

Báo cáo môn học Tính toán đa phương tiện

Nén và giải nén ảnh bằng thuật toán Discrete Cosine Transform và Singular Value Decomposition

Giảng viên hướng dẫn: Ths. Đỗ Văn Tiến

Lớp: CS232.N21.KHCL

Thành viên:

Phạm Thiện Bảo – 20521107

Lê Nguyễn Tiến Đạt – 20521167



TABLE OF CONTENTS

01

**Giới thiệu
đề tài**

02

**Singular Value
Decomposition**

03

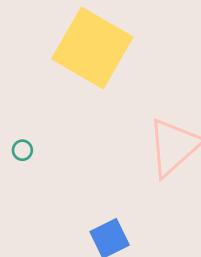
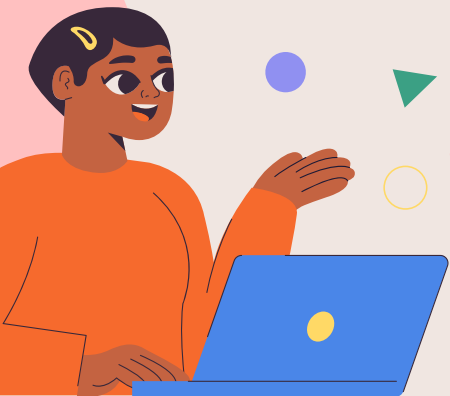
**Discrete Cosine
Transform**

04

Đánh giá

05

Demo



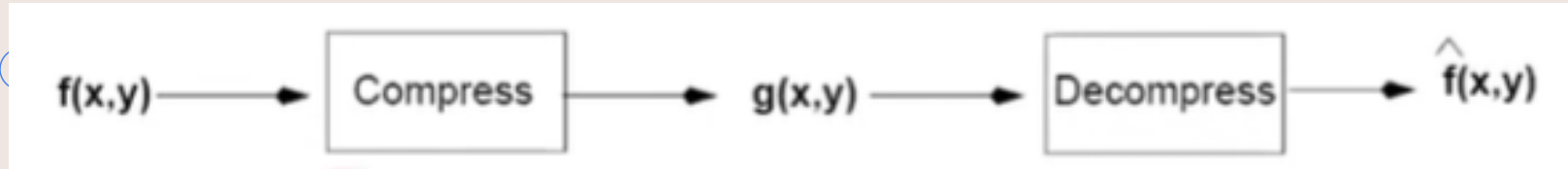
01

Giới thiệu đề tài



1. Giới thiệu đề tài

- Nén dữ liệu (Data compression) là quá trình giảm kích thước dữ liệu để tiết kiệm không gian lưu trữ.
- Có 2 hình thức nén:
 - + Nén bảo toàn thông tin (Lossless Compression)
 - + Nén không bảo toàn thông tin (Lossy Compression)



02

Singular Value Decomposition

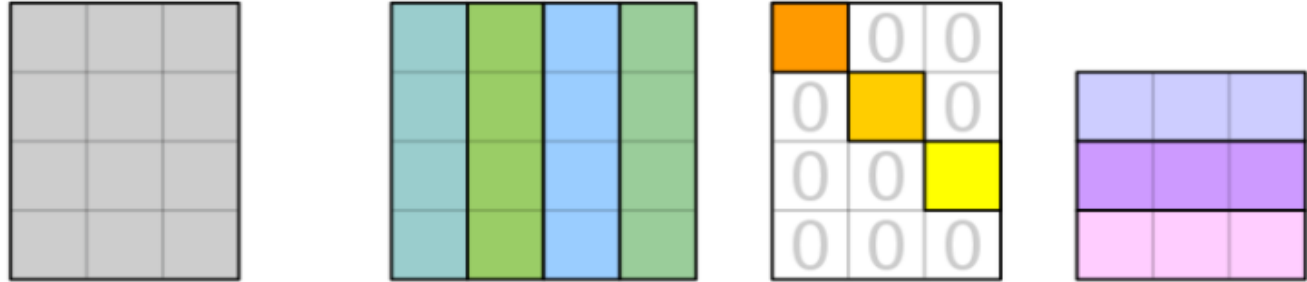


Định nghĩa SVD

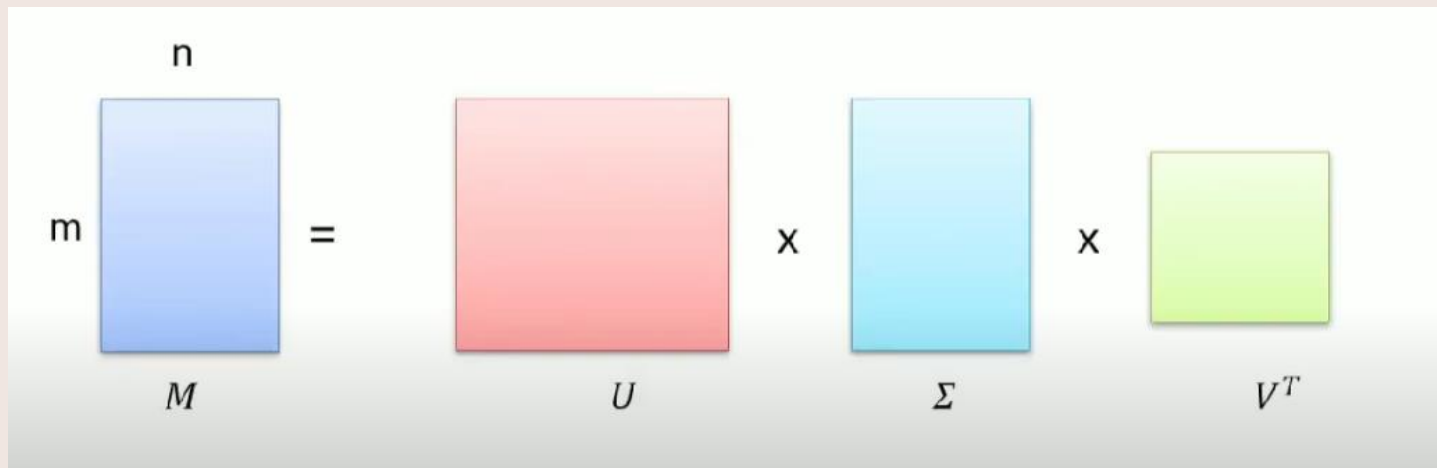


- SVD (Singular Value Decomposition) là một phương pháp phân tích ma trận, có nhiều ứng dụng như nén ảnh, phát hiện ảnh giả mạo, nhận diện khuôn mặt...
- Điểm đặc biệt của SVD là nó có thể áp dụng được trên bất kỳ ma trận nào.

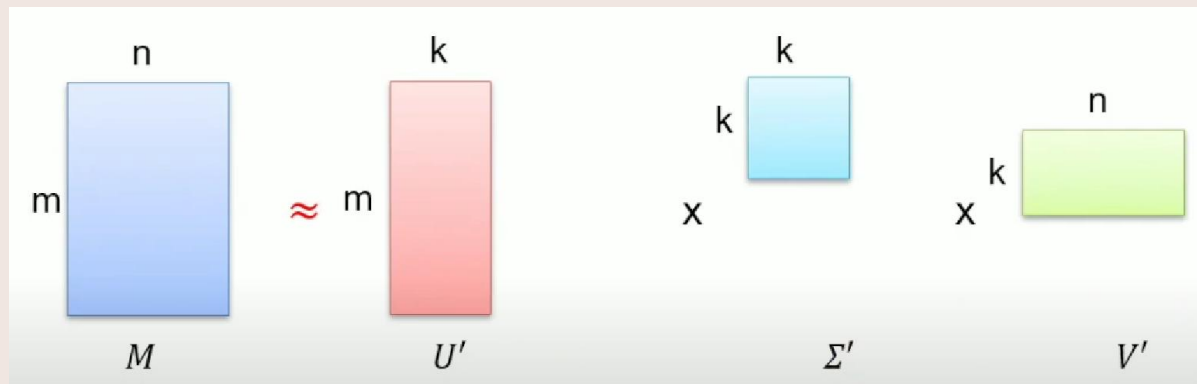
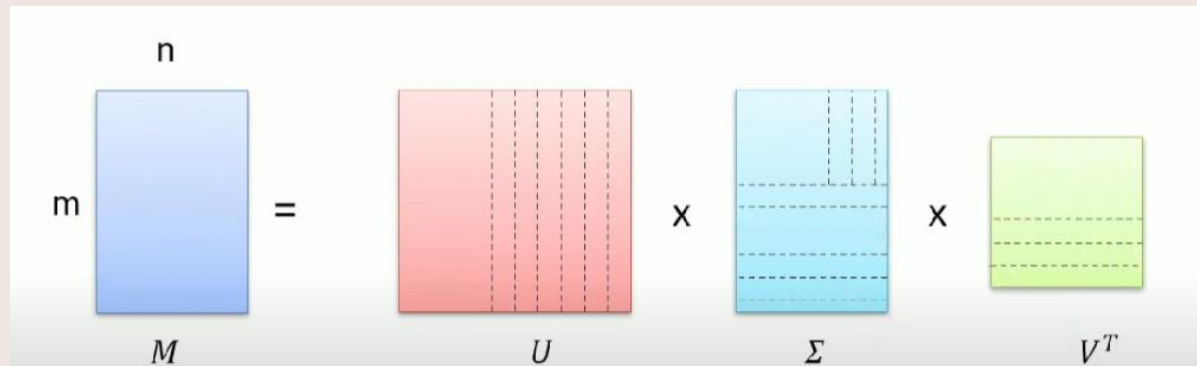
- + Một ma trận A có m hàng và n cột ($n \leq m$).
- + Phép phân tích SVD sẽ phân tích ma trận A thành 3 ma trận \mathbf{U} , \mathbf{S} , \mathbf{V}^t .


$$\begin{matrix} \mathbf{M} \\ m \times n \end{matrix} = \begin{matrix} \mathbf{U} \\ m \times m \end{matrix} \begin{matrix} \mathbf{\Sigma} \\ m \times n \end{matrix} \begin{matrix} \mathbf{V}^* \\ n \times n \end{matrix}$$

Cơ sở cho việc áp dụng SVD vào nén ảnh



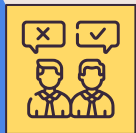
Cơ sở cho việc áp dụng SVD vào nén ảnh



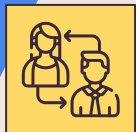
Các bước thực hiện nén ảnh màu bằng SVD



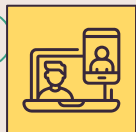
01 Đọc ảnh đầu vào



02 Chuyển đổi ma trận 3 chiều thành 3 ma trận 2 chiều tương ứng với 3 kênh màu



03 Chuyển đổi ảnh màu RGB thành ma trận 3 chiều (Red, Green, Blue)



04 Áp dụng SVD cho mỗi ma trận 2 chiều tương ứng. SVD sẽ phân tích các ma trận thành 3 ma trận U , S , V



05 Áp dụng k trong ma trận U , S , V



06 Kết hợp các ma trận 2 chiều của từng kênh để tạo thành ma trận 3 chiều mới



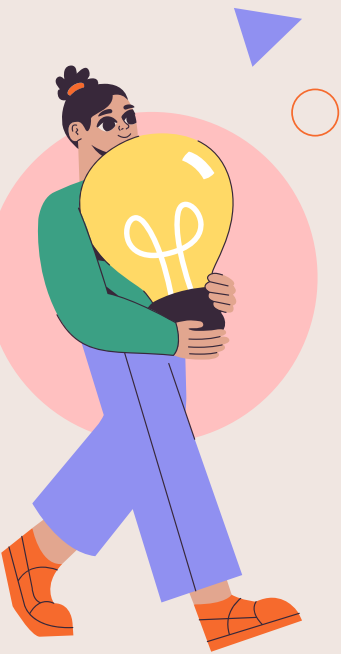
07 Chuyển đổi ma trận 3 chiều thành ảnh màu nén

Cơ sở cho việc áp dụng SVD vào nén ảnh

- Khi cần nén 1 ảnh A có kích thước $(m \times n)$ ta chỉ sử dụng 1 số ít k phần tử và loại bỏ đi các phần tử còn lại.
- Tổng dung lượng của ảnh sau khi nén sẽ là $kn + k + km = k(n + m + 1)$
- Tổng dung lượng của ảnh gốc sẽ là: $m * n$.
- Hệ số nén : $r = \frac{kn + k + km}{m \times n} = \frac{k(n + m + 1)}{m \times n}$
- Xét 2 trường hợp:

$$+ r = 1 \text{ (không nén)}: k = \frac{n \times m}{(m + n + 1)}$$

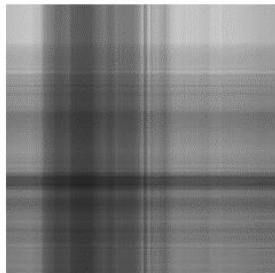
$$+ r = 0.5 \text{ (nén 1 nửa)}: k = \frac{n \times m}{2(m + n + 1)}$$



Chất lượng ảnh khi thay đổi k

Compression at different orders

K = 1



K = 5



K = 10



K = 20



K = 50



K = 80



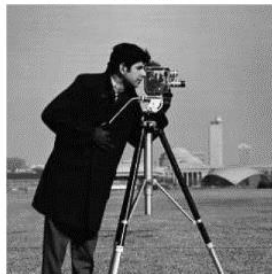
K = 100



K = 150



K = 200



K = 250



03

Discrete Cosine Transform



Lý do ra đời:

- Giá trị cường độ ít khi thay đổi nhiều lần trong một vùng nhỏ của ảnh.
- Khi chuyển bức ảnh sang miền tần số, thông tin chủ yếu nằm ở các vùng tần số thấp, tần số cao tương ứng với nhiễu hoặc không mang lại nhiều thông tin.
- Mắt người khó phát hiện mất mát ở vùng tần số cao.

Chia nhỏ từng khối => Biến đổi về miền tần số => loại bỏ bớt vùng tần số cao, giữ lại tần số thấp => Biến đổi lại về miền không gian.

Ý tưởng:

- Chia dữ liệu thành các khối nhỏ có kích thước đồng nhất (8x8 pixel).
- Áp dụng biến đổi DCT lần lượt lên từng khối dữ liệu.
- Thực hiện lượng tử hóa cho từng khối.
- Để tái tạo lại dữ liệu gốc, áp dụng phép biến đổi nghịch DCT.



Công thức DCT:

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^7 \sum_{y=0}^7 p(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \cos\left[\frac{(2y+1)j\pi}{16}\right]$$

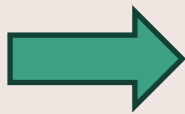
$$T_{ij} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N}\right] & \text{if } i > 0 \end{cases}$$

Khởi tạo ma trận DCT

$$T = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

-128



$$M = \begin{bmatrix} 26 & -5 & -5 & -5 & -5 & -5 & -5 & 8 \\ 64 & 52 & 8 & 26 & 26 & 26 & 8 & -18 \\ 126 & 70 & 26 & 26 & 52 & 26 & -5 & -5 \\ 111 & 52 & 8 & 52 & 52 & 38 & -5 & -5 \\ 52 & 26 & 8 & 39 & 38 & 21 & 8 & 8 \\ 0 & 8 & -5 & 8 & 26 & 52 & 70 & 26 \\ -5 & -23 & -18 & 21 & 8 & 8 & 52 & 38 \\ -18 & 8 & -5 & -5 & -5 & 8 & 26 & 8 \end{bmatrix}$$

Biến đổi DCT theo công thức:

$$D = TMT'$$

$$D = \begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix}$$

Ma trận lượng tử:

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

- Giảm số bit trên một mẫu giá trị.
- Với **level** = n , ma trận lượng tử 8×8 được tính như sau :
 $Q(n) = Q_{50} * (100 - n) / 50$ với $n \geq 50$
 $Q(n) = Q_{50} * (50 / n)$ với $n < 50$

Lượng tử hóa:

$$C_{i,j} = \text{round}\left(\frac{D_{i,j}}{Q_{i,j}}\right)$$

$$C = \begin{bmatrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Giải nén:

$$R_{i,j} = Q_{i,j} \times C_{i,j}$$



$$N = \text{round}(T' R T) + 128$$



Decompressed =

149	134	119	116	121	126	127	128
204	168	140	144	155	150	135	125
253	195	155	166	183	165	131	111
245	185	148	166	184	160	124	107
188	149	132	155	172	159	141	136
132	123	125	143	160	166	168	171
109	119	126	128	139	158	168	166
111	127	127	114	118	141	147	135

04 Đánh giá



RMSE



- RSME (Root Mean Square Error): là phép đo sử dụng để đánh giá sự khác biệt của ảnh gốc và ảnh sau khi nén.
 - + RMSE thấp cho thấy sự tương đồng cao giữa hai ảnh và chất lượng nén tốt.
 - + Một giá trị RMSE cao cho thấy sự khác biệt lớn giữa hai ảnh và chất lượng nén kém.
- Công thức tính RMSE:

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

PSNR







- PSNR (Peak Signal-to-Noise Ratio) là một đại lượng được sử dụng để đánh giá mức độ chất lượng của ảnh sau khi nén so với ảnh gốc.
 - + PSNR cao cho thấy sự tương đồng cao giữa hai ảnh và chất lượng nén tốt.
 - + PSNR thấp cho thấy sự khác biệt lớn giữa hai ảnh và chất lượng nén kém.
- Công thức tính PSNR:

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$



The PSNR is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \end{aligned}$$

Kết quả thuật toán SVD :

Image	Size	K	Time(s)	RMSE	PSNR	Hiệu suất nén (%)
	(256, 256, 3)	10	0.77	28.07	19.17	4
		50	1.05	12.05	26.51	20
		100	1.02	3.38	37.56	39
		200	1.37	0.23	61.08	78
	(512, 512, 3)	10	7.49	18.73	22.68	2
		50	7.35	8.12	29.94	10
		100	7.68	4.73	34.64	20
		200	8.58	2.29	40.94	39
		400	8.75	0.44	55.17	78
	(400, 615, 3)	10	5.49	20.29	21.98	2
		50	5.8	10.92	27.37	10
		100	6.76	6.65	31.67	21
		200	6.32	2.61	39.81	41
		300	5.96	0.79	50.17	62
	(1024, 1024, 3)	50	63.2	12.34	26.3	5
		100	61.9	7.53	30.59	10
		400	67.12	1.29	45.93	39
		750	77.35	0.02	83.91	73

Kết quả thuật toán DCT :

Image	Size	level	Time(s)	RMSE	PSNR	Hiệu suất nén(%)
	(256, 256, 3)	10	0.96	12.38	26.21	66
		30	0.83	8.29	29.69	47
		50	1.12	6.68	31.57	36
		90	0.87	2.57	39.85	6
	(512, 512, 3)	10	3.33	8.14	29.91	56
		30	3.02	5.58	33.20	41
		50	3.11	4.80	34.51	33
		90	3.21	2.71	39.59	7
	(400, 615, 3)	10	2.94	27.09	19.47	46
		30	2.86	25.68	19.94	31
		50	2.97	25.29	31.67	23
		90	2.94	20.6	20.31	4
	(1024, 1024, 3)	10	12.26	7.11	31.09	31
		30	12.44	4.25	35.56	17
		50	12.22	3.32	37.71	8
		90	12.63	0.82	49.83	5

Nhận xét:

- Khi thay đổi level cho ma trận lượng tử, thời gian chạy của thuật toán DCT không có sự thay đổi nhiều.
- Khi thay đổi giá trị k càng cao cho SVD thì thời gian chạy càng cao.
- **RMSE** của thuật toán SVD có giá trị **nhỏ** với **k càng lớn** và RMSE của DCT cũng **nhỏ** khi **level cao**.
- **PSNR** của thuật toán SVD có giá trị **lớn** với **k càng lớn** và PSNR của DCT cũng lớn khi **level cao**.
- Với mỗi bức hình, SVD có giá trị **PSNR cao** hơn hẳn DCT (Hình 4: PSNR = 83.91)





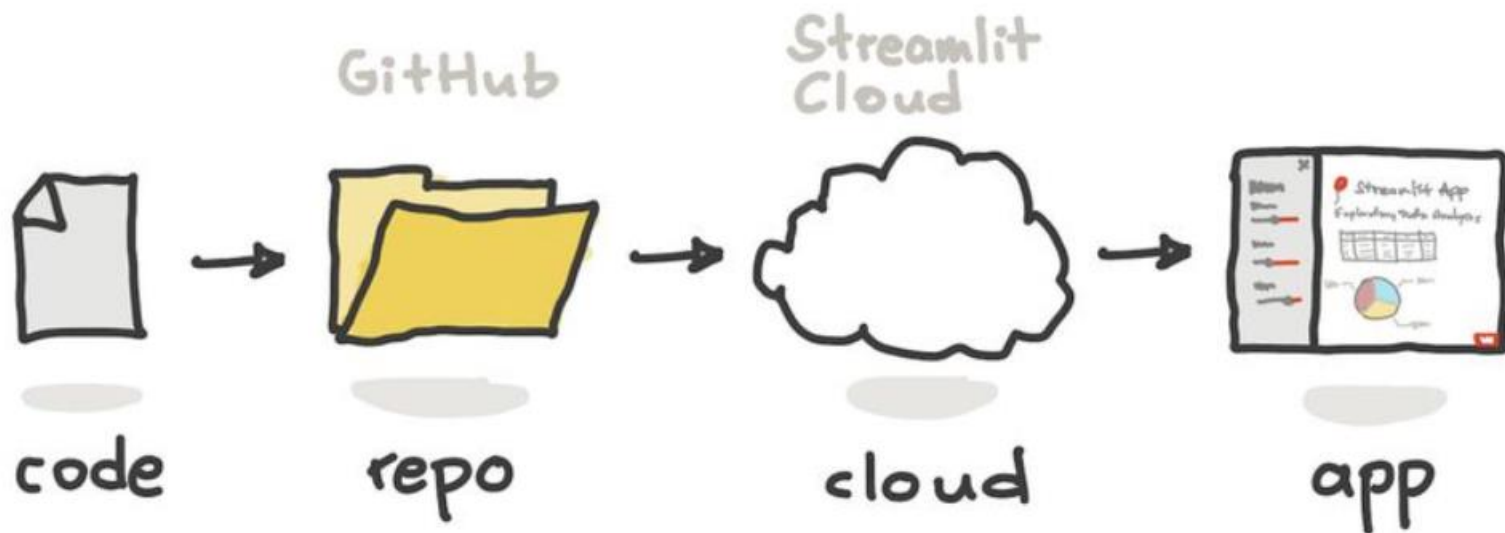
05

Demo

Streamlit



Với Streamlit Cloud, bạn có thể đẩy mã nguồn ứng dụng Streamlit của mình lên đám mây và tạo ra một liên kết trực tuyến để chia sẻ ứng dụng với người khác.



Phân công công việc:

Họ và tên	MSSV	Công việc
Phạm Thiện Bảo	20521107	Thuật toán DCT Xây dựng demo bằng streamlit
Lê Nguyễn Tiến Đạt	20521167	Thuật toán SVD

Thank you for listening !

