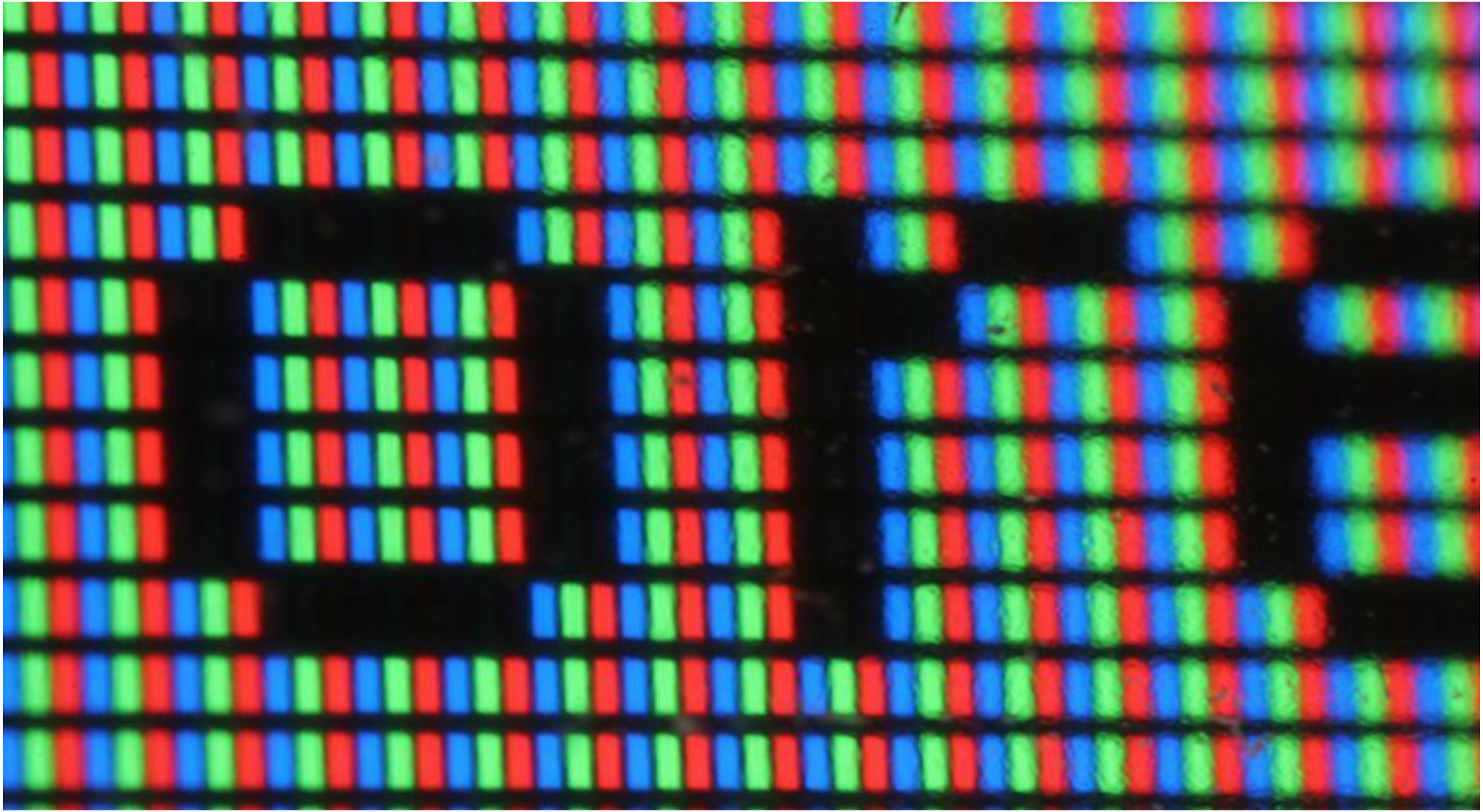
SVD Image Compression


PAA
IBM

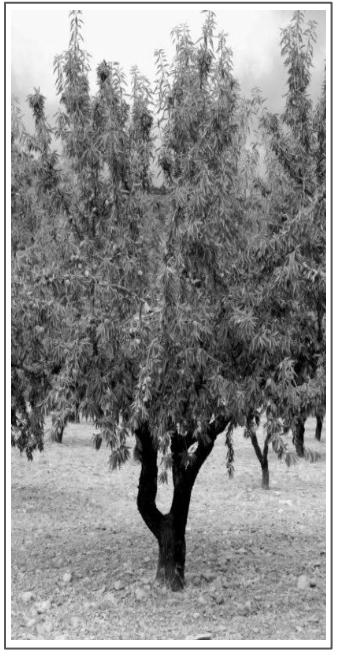

Maxtor
CAUTION


SanDisk
Ultra
64GB
micro SD XC I



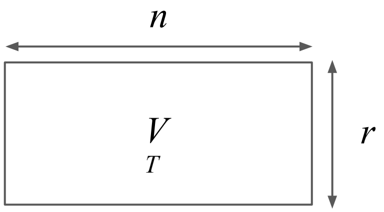
n

$A = m$



$= (m \cdot n)$ *bytes*

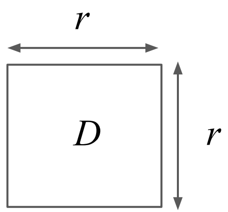
V is an *n x r* column orthonormal matrix, the rows of $\mathbf{V}^{\mathbf{T}}$ are also orthonormal.



A^TA is symmetric, so has an orthonormal basis of *eigenvectors* v_i

Use these *eigenvectors* to form ***V***

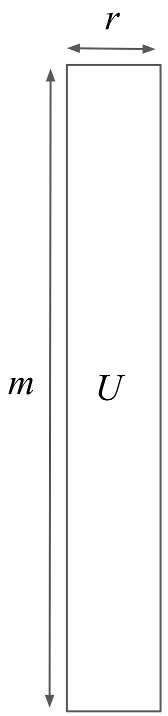
D is a diagonal matrix, with the sorted singular values on the diagonal and all other entries are zero.

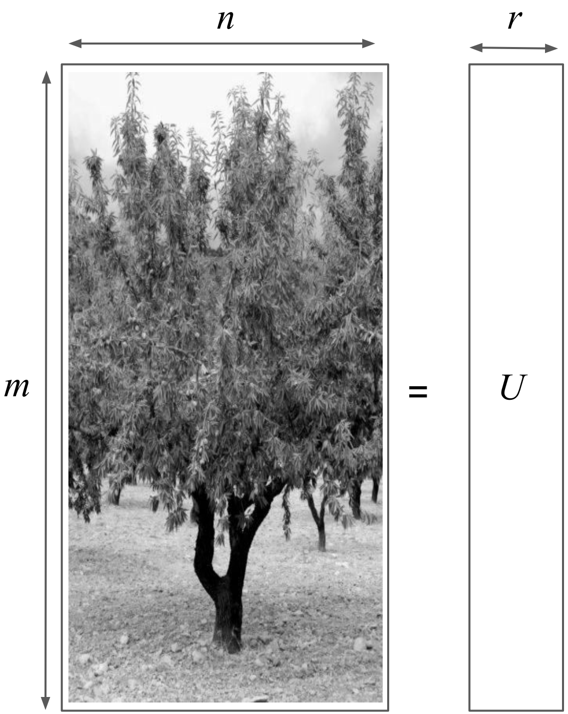


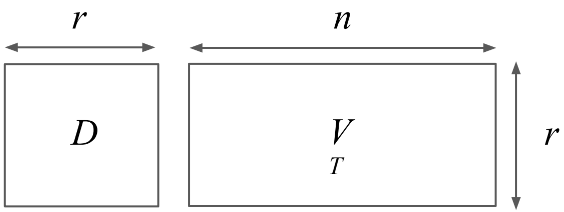
λ_i denotes the *eigenvalues* of the above *eigenvectors*
Singular values denoted by $\sigma = \sqrt{\lambda}$
Remaining columns padded with zeros

U is a column orthonormal matrix, with s_i as its columns

- s_i denotes $\frac{1}{\sigma_n} A v_n$
- In the case where we need *n* columns to for an orthonormal basis for $\mathbb{R}^n$
  - Append columns of ***I*** and find pivots
  - s_i are already independent, keep them and some ***I*** columns
  - GS to find orthogonal vectors, unitize to form orthonormal







A is a 400 x 600 image.

U is a 400 x 400 matrix.

$\boldsymbol{V^T}$ is a 600 x 600 matrix.

D is a 400 x 600 matrix.

Has 200 columns of zeros

SVD states that $\boldsymbol{A} = \boldsymbol{UDV^T}$

$$\boldsymbol{UD} \rightarrow \sigma_i s_i = \sigma_i \frac{Av_i}{\sigma_i} = Av_i$$

A matrix multiplied by a column vector is a column vector

Diagonal entries of $\boldsymbol{D}$ essentially scale the columns of $\boldsymbol{U}$

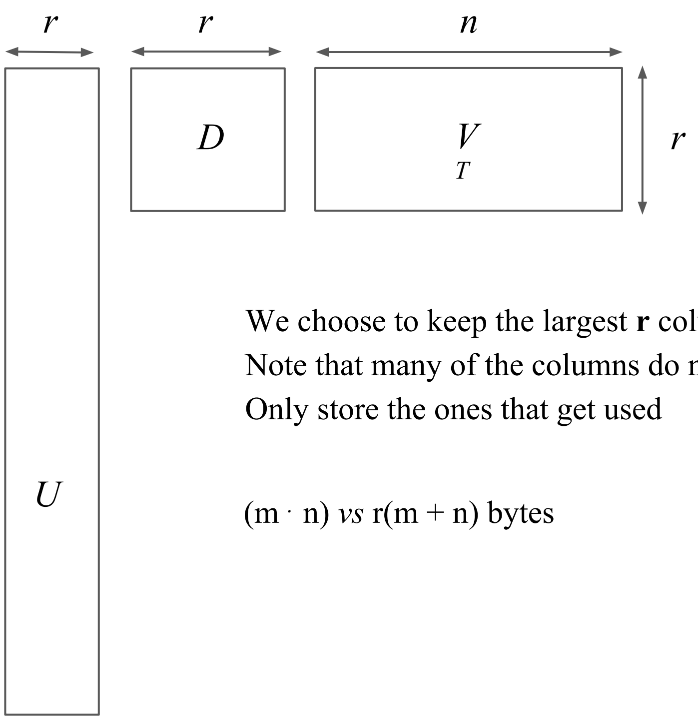
$$\boldsymbol{UDV^T} \rightarrow Av_i v_j^T$$

Note that we are now multiplying by row vectors

The eigenvectors, v_i came from a symmetric matrix, so they form an orthonormal basis

$$\boldsymbol{A} = \boldsymbol{UDV^T} = \sigma_i s_i v_i^T + \ldots + \sigma_r s_r v_r^T$$

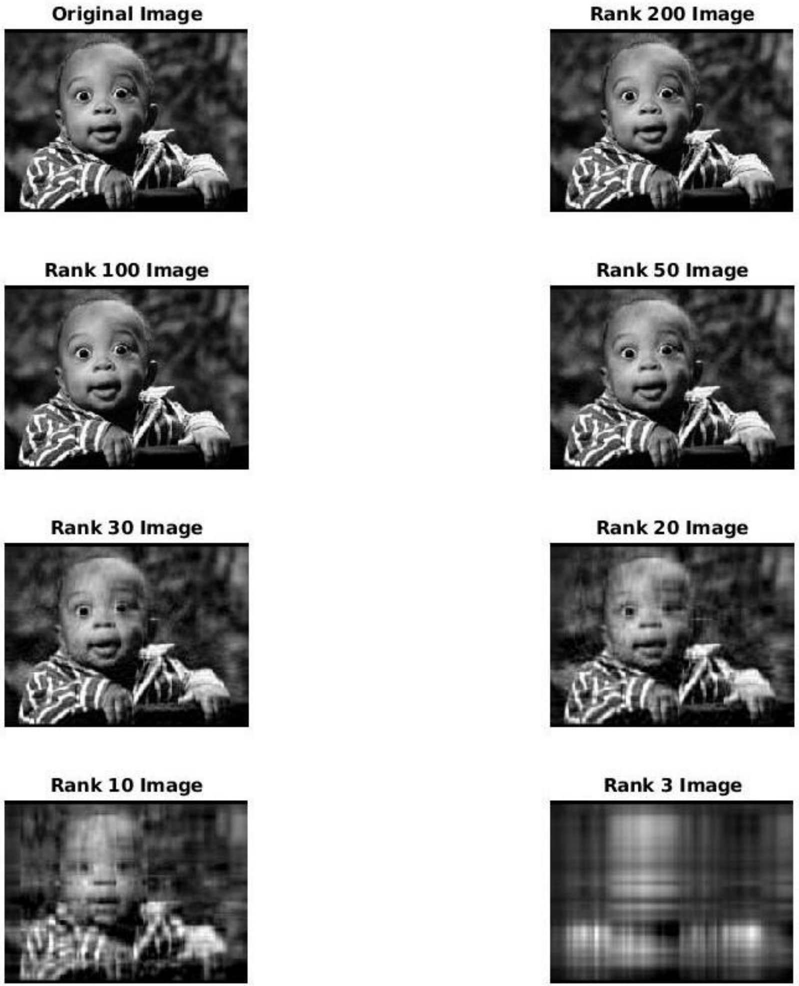
How is this compression?

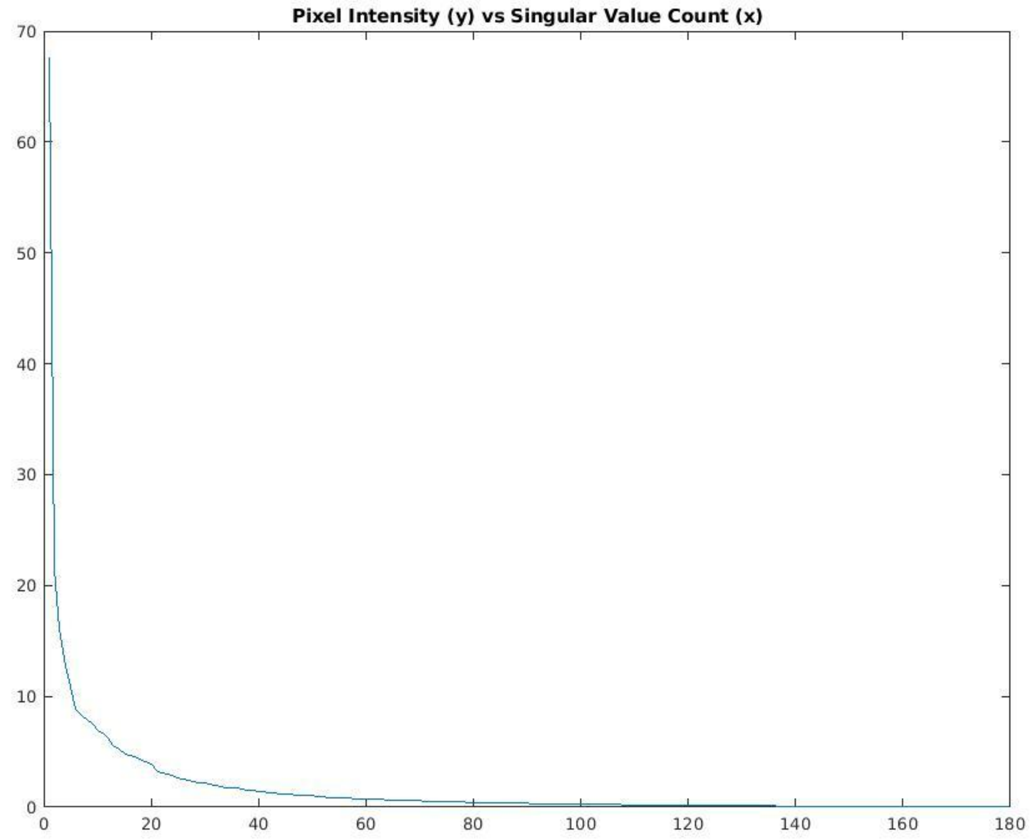


We choose to keep the largest **r** columns from ***D,*** set the rest to zero
Note that many of the columns do not get used as **r** changes, they are multiplied by zero
Only store the ones that get used

$(m \cdot n)$ *vs* $r(m + n)$ bytes

Original Image


Original Image
Rank 200 Image
Rank 100 Image
Rank 50 Image
Rank 30 Image
Rank 20 Image
Rank 10 Image
Rank 3 Image


Pixel Intensity (y) vs Singular Value Count (x)
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180

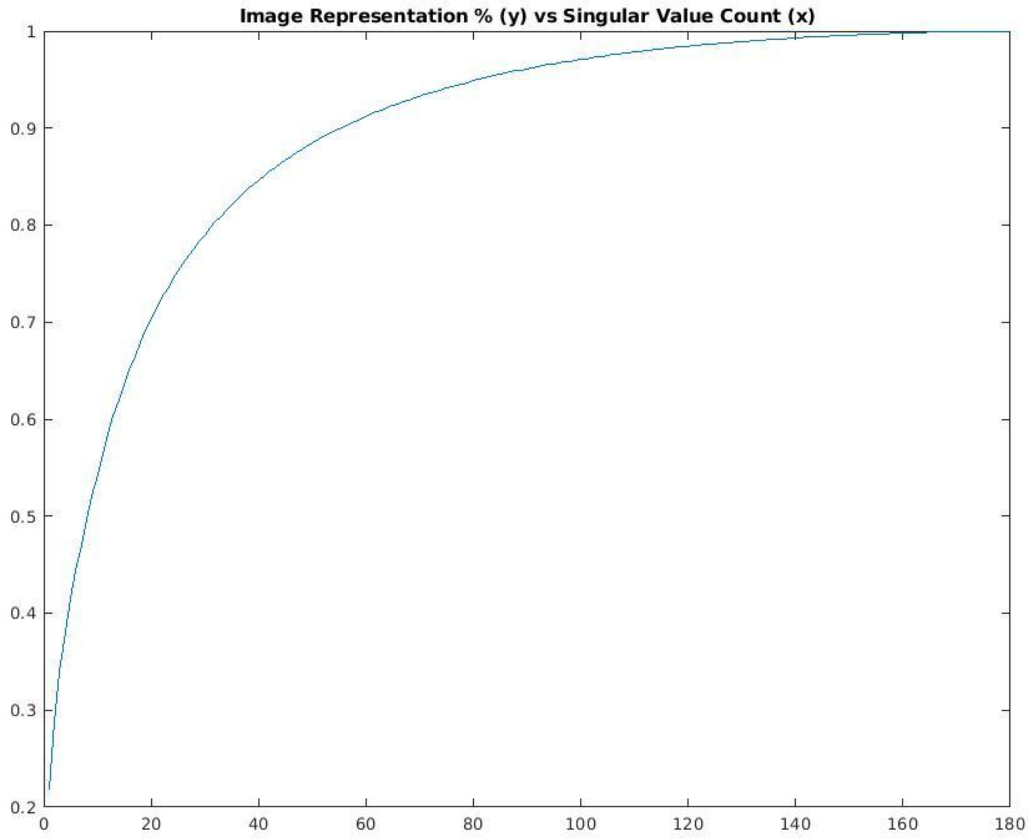


Image Representation % (y) vs Singular Value Count (x)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140
160
180