利用 SVD 做影像壓縮

數值分析 (第五組)

NTNU MATH

January 7, 2019

指導教授: 謝世峰

Introduction

- 問題：電腦儲存影像本來就很耗費功夫，例如儲存 800×600 的全彩點陣圖需要花 $800 \times 600 \approx 31.4$ MB 的空間
- 目的：先對照片做壓縮，再把壓縮後的資料經由網路傳送出去，對方收到了以後對其進行解壓縮取得照片，可以加快傳輸的速度
- 作法：奇異值分解進行影像壓縮

List

Singular value decomposition

Theorem 1 (Singlular Value Decomposition)

給定 $A \in \mathbb{R}^{m\times n}$ 為任意的實數矩陣，則存在兩個 orthogonal matrices $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ 與另外一個 diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ 使得

$$A = U\Sigma V^{\top}$$

Singular value decomposition

Theorem 1 (Singlular Value Decomposition)

給定 $A \in \mathbb{R}^{m\times n}$ 為任意的實數矩陣，則存在兩個 orthogonal matrices $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ 與另外一個 diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ 使得

$$A = U\Sigma V^{\top}$$

Remark 1

若 $A \in \mathbb{R}^{m\times n}$ 利用上面的定理，把它寫成 $A = U\Sigma V^{\top}$，則

$$\operatorname{rank}(A) = \operatorname{rank}(\Sigma) \text{ (因為 } U, V \text{ 都是可逆的矩陣)}$$

The Partitioned matrices of SVD

給定 $A \in \mathbb{R}^{m\times n}$, 利用 SVD 將其分解，得 $A = U\Sigma V^\top$，將其寫成

$$\begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix} \begin{bmatrix} v_1^\top \\ \vdots \\ v_r^\top \end{bmatrix}$$

，則稱為 partitioned matrices of SVD

Definition 1

給定 $A \in \mathbb{R}^{m\times n}$，將其寫成

$$A = U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i {v_i}^\top$$

若 $1 \leq k < r$，定義

$$A_k := \sum_{i=1}^{k} \sigma_i u_i {v_i}^\top$$

$$E_k := A - A_k = \sum_{i=k+1}^{r} \sigma_i u_i {v_i}^\top$$

上面這個定義中，A_k 被稱為 A 的 rank k approximation

因此我們可以把影像看成一個矩陣 $A \in \mathbb{R}^{m\times n}$，裡面的係數都是實數（更嚴格來說，是非負的整數，比如說 8 位元灰階就是從 0 到 255）接著我們對這個矩陣做 SVD，得

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

為了方便之後的討論，假設

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$$

經過排序後，因為後段的奇異值比較小，對原本的矩陣 A 來說並沒有那麼重要，因此我們採用的方法是選取 $k \in [1, r]$，利用 A_k 對 A 進行近似，捨棄後面幾項，讓電腦只需保留前幾項就好。

Remark 2

從上面的討論，我們可以得知 SVD 是一種有損壓縮法，意思就是資料（影像）經過壓縮，再解壓縮會跟原資料（原影像）有一些差異。

那麼這個 k 要如何選取呢？ 以下我們會提供一個方法

List

Frobenius norm

Definition 2 (Frobenius norm)

給定 $A \in \mathbb{R}^{m\times n}$，定義 A 的 Frobenius norm 為

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

依照這個定義，我們可以把 A 的 Frobenius norm 想成在 $\mathbb{R}^{nm}$ 上的 norm，因此可以定義兩個相同大小矩陣 A,B 的內積

Frobenius norm

Definition 2 (Frobenius norm)

給定 $A \in \mathbb{R}^{m\times n}$，定義 A 的 Frobenius norm 為

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

依照這個定義，我們可以把 A 的 Frobenius norm 想成在 $\mathbb{R}^{nm}$ 上的 norm，因此可以定義兩個相同大小矩陣 A,B 的內積

Remark 3

令 $X \in \mathbb{R}^{m\times n}$, 則 $\|X\|_F^2 = X \cdot X$

Frobenius norm

Lemma 1

給定 $X \in \mathbb{R}^{m \times p}$, $Y \in \mathbb{R}^{p \times n}$

Frobenius norm

Lemma 1

給定 $X \in \mathbb{R}^{m\times p}$, $Y \in \mathbb{R}^{p\times n}$

1. 若 $n = p$ 且 Y 是 orthogonal, 則 $\|XY\|_F = \|X\|_F$.

Frobenius norm

Lemma 1

給定 $X \in \mathbb{R}^{m\times p}$, $Y \in \mathbb{R}^{p\times n}$

1. 若 $n = p$ 且 Y 是 orthogonal, 則 $\|XY\|_F = \|X\|_F$.
2. 若 $m = p$ 且 X 是 orthogonal, 則 $\|XY\|_F = \|Y\|_F$.

Frobenius norm

Lemma 1

給定 $X \in \mathbb{R}^{m\times p}$, $Y \in \mathbb{R}^{p\times n}$

1. 若 $n = p$ 且 Y 是 orthogonal, 則 $\|XY\|_F = \|X\|_F$.
2. 若 $m = p$ 且 X 是 orthogonal, 則 $\|XY\|_F = \|Y\|_F$.

Theorem 2 (The Frobenius norm of matrix A)

給定 $A = U\Sigma V^\top$ 為 A 的奇異值分解形式則 A 的 Frobenius norm

$$\|A\|_F = \|\Sigma\|_F = \left(\sum_{i=1}^{r} \sigma_i^2\right)^{\frac{1}{2}}$$

其中 $r = \operatorname{rank}(A)$

Frobenius norm

Theorem 3

給定 $A \in \mathbb{R}^{m\times n}$, $A = U\Sigma V^{\top} = \sum_{i=1}^{k} \sigma_i \, \mathbf{u_i}\mathbf{v_i}^{\top}$ ，則以下的結論都會成立

Frobenius norm

Theorem 3

給定 $A \in \mathbb{R}^{m\times n}$, $A = U\Sigma V^{\top} = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^{\top}$ ，則以下的結論都會成立

1. $\|A - A_r\|_F = \sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_k^2$

Frobenius norm

Theorem 3

給定 $A \in \mathbb{R}^{m\times n}$, $A = U\Sigma V^\top = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^\top$ ，則以下的結論都會成立

1. $\|A - A_r\|_F = \sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_k^2$
2. $\|A\|_F^2 = \|A_r\|_F^2 + \|E_r\|_F^2$

Frobenius norm

如果我們用 A_k 近似 A ，那它在 Frobenius norm 之下的相對誤差就是

$$e(k) = \frac{\|A - A_k\|_F}{\|A\|_F} = \sqrt{\frac{\sigma_{k+1}^2 + \cdots + \sigma_r^2}{\sigma_1^2 + \cdots + \sigma_r^2}}$$

其中 $e(k) \in [0, 1), \forall k = 1, \cdots, r$

但是電腦計算根號容易產生誤差，因此考慮平方後的結果

$$e^2(k) = \frac{\sigma_{k+1}^2 + \cdots + \sigma_r^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

其中 $e^2(k) \in [0, 1), \forall k = 1, \cdots, r$。

而我們定義

$$\tau(k) := 1 - e^2(k)$$

利用 Theorem 3 得

$$\tau(k) = 1 - \frac{\|E_k\|_F^2}{\|A\|_F^2} = \frac{\|A_k\|_F^2}{\|A\|_F^2} = \frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

因為 $e(k) \in [0, 1), \forall k = 1, \cdots, r$，所以 $\tau(k) \in (0, 1], \forall k = 1, \cdots, r$
而 $\tau(k)$ 越高則代表 A_k 跟 A 比的相對誤差越低。

根據我們的實驗，對一般比較小的照片 $(800 \times 800$ 內$)$，只要 $\tau(k)$ 超過 0.999，用 A_k 做出來的壓縮與原圖相比，沒有太大差別。

Numerical result for taking $\tau(k) > 0.999$

拿 lena512.jpg 來做 SVD 影像壓縮的示範，

Figure 1: lena512.jpg(圖片)

Numerical result for taking $\tau(k) > 0.999$

下圖是奇異值的分佈 (x, y) 對應的是 $y = \sigma_x$

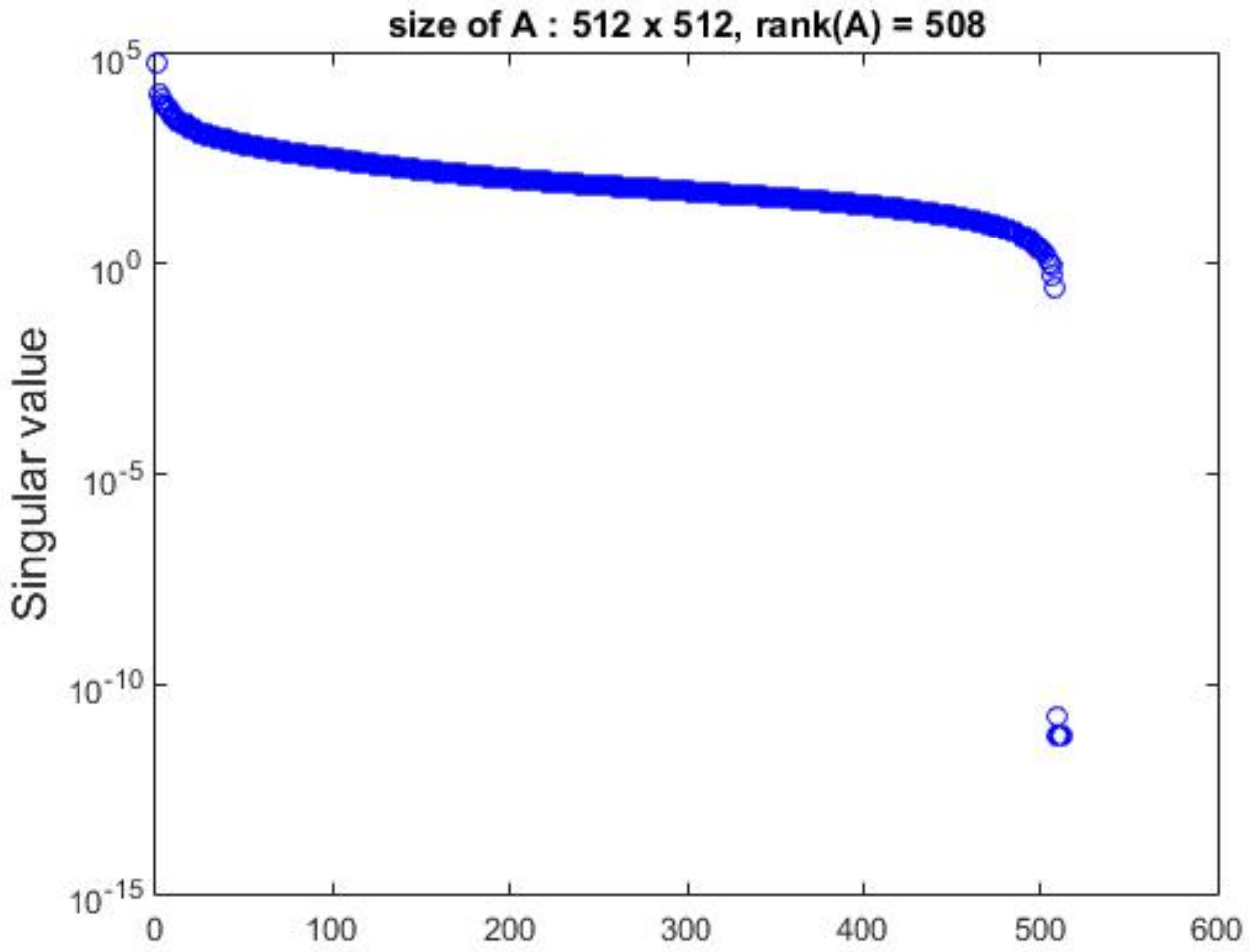


Figure 2: lena512.jpg 奇異值的分佈

Numerical result for taking $\tau(k) > 0.999$

以下就是針對每一個 k，算 $\tau(k)$ 畫出的圖。

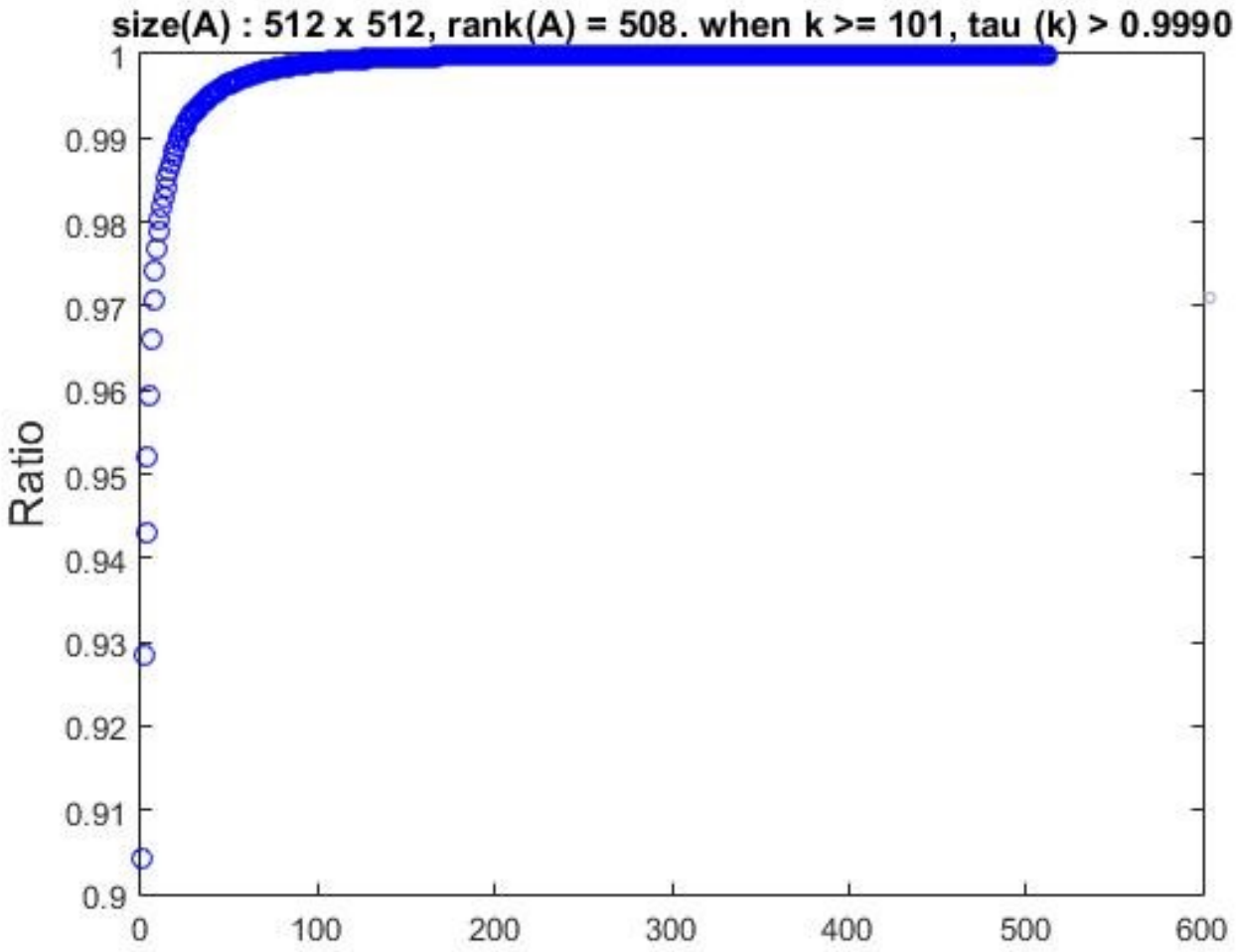


Figure 3: lena512.jpg 根據 k 取的大小 $\tau(k)$ 值的變化

Numerical result for taking $\tau(k) > 0.999$

接下來看一下不同的 rank 的壓縮效果。

Figure 4: lena512.jpg rank 不同造成的清晰度差異

Numerical result for taking $\tau(k) > 0.999$

Figure 5: rank=101 的 lena512.jpg

與原圖相比，k 取 101 ($\tau \geq 0.999$) 就足夠清晰了。

Numerical result for taking $\tau(k) > 0.999$

Figure 6: $\tau(k) > 0.999$ 後造成的清晰度差異

Numerical results for a 1920 × 1200 image

然而，只要照片很大的話 $\tau(k) > 0.999$ 也有可能不夠用，下圖就是一個例子。

Figure 7: violet.png(大圖片)

Numerical result for a 1920 × 1200 image

如果取最小的 k 滿足 $\tau(k) > 0.999$ 的話，這個誤差就很明顯了。

Figure 8: 取 $\tau(k) \approx 0.999$ 的 violet.png

Observation

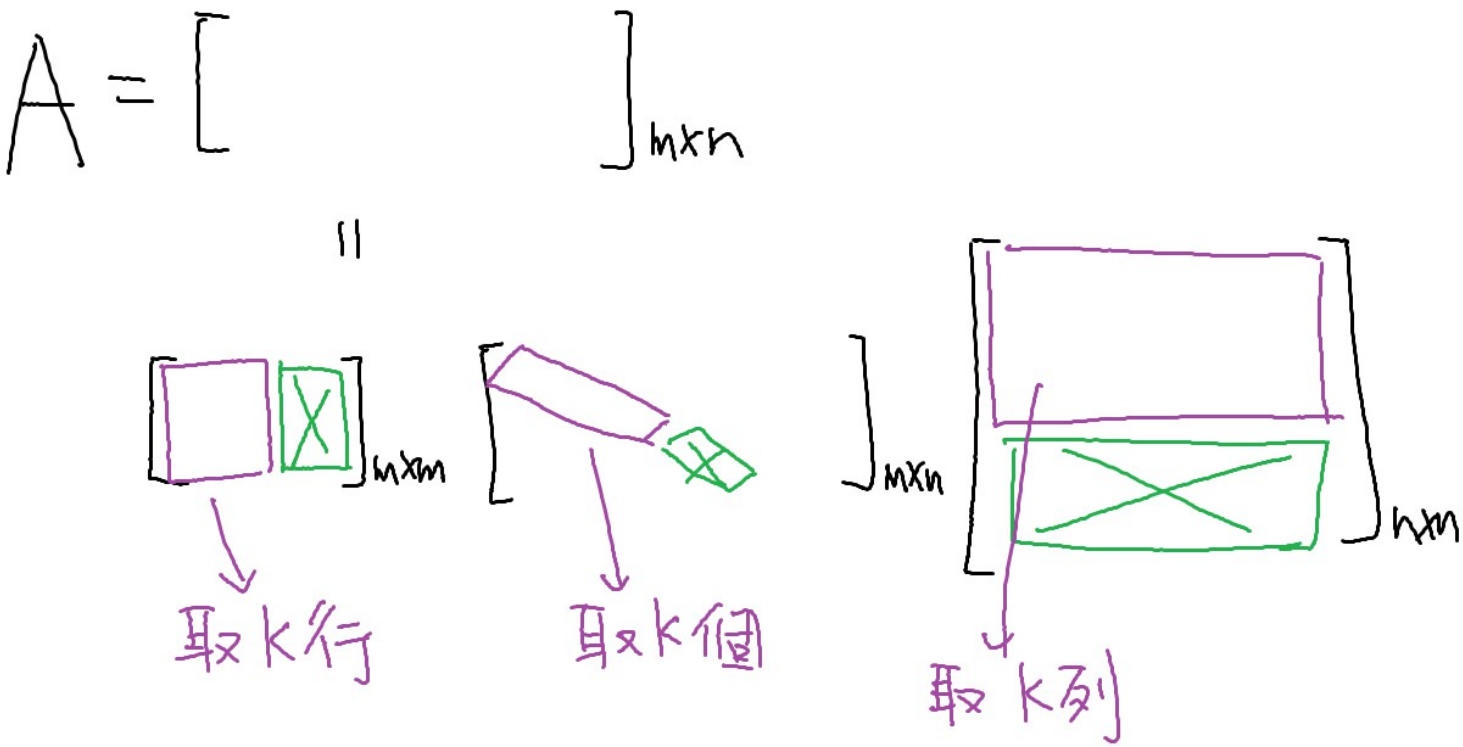
與原圖相比，誤差是看的出來的。

因此，對於一些比較大的圖，$\tau(k)$ 的標準或許要調高一點 (例如: 0.9995)。

List

假設我們拿到了一張 $m \times n$ 的照片，不做壓縮的話 A 要存 mn 個整數
至於上面我們都在把 A 做 SVD，然後取 A_k 這個近似矩陣那我們也來看看 A_k 到底要存多少資料。首先把 A_k 寫成

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top$$



因此 A_k 要存 $k(1+m+n)$ 個數。

然而，這個時候自然會希望

$$\frac{k(1+m+n)}{mn} < 1 \qquad \Longrightarrow \qquad k < \frac{mn}{1+m+n}$$

當 k 直到接近 $\left\lfloor \frac{mn}{1+m+n} \right\rfloor$ 的時候，壓出來的效果還是不理想的話，那就不要壓了，一來有損，二來要儲存的東西反倒變多了。

不過我們當然不希望每次 k 都要取到最接近的，還是會希望可不可以多砍一點。而對比較大張的圖的話，大概取 k 滿足 $\frac{k(1+m+n)}{mn} < \frac{2}{3}$ 大概就差不多了，以剛剛那張圖做例子

Figure 9: rank492 的 violet.png

List

這部份主要探討背景以白色為底，僅以黑色線條構成的圖。(用手寫板在小畫家上寫的字)

Figure 10: 家.png(文字)

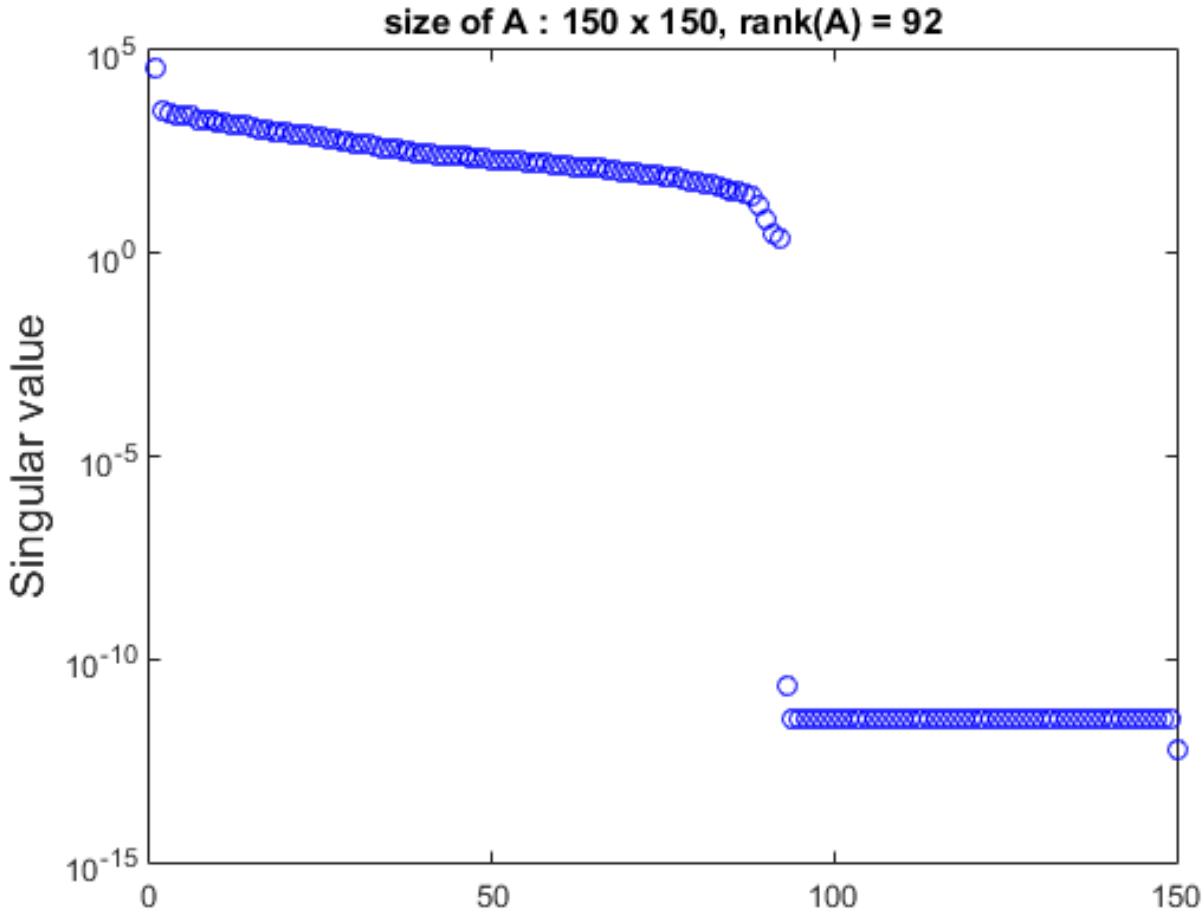


Figure 11: 家.png 奇異值的分佈

其中，取最小的 k 使得 $\tau(k) > 0.999$，$k = 38$。
接下來用剛剛的方式來試試。

Figure 12: rank 38 approximation

由圖可知，出現了大量雜訊。

Figure 13: rank 38 approximation，去噪

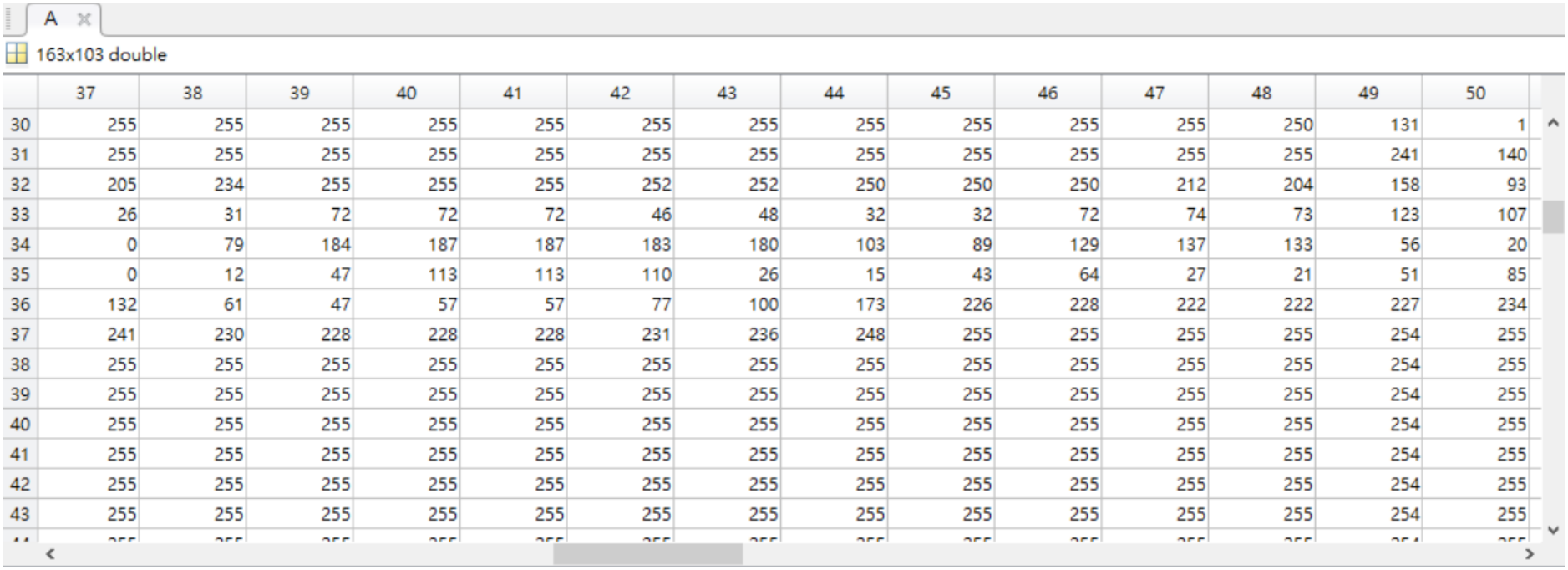
去噪: 對矩陣裡面每一個格子，檢查他的值都沒有低過某個數值 (越低會越黑，是雜訊來源)，如果有就把這個格子的值調成 255(全白)。

因此，我們還是可以用取 $\tau(k) = 0.999$ 的方法，讓電腦存下這些資料，只不過在解壓縮的時候要多加一步去噪的動作。

接著，我們試試用手寫在白紙上，用掃描器去掃出來後的情形。

Figure 14: 掃描家.png

我們來檢查這個矩陣。



A

163x103 double

	37	38	39	40	41	42	43	44	45	46	47	48	49	50
30	255	255	255	255	255	255	255	255	255	255	255	250	131	1
31	255	255	255	255	255	255	255	255	255	255	255	255	241	140
32	205	234	255	255	255	252	252	250	250	250	212	204	158	93
33	26	31	72	72	72	46	48	32	32	72	74	73	123	107
34	0	79	184	187	187	183	180	103	89	129	137	133	56	20
35	0	12	47	113	113	110	26	15	43	64	27	21	51	85
36	132	61	47	57	57	77	100	173	226	228	222	222	227	234
37	241	230	228	228	228	231	236	248	255	255	255	255	254	255
38	255	255	255	255	255	255	255	255	255	255	255	255	254	255
39	255	255	255	255	255	255	255	255	255	255	255	255	254	255
40	255	255	255	255	255	255	255	255	255	255	255	255	254	255
41	255	255	255	255	255	255	255	255	255	255	255	255	254	255
42	255	255	255	255	255	255	255	255	255	255	255	255	254	255
43	255	255	255	255	255	255	255	255	255	255	255	255	254	255

Figure 15: 掃描家.png 的部分矩陣

結果造成很多應該要是 0 的地方卻變成非 0 整數，讓整個矩陣變得更複雜。例如：(255,0,0) 和 (255,0,1) 只差了一點點，但在 $\mathbb{R}^3$ 就是線性獨立的了。

接下來我們針對一些常用的字，去跑一次 SVD (取 $\tau(k) > 0.999$)，希望可以粗略估計用多少奇異值可以讓字壓的清楚，而不需要每次都跑一次這樣的流程。

另一方面，因為用掃描器掃進來的字，變成矩陣後的大小都不一樣，因此要討論的並不是「取多少個奇異值」，而是想因為 $A \in M_{m\times n}(\mathbb{R})$，SVD 之後的奇異值會有 $\min\{m, n\}$ 個 (如果把 0 也算進來的話)。
所以我們接著主要討論的是「所取奇異值個數與短邊之比值」。

Type	word	rows	columns	min(m,n)	σ	ratio
掃描器	家	163	103	103	37	0.359223
手寫板	家	150	150	150	38	0.253333
掃描器	大	132	141	132	36	0.272727
手寫板	大	132	141	132	34	0.257575
掃描器	花	109	94	94	33	0.351064
手寫板	花	109	94	94	36	0.382978
掃描器	都	124	88	88	35	0.397727
手寫板	都	124	88	88	30	0.340909
掃描器	是	218	129	129	30	0.232558
手寫板	是	218	129	129	29	0.224806
掃描器	叫	113	107	107	17	0.158879
手寫板	叫	113	107	107	17	0.158879
掃描器	我	125	110	110	36	0.327273
手寫板	我	125	110	110	36	0.327273

經過前頁的資料，我們或許可以推斷說「當比值到達 0.4 的時候，壓縮出來 (含去噪) 的字就會看得清楚了」。

目前我們有 2 種方法來取 k 值

1. 取最小的正整數 k 使得 $\tau(k) > 0.9990$。

經過前頁的資料，我們或許可以推斷說「當比值到達 0.4 的時候，壓縮出來 (含去噪) 的字就會看得清楚了」。

目前我們有 2 種方法來取 k 值

1. 取最小的正整數 k 使得 $\tau(k) > 0.9990$。
2. 取 ratio≈0.4 的 k 並且去噪。

接下來會探討「一篇文章」，我們會怎麼做。
以下這張圖像 (文章) 取自老師的檔案。

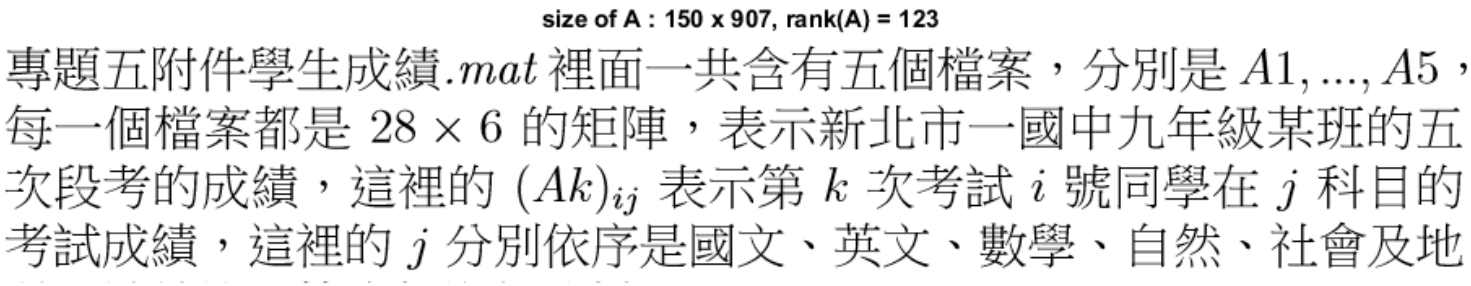


Figure 16: 文章.png（原圖）

其中 95 是最小的 k 使得 $\tau(k) > 0.9990$

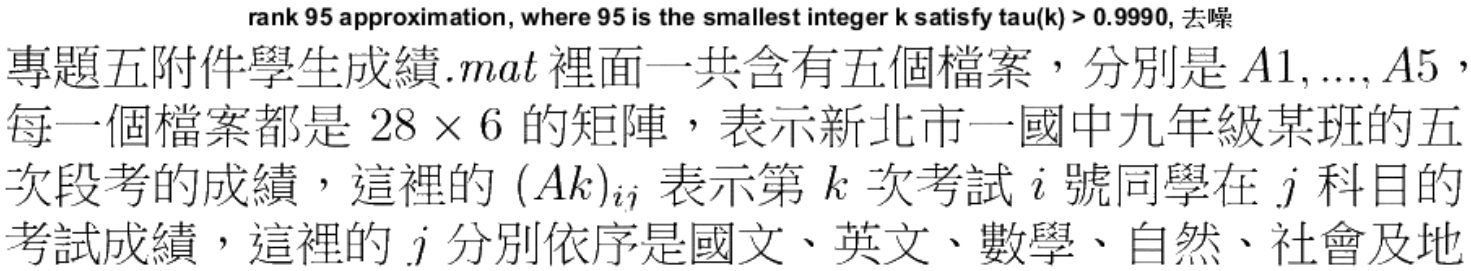


Figure 17: 文章.png 的 rank95 approximation

專題五附件學生成績.mat 裡面一共含有五個檔案，分別是 $A1, ..., A5$，每一個檔案都是 28×6 的矩陣，表示新北市一國中九年級某班的五次段考的成績，這裡的 $(Ak)_{ij}$ 表示第 k 次考試 i 號同學在 j 科目的考試成績，這裡的 j 分別依序是國文、英文、數學、自然、社會及地

Figure 18: ratio≈0.4 的奇異值去做壓縮，去噪

比起單字來說，文章顯然的，ratio 不能只取 0.4。

這時我們來探討「所占空間」

1. 不壓縮 (原圖): $150 \times 907 = 136050$

其中 $\frac{100510}{136050} \approx 73.877\%$
看起來壓的並不多，因此我們提供一個新的方法來壓「一篇文章」。

專題五附件學生成績.mat 裡面一共含有五個檔案，分別是 $A1, ..., A5$，每一個檔案都是 28×6 的矩陣，表示新北市一國中九年級某班的五次段考的成績，這裡的 $(Ak)_{ij}$ 表示第 k 次考試 i 號同學在 j 科目的考試成績，這裡的 j 分別依序是國文、英文、數學、自然、社會及地

Figure 18: ratio≈0.4 的奇異值去做壓縮，去噪

比起單字來說，文章顯然的，ratio 不能只取 0.4。

這時我們來探討「所占空間」

1. 不壓縮 (原圖): $150 \times 907 = 136050$
2. rank 95 approximation : $95(150+907+1) = 100510$

其中 $\frac{100510}{136050} \approx 73.877\%$
看起來壓的並不多，因此我們提供一個新的方法來壓「一篇文章」。

首先我們可以先對這篇文章做分塊的動作，如圖：

Figure 19: 文章.png 的分塊動作

將這篇文章切開 4×30 份，每一份大概都被一個字佔據。

對每一塊分別做 SVD，取 ratio=40% 的奇異值去做壓縮及去噪，壓出來的結果如下

專題五附件學生成績.*mat* 裡面一共含有五個檔案，分別是 $A1, ..., A5$，每一個檔案都是 28×6 的矩陣，表示新北市一國中九年級某班的五次段考的成績，這裡的 $(Ak)_{ij}$ 表示第 k 次考試 i 號同學在 j 科目的考試成績，這裡的 j 分別依序是國文、英文、數學、自然、社會及地

Figure 20: 文章.png 切塊後分別 SVD 後的結果

那我們來看看切過再 SVD 後的大小:

$(0.4 \times 30) \times (\frac{150}{4} \times \frac{907}{30} + 1) \times (4 \times 30) \approx 99882$

考慮 $\frac{99882}{136050} \approx 73.415\%$

雖然這個方法壓了比較多，但是耗費時間量大，所以依然存在他的缺點。

List

先前幾個部份都是在用灰階影像，在最後我們就針對彩圖來做一下 SVD。如果我們把一張大小為 $m \times n$ 的影像讀進 matlab 的話，會得到一個 $m \times n \times 3$ 的矩陣，是由 R、G、B 混合而成。

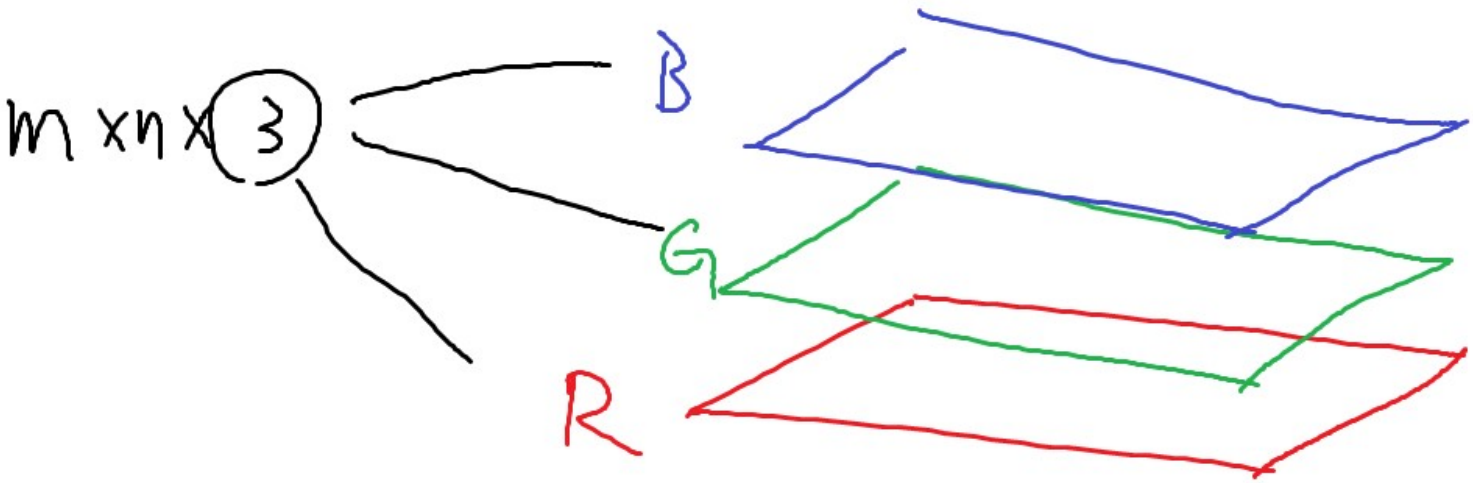


Figure 21: 彩圖的儲存方式

Method 1: (Using τ)

對三個矩陣 R,G,B 各自做 SVD，各自取 $\tau(k_i) > 0.999$，分別生出 $R_{k_1}, G_{k_2}, B_{k_3}$ 三個矩陣，再把它們疊起來。

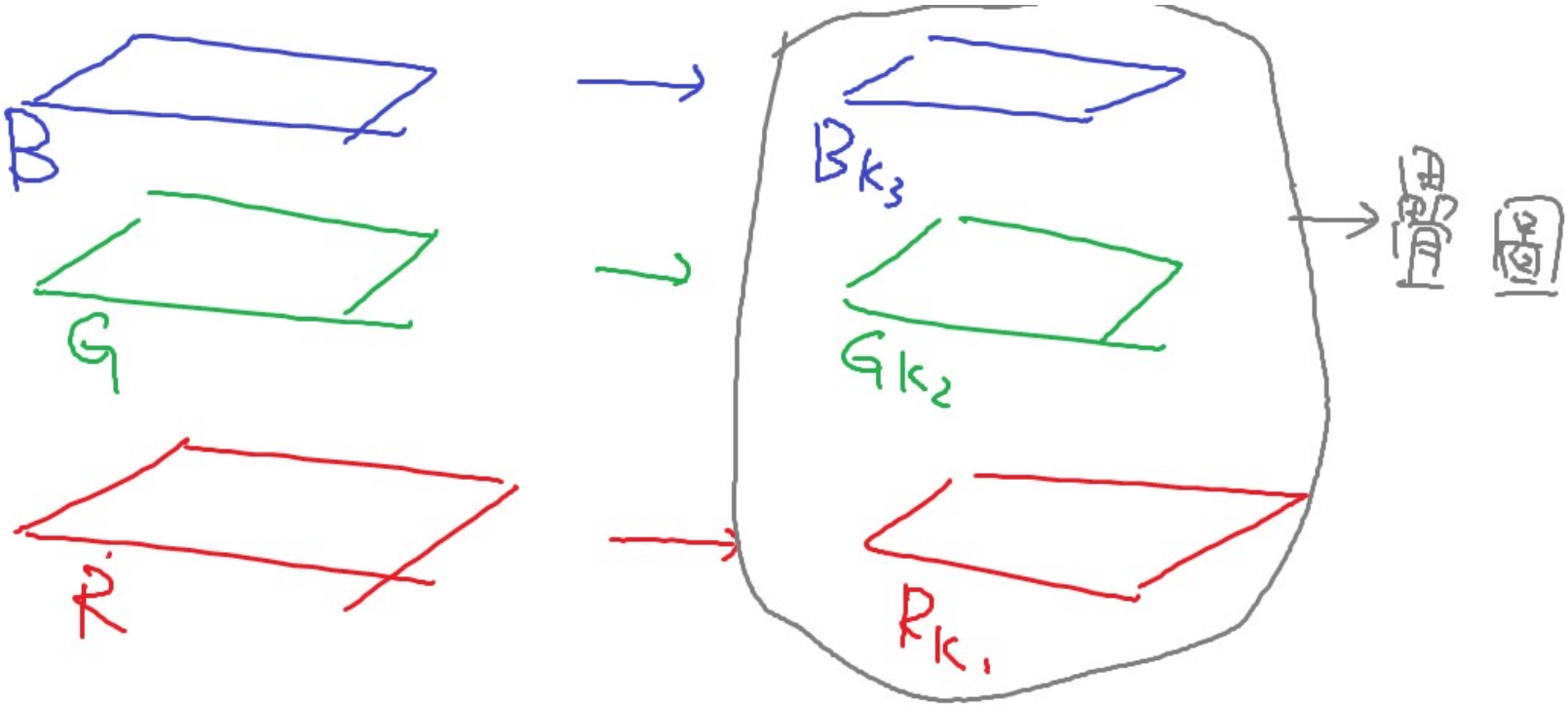


Figure 22: R、G、B 分別做 SVD 後再疊圖

Method 2: (Find k such that $\frac{k(m+n+1)}{mn} < \frac{2}{3}$)

找到 k_0 使得 $\frac{k_0(m+n+1)}{mn} < \frac{2}{3}$，再對 R、G、B 各自做 SVD，但取共同的 k_0，生出 R_{k_0} G_{k_0} B_{k_0} 三個矩陣，再把它們疊起來。

Figure 23: 柴犬.bmp (原圖)

Figure 24: 柴犬.bmp (using Method 1)

Figure 25: 柴犬.bmp (using Method 2)

這張 720×720 的照片，Method 1 的 R、G、B 分別取 $\tau > 0.999$ 還是可以看出一些差異。而 Method 2 雖然要佔的空間比 Method 1 較多，但至少它可以確保一定比原圖存得少，而且品質相對穩定。

Future Works

1. 沒有所謂的「常用圖片」，故難找固定的 $\tau(k)$，遇到一張新影像還是得讓程式跑，才能找出適合的 k。

Future Works

1. 沒有所謂的「常用圖片」，故難找固定的 $\tau(k)$，遇到一張新影像還是得讓程式跑，才能找出適合的 k。
2. 「切過再用 SVD 壓縮」的方式，「全白處」還是得讓他壓，浪費時間浪費資源，能否先行判斷哪些不壓?

Future Works

1. 沒有所謂的「常用圖片」，故難找固定的 $\tau(k)$，遇到一張新影像還是得讓程式跑，才能找出適合的 k。
2. 「切過再用 SVD 壓縮」的方式，「全白處」還是得讓他壓，浪費時間浪費資源，能否先行判斷哪些不壓?
3. 有沒有一個好的切的方法可以讓「切過再用 SVD 壓縮」發揮更高效能?

Thanks for listening!