Compression of Large Images via the Random Singular Value Decomposition

Thomas Anzalone and Elijah Sanderson

Motivation

The SVD is a powerful tool in data storage because it only requires a fraction of the singular values be stored as a low-rank approximation of the data matrix that preserves a very high percentage of the data structure.

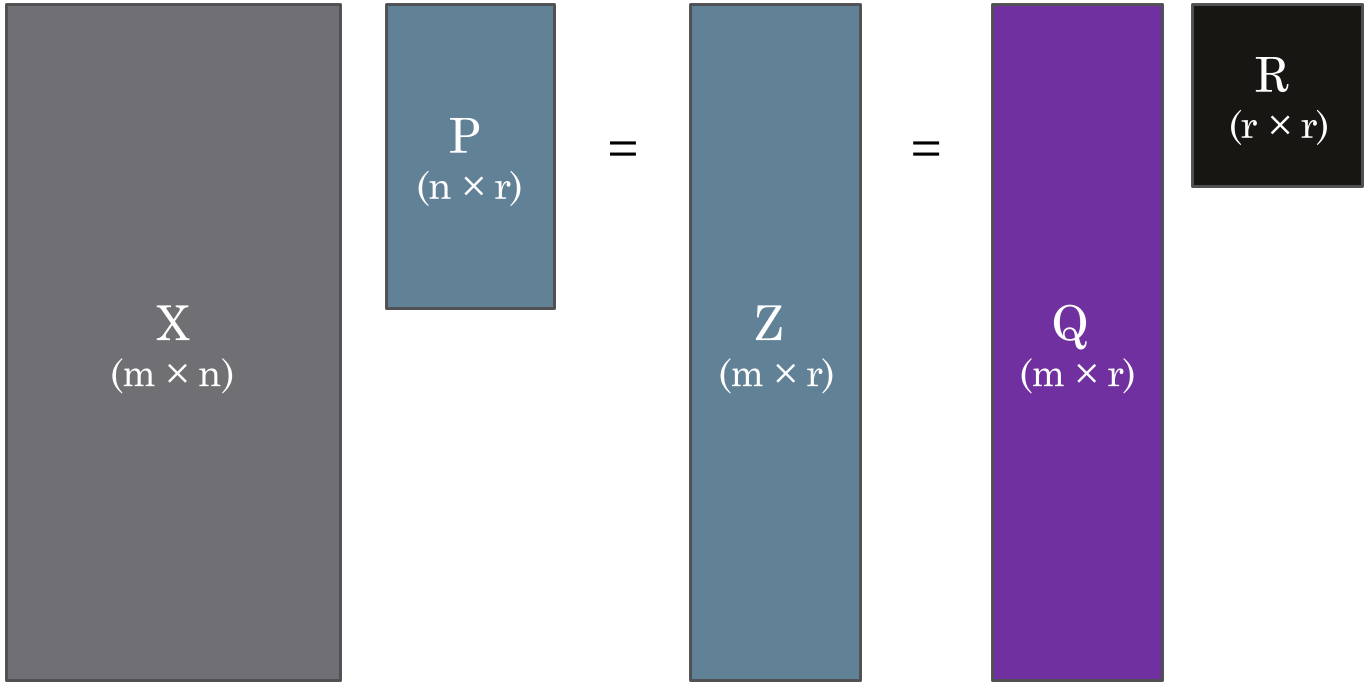
While this "truncated" SVD is very useful, as our dataset grows, the computational complexity increases greatly.

An emerging topic in Randomized Numerical Linear Algebra – the Random SVD (rSVD) – can significantly improve the SVD's cost.

How much of an improvement? Does this reduce the quality?

The Algorithm

Step 1: Random Projection and QR Decomposition



Code:

```
P = randn(n,r);

Z = X*P;

[Q,~] = qr(Z,0);
```

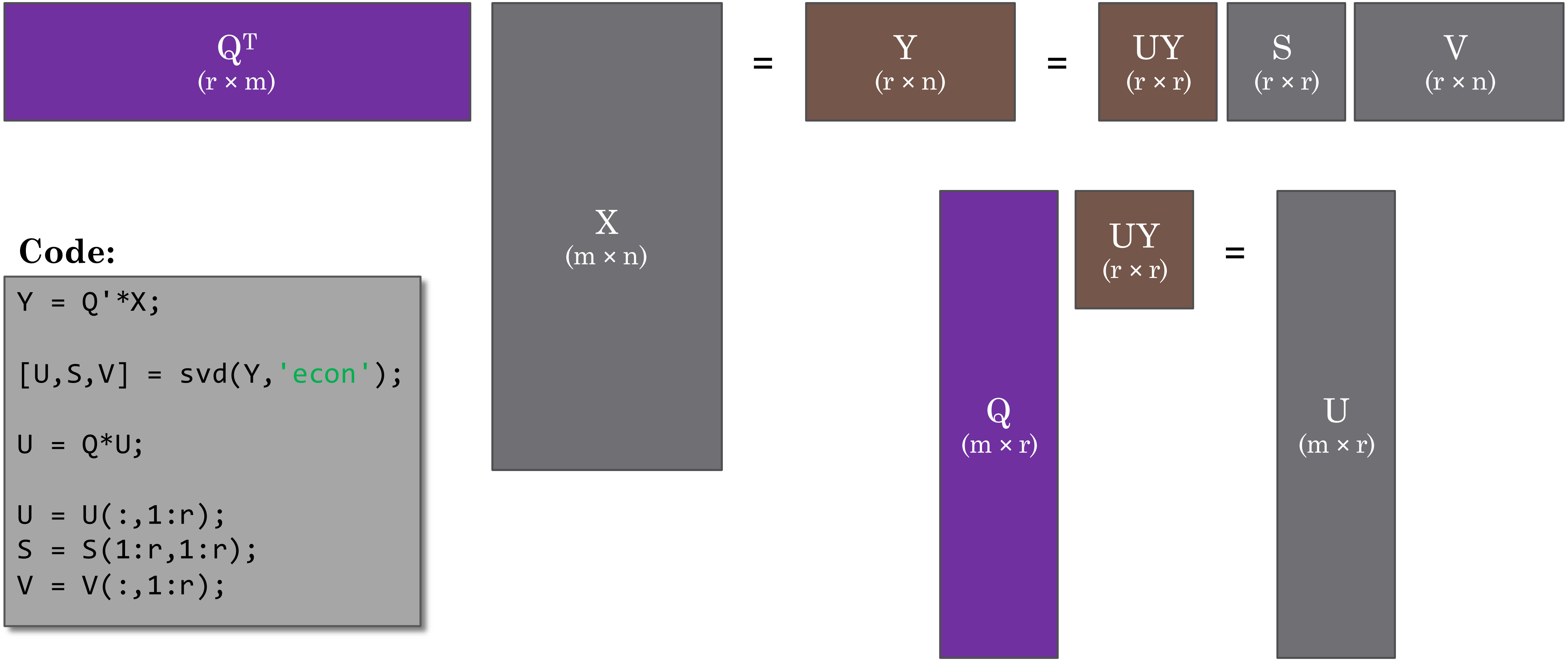
Note if X is not tall: (m < n)

$$\boldsymbol{X^T = (USV^T)^T = VSU^T}$$

Credit: Halko, N., Martinsson, P.-G., & Tropp, J. A.

The Algorithm

Step 2: Project into the Orthogonal Basis and Compute SVD

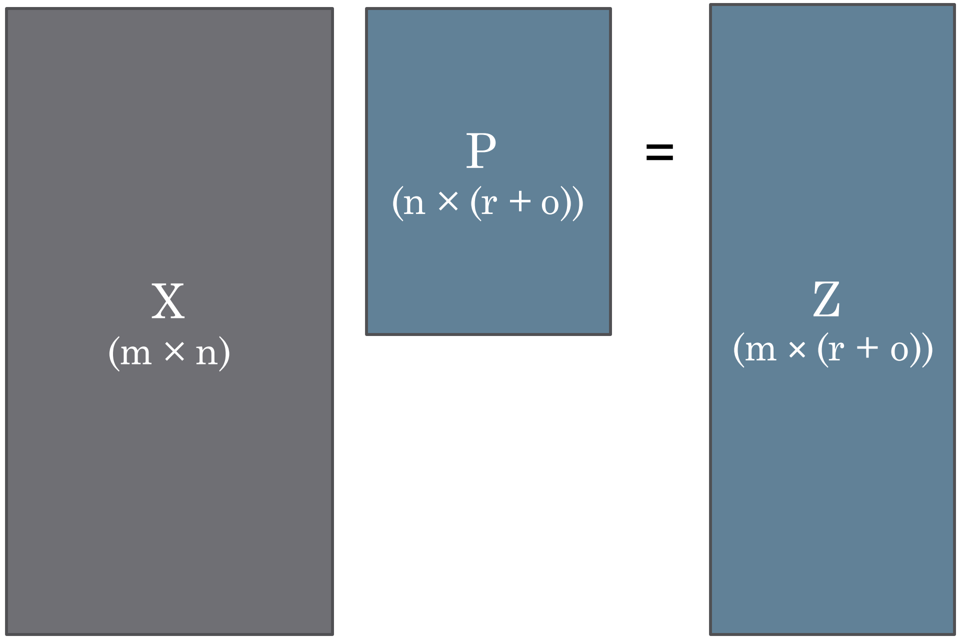


Code:

```
Y = Q'*X;

[U,S,V] = svd(Y,'econ');

U = Q*U;

U = U(:,1:r);
S = S(1:r,1:r);
V = V(:,1:r);
```

Tuning Parameters for Optimization:
Oversampling

- Suppose the target rank for the SVD is **`r = 100`**.
- If we create the Random Projection matrix with a few extra columns, we can reduce some of the error with relatively small additional cost.
- **`P = randn(n, r+o)`** (oversampling parameter "o"; ideally a small number < 25)



Tuning Parameters for Optimization:

Power Iterations

- Prior to taking the QR decomposition, we can transform the data matrix: $\boldsymbol{Z^{(q)} = (X \times X^T)^q \times Z}$
- Instead of computing the very expensive $\boldsymbol{(X \times X^T)^q}$, we can just multiply $\boldsymbol{Z}$ by $\boldsymbol{X^T}$ and then $\boldsymbol{X}$, $\boldsymbol{q}$ times: $\boldsymbol{Z^{(q)} = X \times (X^T \times Z^{(q-1)})}$.
- This significantly improves the singular value decay of a data matrix whose low-rank approximation requires more than a desirable number of singular values.
- Think: $\boldsymbol{\sigma_i > \sigma_j \rightarrow \sigma_i^q \gg \sigma_j^q}$

Recall: Even if we have $\boldsymbol{(X \times X^T)}$ stored in memory:

- $\boldsymbol{(X \times X^T) \times Z \sim O(m^2 n)}$
- $\boldsymbol{X \times (X^T \times Z) \sim O(mnr)}$

Deterministic SVD (truncated with `k = 300`)

Original Image

Truncated SVD

Statistics for Original Image:

```
>> whos X
  Name         Size

  X         6048x4024

  Bytes       Class

24337152    uint8
```

Basic rSVD Algorithm

Truncated SVD

rSVD1:
Basic rSVD algorithm

Time Complexity and Error

```
>> t_det_svd    17.8441
>> err_det_svd   0.0013
```

```
>> t_rsvd1       0.3171
>> err_rsvd1     0.0038
```

Tuned rSVD Algorithm (Oversampling)

Truncated SVD

rSVD2:
Tuned rSVD algorithm with oversampling parameter 20

Time Complexity and Error

```
>> t_det_svd     17.8441
>> err_det_svd    0.0013
```

```
>> t_rsvd1        0.3171
>> err_rsvd1      0.0038
```

```
>> t_rsvd2        0.3573
>> err_rsvd2      0.0035
```

Tuned rSVD Algorithm (Power Iteration)

Truncated SVD

rSVD3:
Tuned rSVD algorithm with oversampling parameter 20 and 1 step power iteration

Time Complexity and Error

```
>> t_det_svd    17.8441
>> err_det_svd   0.0013
```

```
>> t_rsvd1       0.3171
>> err_rsvd1     0.0038
```

```
>> t_rsvd2       0.3573
>> err_rsvd2     0.0035
```

```
>> t_rsvd3       0.5288
>> err_rsvd3     0.0016
```

Comparing the Algorithms

Truncated SVD

Time complexity: #4
Error: #1

```
>> t_det_svd    17.8441
>> err_det_svd   0.0013
```

rSVD1:
Basic rSVD algorithm

Time complexity: #1
Error: #4

```
>> t_rsvd1       0.3171
>> err_rsvd1     0.0038
```

rSVD2:
Tuned rSVD algorithm with oversampling parameter 20

Time complexity: #2
Error: #3

```
>> t_rsvd2       0.3573
>> err_rsvd2     0.0035
```

rSVD3:
Tuned rSVD algorithm with oversampling parameter 20 and 1 step power iteration

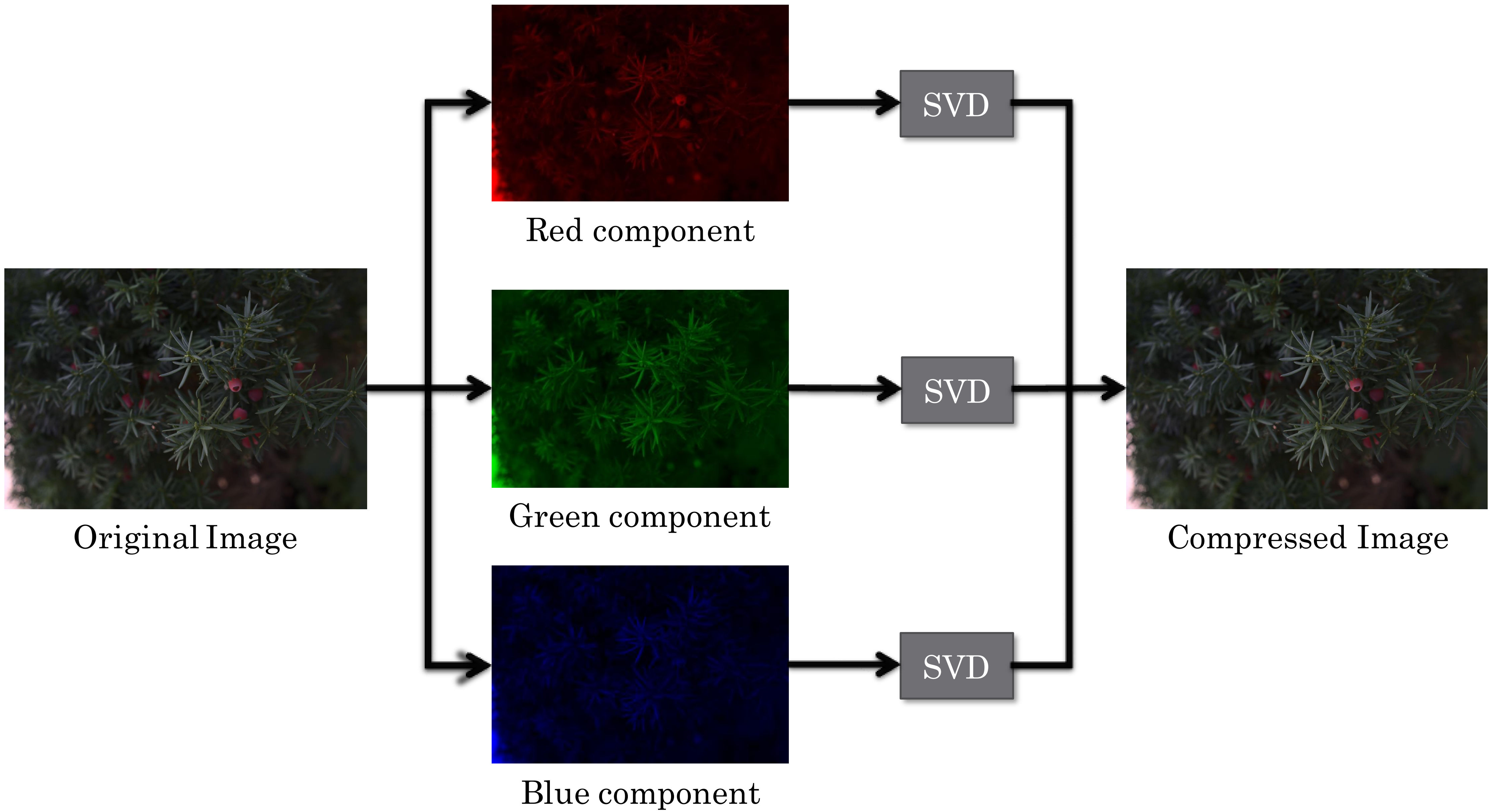
Time complexity: #3
Error: #2

```
>> t_rsvd3       0.5288
>> err_rsvd3     0.0016
```

Note: If the original matrix has intrinsically high rank (e.g. > 1000) then any rSVD algorithm will necessarily lose some of the detail.

Applications to Color Images

- **Construction**
  - Color images are composed of multiple "layers" of data matrices.
  - For example, RGB images contain an $\boldsymbol{m} \times \boldsymbol{n}$ matrix for the red component, an $\boldsymbol{m} \times \boldsymbol{n}$ matrix for the green component, and an $\boldsymbol{m} \times \boldsymbol{n}$ matrix for the blue component.
  - In MATLAB, these are stored as 3-D matrices of dimension $\boldsymbol{m} \times \boldsymbol{n} \times \mathbf{3}$.
- **Intuitive / Natural SVD**
  - Perform an SVD algorithm on each color component matrix and then reconstruct the color image as a 3-D matrix.
  - 3 computations $\Rightarrow$ 3× the time complexity & 3× the error accrued (roughly; depends on the image at hand)
  - When the timing gap between the deterministic SVD and the rSVD is already so large, this makes that gap roughly 3x larger.



SVD
SVD
SVD
Red component
Green component
Blue component
Original Image
Compressed Image

rSVD Applied to RGB (Power Iteration)

Original Image

rSVD3:
Tuned rSVD algorithm with oversampling parameter 20 and 1 step power iteration

Time Complexity and Error

```
>> t_det_svd    63.0543
>> err_det_svd   0.0082
```

```
>> t_rsvd        2.2933
>> err_rsvd      0.0105
```

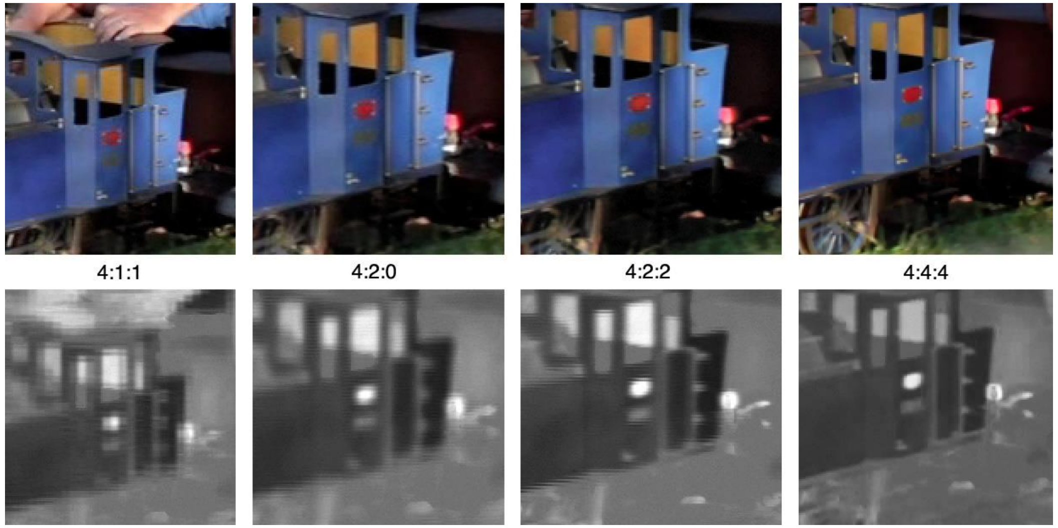
Other Applications of SVD Compression

- **Pseudoinverse**
  - The **pseudoinverse** of a matrix $\boldsymbol{M}$ with SVD $\boldsymbol{M = USV^T}$ is $\boldsymbol{M^\dagger = V\Sigma^\dagger U^T}$ where $\boldsymbol{\Sigma^\dagger}$ is formed by replacing every nonzero diagonal entry by its reciprocal, then transposing this resulting matrix.
- **Total Least Squares Minimization**
  - This seeks a vector $\boldsymbol{x}$ that minimizes the 2-norm of $\boldsymbol{Ax}$ under $\|\boldsymbol{x}\| = \mathbf{1}$; the solution is the right singular vector of $\boldsymbol{A}$ corresponding to the smallest singular value.
- **The Kabasch Algorithm**
  - Computes the optimal rotation matrix (w.r.t. LSM) that aligns a set of points with a corresponding set of points.
  - Major application: comparing structure of molecules.
- **Correlation and Principal Component Analysis (PCA)**
- **Netflix Recommendation Algorithm**
- **Google Page Ranking Algorithm**
- **Denoising (Separation of Noise Subspace from Signal Subspace)**

Other forms of Lossy Image Compression

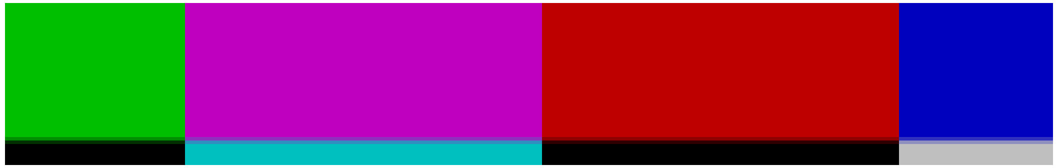
- **Discrete Cosine Transform**
  - Most widely used transformation in signal processing and data compression
  - Capable of achieving compression ratios between **8:1** and up to **100:1** for average quality
  - Used in almost all digital media: JPEG, images, video, streaming, television, cinema, HD video, etc.

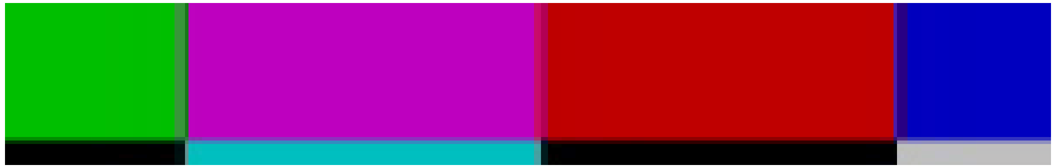
- **Chroma Subsampling**
  - Our eyes perceive changes in **brightness** more than changes in **color**
  - **Chroma subsampling** allocates less data for color information than luminosity information



Difference between four subsampling methods (first row). The second row corresponds to the resolution of the color information
(note that the images in the first row are very similar)

Before (left) and after (right) of color subsampling. Notice the "bleeding" in lightness near the color boundaries





Other forms of Lossy Image Compression

- **Reduction of Color Space**
  - We can compress an image by restricting the colors to only the most common colors, given in the specific **color palette** of the image
  - Each pixel simply references a specific color index in the palette
  - **Posterization** can be avoided by combining this method with **dithering**
    - In this context, **dithering** is intentional noise applied to an image to prevent large patterns such as **color banding**

Grayscale image of David *with dithering*

An image originally in JPEG (24-bit color; 16.7M colors), then posterized by saving to the GIF format (256 colors).

Most obvious in areas of small tonal variation

Sources

- Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288.
- Steven Brunton, J. Nathan Kutz (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. 37–41
- Xinyu Chen (2020). "Intuitive Understanding of Randomized Singular Value Decomposition". Towardsdatascience.com
- Muchahary, D.; Mondal, A. J.; Parmar, R. S.; Borah, A. D.; Majumder, A. (2015). "A Simplified Design Approach for Efficient Computation of DCT". *2015 Fifth International Conference on Communication Systems and Network Technologies*: 483–487.
- Barbero, M.; Hofmann, H.; Wells, N. D. (14 November 1991). "DCT source coding and current implementations for HDTV". *EBU Technical Review*. European Broadcasting Union (251): 22–33.
- S. Winkler, C. J. van den Branden Lambrecht, and M. Kunt (2001). "Vision and Video: Models and Applications". In Christian J. van den Branden Lambrecht (ed.). *Vision models and applications to image and video processing*. Springer. p. 209.
- Kegan Stellato (photography). @keganstellato.