exist a non-trivial linear combination of the filters that equals the zero vector, we take $\mu_1, \mu_2, \ldots, \mu_k \neq \mathbf{0}_k$ such that $\mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k = 0$. We coule then pick $\lambda_j = \mu_j$ and rewrite the linear combination as:

$$G \circ (\mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k) = G \circ \mathbf{0}$$

Thus, since $G \neq \mathbf{0}$, there also exist a non-trivial linear combination of the filtered images that equals the zero vector, which implies that the set of filtered images is linearly dependent.

("$\Leftarrow$") Assume, for the sake of contrapositive, that the set of filtered images is linearly dependent. Then, we proof that there exist a non-trivial linear combination of the filters that equals the zero vector. We first consider the linear combination of the filters:

$$\mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k$$

Since the images are of any real matrix of a specific size, we can take the Hadamard product of the linear combination with the image G, again, by the Distributive Property, we have:

$$G \circ (\mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k) = G \circ \mathbf{0} = \mathbf{0}$$

Again, by communativity of the Hadamard product, we can rewrite the linear combination as:

$$\mu_1(F_1 \circ G) + \mu_2(F_2 \circ G) + \cdots + \mu_k(F_k \circ G) = \mathbf{0}$$

Since $\{\mu_1, \mu_2, \ldots, \mu_k\} \neq \mathbf{0}_k$, there exist a non-trivial linear combination of the filtered images that equals the zero vector, which implies that the set of filters is linearly dependent.

(d) Assume that the entries of the image G are all nonzero. Let $\mathcal{W}$ be the set of the filtered images $\{F_1 \circ G, F_2 \circ G, \ldots, F_k \circ G\}$. Suppose a new image filter F_{k+1} is introduced. Prove that the filtered image $F_{k+1} \circ G$ lies in $\operatorname{span} \mathcal{W}$ if and only if F_{k+1} is a linear combination of $\{F_1, F_2, \ldots, F_k\}$.

("$\Rightarrow$") Assume that $F_{k+1} \circ G$ lies in $\operatorname{span} \mathcal{W}$. Then, we proof that F_{k+1} is a linear combination of $\{F_1, F_2, \ldots, F_k\}$. Since $F_{k+1} \circ G$ lies in $\operatorname{span} \mathcal{W}$, we can write $F_{k+1} \circ G$ as a linear combination of the filtered images:

$$F_{k+1} \circ G = \lambda_1(F_1 \circ G) + \lambda_2(F_2 \circ G) + \cdots + \lambda_k(F_k \circ G)$$

By the Distributive Property, we can rewrite the linear combination as:

$$F_{k+1} \circ G = (\lambda_1 F_1 + \lambda_2 F_2 + \cdots + \lambda_k F_k) \circ G$$

Since $G = \{g_{ij}\}$ is a non-zero matrix, we can multiply both sides by $G^{-1} = \{1/g_{ij}\}$ (The inverse such that $G \circ G^{-1} = \mathbf{1}$):

$$F_{k+1} \circ G \circ G^{-1} = (\lambda_1 F_1 + \lambda_2 F_2 + \cdots + \lambda_k F_k) \circ G \circ G^{-1}$$

Since $G \circ G^{-1} = \mathbf{1}$, we have:

$$F_{k+1} \circ \mathbf{1} = (\lambda_1 F_1 + \lambda_2 F_2 + \cdots + \lambda_k F_k) \circ \mathbf{1}$$

So we have:

$$F_{k+1} = \lambda_1 F_1 + \lambda_2 F_2 + \cdots + \lambda_k F_k$$

Thus, F_{k+1} is a linear combination of $\{F_1, F_2, \ldots, F_k\}$.

("$\Leftarrow$") Assume that F_{k+1} is a linear combination of $\{F_1, F_2, \ldots, F_k\}$. Then, we proof that $F_{k+1} \circ G$ lies in $\operatorname{span} \mathcal{W}$. Since F_{k+1} is a linear combination of $\{F_1, F_2, \ldots, F_k\}$, we can write F_{k+1} as:

$$F_{k+1} = \mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k$$

We can then multiply both sides by G:

$$F_{k+1} \circ G = (\mu_1 F_1 + \mu_2 F_2 + \cdots + \mu_k F_k) \circ G$$

By the Distributive Property, we can rewrite the linear combination as:

$$F_{k+1} \circ G = \mu_1(F_1 \circ G) + \mu_2(F_2 \circ G) + \cdots + \mu_k(F_k \circ G)$$

Thus, $F_{k+1} \circ G$ lies in $\operatorname{span} \mathcal{W}$.

(e) Assume $F_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $F_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and $F_3 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ are filters for a grey-scale image $G \in {}^2\mathbb{R}^2$. Verify whether the filters F_1, F_2, F_3 are linearly independent, and determine whether $F_4 = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$ lies in the span of $\{F_1, F_2, F_3\}$.

Consider the linear combination of the filters, with $a, b, c \in \mathbb{R}$:

$$aF_1 + bF_2 + cF_3 = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} + \begin{bmatrix} 0 & b \\ b & 0 \end{bmatrix} + \begin{bmatrix} c & c \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} a+c & b+c \\ b & a \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

So we have the system of equations:

$$S = \begin{cases} a + c = 0 \\ b + c = 0 \\ b = 0 \\ a = 0 \end{cases}$$

The system of equations has only the trivial solution $a = b = c = 0$, so the filters F_1, F_2, F_3 **are linearly independent**.

To determine whether F_4 lies in the span of $\{F_1, F_2, F_3\}$, we can write F_4 as a linear combination of $\{F_1, F_2, F_3\}$, we consider the system of equations:

$$S = \begin{cases} a + c = 2 \\ b + c = 1 \\ b = 1 \\ a = 4 \end{cases}$$

Substitute $b = 1$ and $a = 4$ into the first two equations, we have $c = -2$ and $c = 0$, which is a contradiction. Thus, F_4 **does not** lie in the span of $\{F_1, F_2, F_3\}$.

(f) For saving grey-scale images A of size $n \times m$ pixels in an efficient way, each pixel is stored as an 8 Bit integer, where 0 represents black and 255 represents white (see preamble). Let $\mathcal{V}$ be the set of these grey-scale images. Is $\mathcal{V}$ a vector space if one uses entry-wise vector addition and scalar multiplication?

NO, $\mathcal{V}$ is not a vector space. The set of grey-scale images $\mathcal{V}$ is not closed under addition. For example, we take two images A and B where they are both full white images, we have:

$$A = \{255\}_{n\times m}, B = \{255\}_{n\times m} \quad A + B = \{510\}_{n\times m} \notin \mathcal{V}$$

Question 2:

Let $A \in {}^m\mathbb{R}^n$ be a grey-scale image of size $m \times n$, where each entry is representing a grey pixel. As you learnt in ESC103, the rank of an $n \times m$ matrix is the number of linear independent rows, which is the same as the number of the linear independent columns.

The Singular Value Decomposition[1] of a matrix A is a way of writing A as the product of three matrices:

$$A = U\Sigma V^T \tag{3}$$

with the matrices U, Σ, and V^T defined as

$$A = \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_{n-1}} & \mathbf{u_r} \end{bmatrix} \begin{bmatrix} \sigma_1 & & \cdots & & 0 \\ & \sigma_2 & & & \\ \vdots & & \ddots & & \vdots \\ & & & \sigma_{r-1} & \\ 0 & & \cdots & & \sigma_r \end{bmatrix} \begin{bmatrix} \mathbf{v_1}^T \\ \mathbf{v_2}^T \\ \vdots \\ \mathbf{v_{r-1}}^T \\ \mathbf{v_r}^T \end{bmatrix} \tag{4}$$

where $\mathbf{u}_i$ are the columns of U and $\mathbf{v}_i^T$ are the rows of V^T. The matrix Σ is a diagonal matrix with the entries σ_i on the diagonal (the diagonal entries of Σ are nonnegative and all other entries of Σ are zero). Because Σ is diagonal, Equation (4) can be written as:

$$A = \mathbf{u_1}\sigma_1\mathbf{v_1}^T + \mathbf{u_2}\sigma_2\mathbf{v_2}^T + \cdots + \mathbf{u_{r-1}}\sigma_{r-1}\mathbf{v_{r-1}}^T + \mathbf{u_r}\sigma_r\mathbf{v_r}^T. \tag{5}$$

As you can see, A is separated into the sum of (rank=1) matrices of the form $\sigma_i\mathbf{u_i}\mathbf{v_i}^T$. With that, the sum in Equation (5) can be expressed as

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i}\mathbf{v_i}^T \tag{6}$$

with r so called singular values $\{\sigma_i \in \mathbb{R} \mid \sigma_i > 0\}$ and both $\{\mathbf{u}_1, \ldots \mathbf{u}_r\} \subseteq {}^m\mathbb{R}$ and $\{\mathbf{v}_1, \ldots \mathbf{v}_r\} \subseteq {}^n\mathbb{R}$ are orthonormal sets of vectors.

What's an "orthonormal set of vectors"? In simple language, each vector has length one and any two vectors are perpendicular (also called "orthogonal"). More specifically, two vectors, $\mathbf{x}, \mathbf{y} \in {}^n\mathbb{R}$, are called *orthogonal* if and only if their dot-product is equal to zero: $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T\mathbf{y} = 0$. The set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k\} \subseteq {}^n\mathbb{R}$ is an *orthogonal set of vectors* if and only if every pair of vectors is orthogonal: $\mathbf{x}_i \cdot \mathbf{x}_j = 0$ if $i \neq j$. The set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k\} \subseteq {}^n\mathbb{R}$ is an *orthonormal set of vectors* if and only if every pair of vectors is orthogonal and each vector has length 1: $\mathbf{x}_i \cdot \mathbf{x}_j = 0$ if $i \neq j$ and $\mathbf{x}_i \cdot \mathbf{x}_i = 1$ for every $1 \leq i, j \leq k$.

At the end of this course, you will have the tools you need to be able to find the SVD of a general matrix.[2] We can use Equation (5) to compress the image A by keeping the "most relevant" terms in the sum: As you can see from Equation (5), some matrices $\sigma_i\mathbf{u_i}\mathbf{v_i}^T$ could be neglected if σ_i is very small. Therefore, a typical strategy for compressing images is putting the singular values σ_i in nonincreasing order, $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$ (as well as sorting the corresponding $\mathbf{u}_i$ and $\mathbf{v}_i$ in U and V, respectively) and only storing $k < r$ of these components in an image file (k is an integer of at least 1). The compressed image A_k is constructed as

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^T \tag{7}$$

This will reduce the image size dramatically, however, the image quality will be degraded if k was picked too small, see for example Figure 2 where 30 components were used ($k = 30$) for the compression of the initial grey-scale image.

[1] In this writing assignment, we are introducing you to the compact (or reduced) SVD, which is slightly different from the full SVD. To save space, we refer to it as the "SVD" throughout.

[2] *For the curious:* $\mathbf{u}_i$ are eigenvectors of the matrix AA^T and $\mathbf{v}_i$ are eigenvectors of the matrix A^TA.

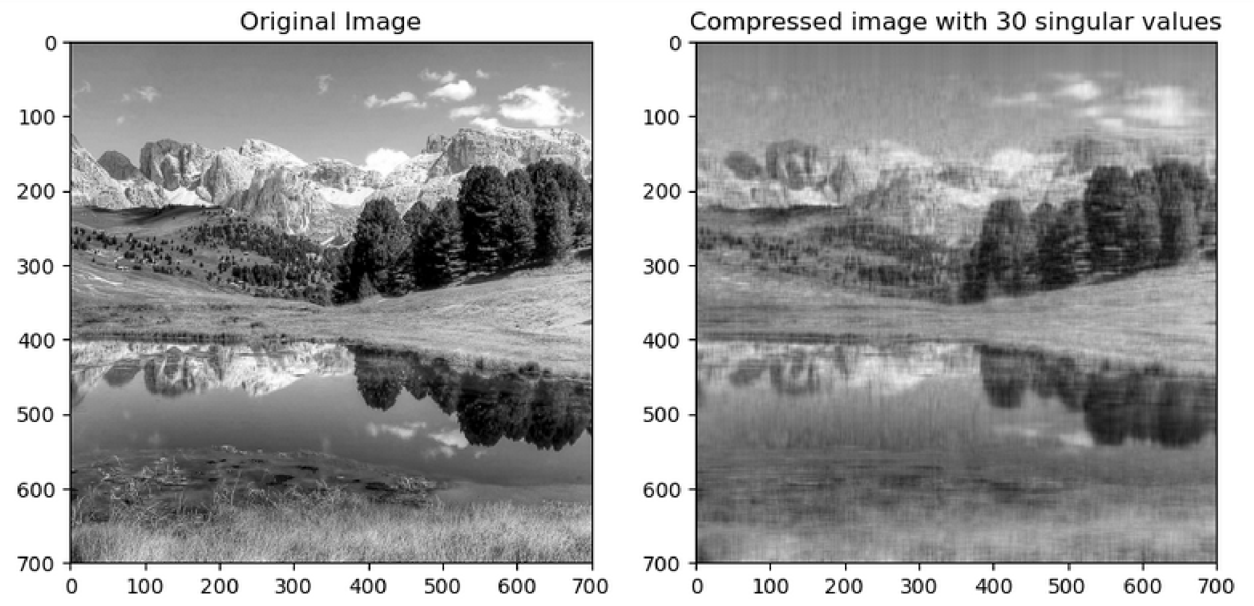


Figure 2: Lossy compression of a 700×700 pixel grey-scale image via Single Value Decomposition (SVD).

(a) For what values of k would A_k corresponds to a lossy compressed image? No additional explanation is necessary.

For all $0 \leq k < \text{rank}(A)$

(b) Let $A = \sigma \mathbf{x}\mathbf{y}^T$ where $\mathbf{x}, \mathbf{y} \in {}^n\mathbb{R}$ and $\sigma \in \mathbb{R}$ is nonzero. Prove that $\text{rank}(A) = 1$. *Hint:* Try constructing a 3×3 example, A, by choosing some nonzero $\sigma \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in {}^3\mathbb{R}$. This should give you the insight you need to answer this question for general $\mathbf{x}, \mathbf{y} \in {}^n\mathbb{R}$ below.

Let $A = \sigma \mathbf{x}\mathbf{y}^T$ where $\mathbf{x}, \mathbf{y} \in {}^3\mathbb{R}$ and $\sigma \in \mathbb{R}$ is nonzero. Then, WLOG, we use a 3×3 example:

$$A = \sigma \mathbf{x}\mathbf{y}^T = \sigma \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} \sigma x_1 y_1 & \sigma x_1 y_2 & \sigma x_1 y_3 \\ \sigma x_2 y_1 & \sigma x_2 y_2 & \sigma x_2 y_3 \\ \sigma x_3 y_1 & \sigma x_3 y_2 & \sigma x_3 y_3 \end{bmatrix}$$

This extends to a general $n \times n$ matrix A:

$$A = \sigma \mathbf{x}\mathbf{y}^T = \sigma \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} \sigma x_1 y_1 & \sigma x_1 y_2 & \cdots & \sigma x_1 y_n \\ \sigma x_2 y_1 & \sigma x_2 y_2 & \cdots & \sigma x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ \sigma x_n y_1 & \sigma x_n y_2 & \cdots & \sigma x_n y_n \end{bmatrix}$$

We can see that the matrix A is a rank-1 matrix, since all the columns are scalar multiples of each other. Thus, $\text{rank}(A) = 1$.

(c) Let $A = \begin{bmatrix} 36 & 9 & 12 \\ -48 & -12 & -16 \\ 144 & 36 & 48 \end{bmatrix}$. Determine the singular value decomposition $A = \sigma \mathbf{x}\mathbf{y}^T$.

$\mathrm{Col}(A) = \begin{bmatrix} 3 & -4 & 6 \end{bmatrix}^T$ so Rank $A = 1$. We normalize the column vector to get $\mathbf{x} = \begin{bmatrix} 3/\sqrt{61} \\ -4/\sqrt{61} \\ 6/\sqrt{61} \end{bmatrix}$. Also, we normalize $y = \begin{bmatrix} 12/13 & 3/13 & 4/13 \end{bmatrix}^T$.

Thus, $A = \sigma \mathbf{x}\mathbf{y}^T = 13\sqrt{61} \begin{bmatrix} 3/\sqrt{61} \\ -4/\sqrt{61} \\ 6/\sqrt{61} \end{bmatrix} \begin{bmatrix} 12/13 & 3/13 & 4/13 \end{bmatrix}$

(d) Assume $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ is an orthogonal set of nonzero vectors. Prove that the set is linearly independent.

We consider the linear combinations of the set of vectors that equals the zero vector:

$$\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \lambda_3 \mathbf{v}_3 = \mathbf{0}$$

Since the set is orthogonal, we have:

$$\begin{cases} \mathbf{v}_1 \cdot \mathbf{v}_2 = 0 \\ \mathbf{v}_1 \cdot \mathbf{v}_3 = 0 \\ \mathbf{v}_2 \cdot \mathbf{v}_3 = 0 \end{cases}$$

We can then take the dot product of the linear combination with $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, also since $0 \cdot \mathbf{v}_i = 0$, we have:

$$\mathbf{v}_1 \cdot (\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \lambda_3 \mathbf{v}_3) = 0 \implies \lambda_1 \mathbf{v}_1 \cdot \mathbf{v}_1 + \lambda_2 \mathbf{v}_1 \cdot \mathbf{v}_2 + \lambda_3 \mathbf{v}_1 \cdot \mathbf{v}_3 = \lambda_1 \|\mathbf{v}_1\|^2 = 0$$

Similarly, we will have:

$$\lambda_2 \|\mathbf{v}_2\|^2 = 0 \quad \lambda_3 \|\mathbf{v}_3\|^2 = 0$$

Since $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are non-zero vectors, we have $\|\mathbf{v}_1\|^2, \|\mathbf{v}_2\|^2, \|\mathbf{v}_3\|^2 > 0$. Thus, $\lambda_1 = \lambda_2 = \lambda_3 = 0$, which implies that the set of vectors is linearly independent.

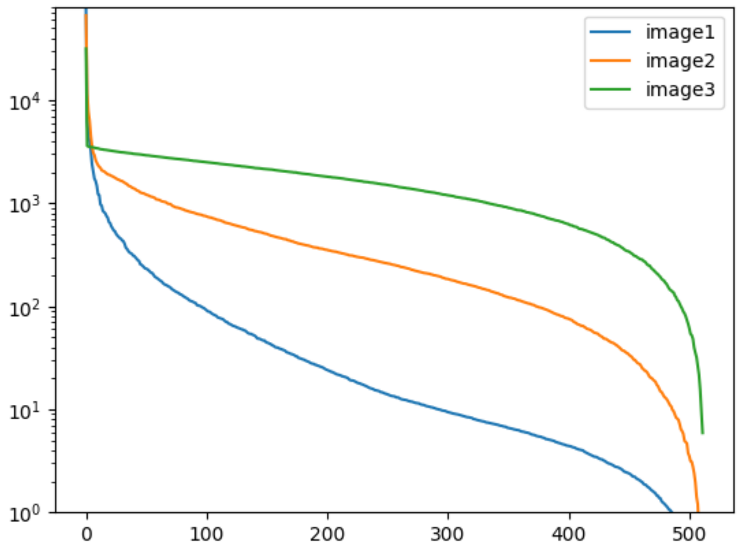
(e) Consider the set $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ from part (d). What property of a 4th vector $\mathbf{v}_4 \in {}^4\mathbb{R}$ would ensure that the extended set $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ spans ${}^4\mathbb{R}$? Give a short explanation. Note: your explanation does not have to include how you would find such a $\mathbf{v}_4$.

v_4 shall be orthogonal to all vectors in the set $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. This ensures that the extended set $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ spans ${}^4\mathbb{R}$, since the set would remain linearly independent. By the Fundamental Theorem of Linear Algebra we learnt in MAT185, the size 4 set of linear independent vectors would span ${}^4\mathbb{R}$, a dim 4 vecotor space. This is beacuse the spanning set of ${}^4\mathbb{R}$ is at least 4, so a set of 4 linearly independent vectors would span ${}^4\mathbb{R}$ as it constitutes the basis thereof.

(f) Download the Jupyter notebook (see Quercus page of *Assignment 2*), which includes the python code of an SVD analysis. Please also download the three example grey-scale images *image1.png*, *image2.png*, and *image3.png* from the same Quercus page. You can run the code on the U of T Jupyter server, on your own local Jupyter installation, or on the ECF lab PCs. We recommend using the U of T Jupyter server, since all necessary packages are already installed: https://jupyter.utoronto.ca/hub/user-redirect/tree. For that, please login with your U of T credentials and click '*upload*' to upload the IPYNB file and images. After that, you can open the uploaded notebook and run the Python code.

Use the Jupyter Python code to analyze the provided three images. Plot the sorted singular values σ_i over the index i in one graph and discuss the results for all three provided images. Do you see a pattern of the result depending on each input image?

Insert your plot below. In addition, briefly discuss which of the 3 images would allow for the highest and which for the lowest compression ratio by singular value decomposition. *Hint:* The compression ratio is the uncompressed file size over the compressed file size. You don't need to calculate this ratio. However, think about which image would need the largest k and which the lowest k in Equation (7) for an accurate representation of the uncompressed image, Equation (6). For plotting σ_i, see the comments and plotting commands in the Jupyter file. Also, keep the logarithmic Y-axis for your graph.



The graph shows the sorted singular values σ_i over the index i for the three provided images. We can see that the singular values for *image1.png* decay the slowest, while the singular values for *image3.png* decay the fastest. This implies that *image1.png* would allow for the highest compression ratio, since it requires less k for an accurate representation of the uncompressed image. On the other hand, *image3.png* would allow for the lowest compression ratio, since it requires more k for an accurate representation of the uncompressed image. If it has a faster decay, it means that the there would be less small singular values.

(g) Calculate the SVD of *image1.png* with the given Jupyter code. For what choice of $k \in \mathbb{Z}$ would you expect a reduction in file size? To answer this question, estimate the memory needed to store the SVD-compressed image and compare it to the memory needed for the uncompressed 8 bpp version of the image. *Hint:* Estimate the memory size of the uncompressed image by using its size and given colour depth. Also note that each floating-point number uses 32 Bit memory when stored.

Size of *image1.png* = 512×512 pixels, 8 bpp.
Memory needed for uncompressed image = $512 \times 512 \times 8 = 2^{21} = 2\,097\,152$ bits.

Assumption We assume that the fullt *image1.png* is rank 512. In reality it might not, but 512 is the maximum rank possible, so any k as a result of this will garantee a reduction in file size.

Memory needed for $\Sigma = k \times 32$ bits.
Memory needed for each of $U, V = 2 \times k \times 512 \times 32$ bits.
Total memory needed for SVD-compressed image = $k \times 32 + 2 \times k \times 512 \times 32 = 32k + 32768k = 32800k$ bits.

To expect a reduction in file size, we would need $32800k < 2\,097\,152 \implies k < 64$.