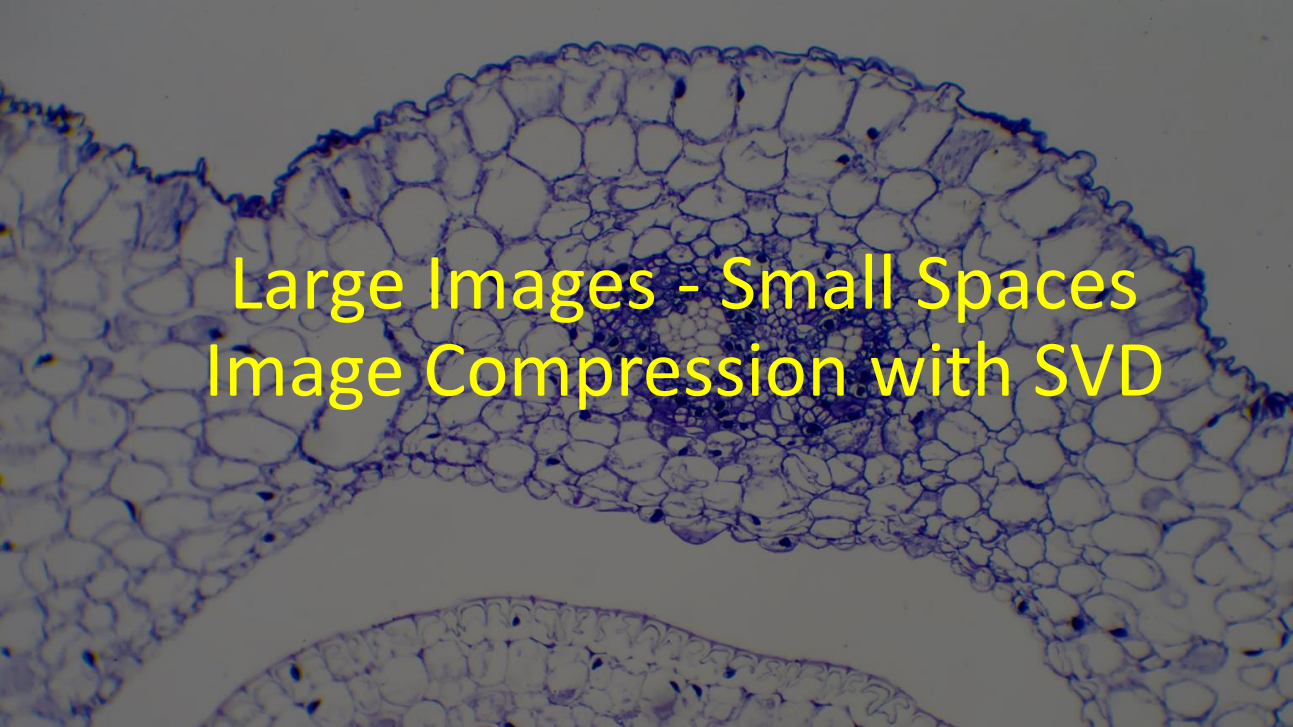

Large Images - Small Spaces
Image Compression with SVD

“*There is nothing quite so terrifying as the results of education.*”

Soseki Natsume (1867 – 1916)
I am a Cat

Motivating Problem

- You have just taken a photograph (with a “smart” phone).
- You want to share this so you decide to send it around as an MMS attachment.

BUT

- It is SO LARGE you quickly come close to exhausting your data.

WHAT TO DO???

Possible Options

1. *Wait* for your data allowance to be *renewed*.
2. *Pay for* (or find someone to pay for) *more data*.
3. *Send it once* and ask for its recipient to forward it.
4. *Forget about it*.

OR

5. Try to *MAKE IT much smaller*

More precisely "*describes its content using less space*".

How do we do this?

- First consideration: what is an *Image File*?
- It's a *table* (or *matrix*) of *integers*.
- In other words an Image has:

  A number of *rows* (its *Height*)

  A number of *columns* (its *Width*)

  The value in *row* p and *column* q describes the *colour*.

Details

- The image on the preceding slide: *388 rows* and *504 columns*.
- Each entry uses *24 bits* to encode the *colour* (RGB scheme).
- The total space occupied is roughly *0.5 Megabytes*.
- Suppose we want to use *much less space*?
- Say roughly *a fifth* as much.
- Is it *possible* to do so *without losing* too much *detail*?

Exploiting Spectra

- Given an $H \times W$ *image*, P, let M be the corresponding *matrix*.
- Notice that, unlike our previous examples, M will typically *not be a square matrix*.
- This is *unimportant*.
- **Key result**:

"*any $H \times W$ matrix, M, can be described as the product of three matrices – U, S, V with U an $H \times W$ matrix; S a $W \times W$ matrix; and V a $W \times W$ matrix: $M = U \cdot S \cdot V$*"

The Three Matrices

- We *appear* to be using *more space*.
- *Originally* we had HW integers.
- *Now* we have: $HW + W^2 + W^2 = W \cdot (H + 2W)$.
- The "*trick*" is what we can do with the matrix S.
- If we could "*throw away*" some rows and columns in S then we would *also* have to "*throw away*" *columns* in U and *rows* in V: otherwise the matrix product would be ill-formed.
- If we keep only k rows and columns: S_k
- We end up with space: $k \cdot (H + k + W) \ll HW$.

So what are U, S, V?

- Given M form the matrices: $M \cdot M^T$; $M^T \cdot M$.
- These are *square matrices* and they are *symmetric*.
- All of their *eigenvalues* are *Real numbers*.
- Their *spectra* are "*identical*": $(\lambda_1, \lambda_2, \dots, \lambda_i, \dots, \lambda_n)$ $(\lambda_i \leq \lambda_{i+1})$

S : $s_{ij} = 0$ if $i \neq j$; $s_{ii} = \sqrt{\lambda_i}$.

U : $k'th$ *column* an *eigenvector* of $M \cdot M^T$ with λ_k.

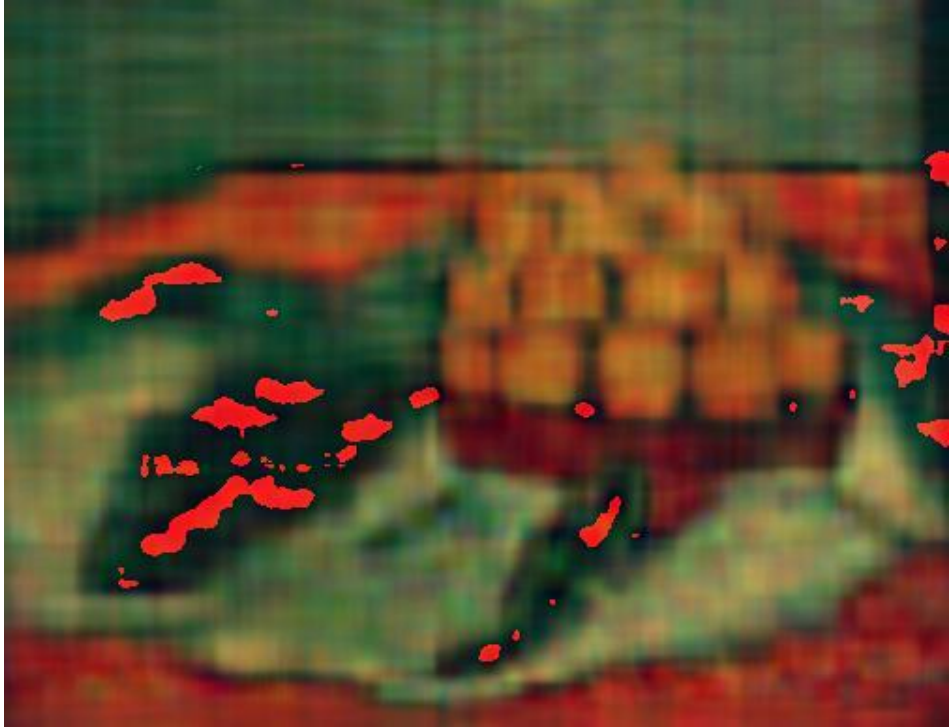
V : $k'th$ *row* an *eigenvector* of $M^T \cdot M$ with λ_k.

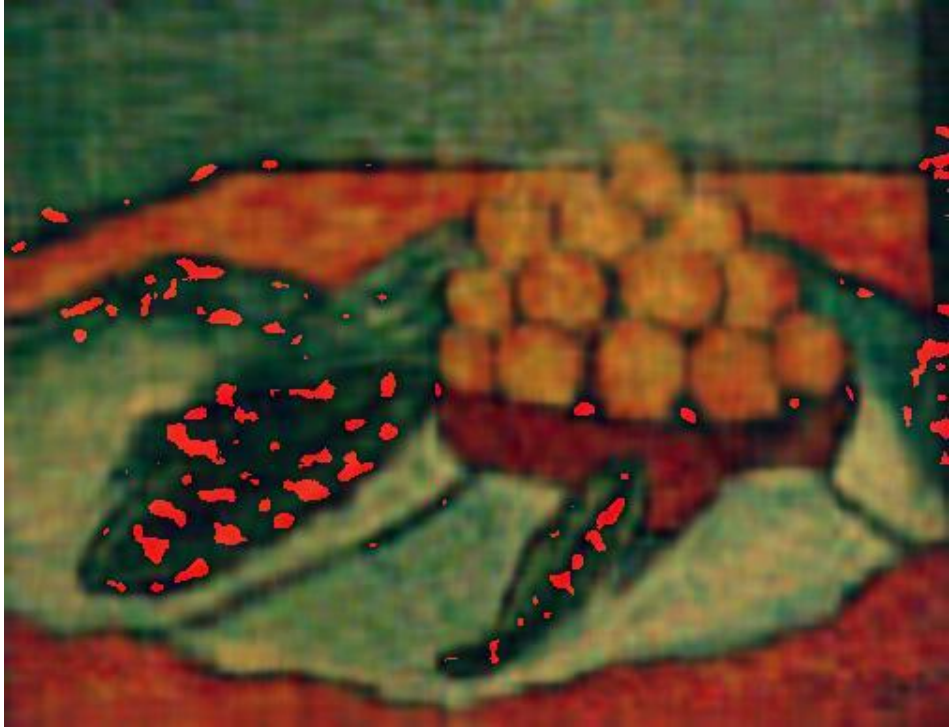
Choosing what to “throw away”

- We know that $M = U \cdot S \cdot V$.
- We also know that S is formed from eigenvalues of $M \cdot M^T$.
- But “*very small*” eigenvalues aren’t going to “*make much difference*” to the matrix product outcome.
- So *to reduce space*:

  “*throw away*” all but the k *largest values* in S

  “*throw away*” the matching eigenvectors in U and V
- and the result with our original image?
- with 10, 20, 50, 100 (instead of 388) we obtain …





Summary

- The *last image* uses space ~100Kb ie ~0.1Mb.
- The *original image* required ~0.5Mb.
- The technique of using a decomposition into three matrices that was described is called *Singular Value Decomposition* (SVD).
- Both Python (via numpy) and Java (via JAMA) provide packages to compute SVD decompositions.
- A Java *implementation* of *Image Compression* (needs JAMA) is at: www.csc.liv.ac.uk/~ped/COMP116/TUTORIALS/ImageCompression.java
- SVD will be seen again in *Machine Learning* (advanced AI module) and in *Data Science*.
- Its presentation concludes our discussion of *Spectral Methods in CS*: these will be seen again later in the programme.