

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH



BÁO CÁO ĐỒ ÁN
MÔN TÍNH TOÁN ĐA PHƯƠNG TIỆN

Đề tài: Thực hiện nén và giải nén kiểu dữ liệu ảnh bằng
thuật toán Discrete Cosine Transform và
thuật toán Singular Value Decomposition

Giảng viên hướng dẫn: ThS. Đỗ Văn Tiến

Lớp: CS232.N21.KHCL

Sinh viên thực hiện:

- | | |
|-----------------------|----------------|
| 1. Phạm Thiện Bảo | MSSV: 20521107 |
| 2. Lê Nguyễn Tiên Đạt | MSSV: 20521167 |

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

....., ngày.....tháng.....năm 20...

Người nhận xét

(Ký tên và ghi rõ họ tên)

BẢNG PHÂN CÔNG, ĐÁNH GIÁ THÀNH VIÊN

Bảng 1. Bảng phân công, đánh giá thành viên

Họ và tên	MSSV	Phân công	Đánh giá
Phạm Thiện Bảo	20521107	<ul style="list-style-type: none">- Nghiên cứu và viết code cài đặt thuật toán DCT cho nén và giải nén ảnh.- Xây dựng giao diện và ứng dụng web demo bằng Streamlit.- Làm slide báo cáo.- Viết báo cáo.	- Hoàn thành xuất sắc và đúng tiến độ công việc được giao.
Lê Nguyễn Tiến Đạt	20521167	<ul style="list-style-type: none">- Nghiên cứu và viết code cài đặt thuật toán SVD cho nén và giải nén ảnh.- Làm slide báo cáo.	- Hoàn thành xuất sắc và đúng tiến độ công việc được giao.

MỤC LỤC

LỜI MỞ ĐẦU	5
CHƯƠNG I: TỔNG QUAN	6
I. Tính toán đa phương tiện	6
II. Dữ liệu hình ảnh.....	6
III. Nén và giải nén.....	7
CHƯƠNG II: THUẬT TOÁN	9
I. Discrete Cosine Transform.....	9
1. Giới thiệu	9
2. Ý tưởng thuật toán	9
2.1) Quá trình nén.....	9
2.2) Quá trình giải nén.....	14
3. Áp dụng ý tưởng của thuật toán cho ảnh màu RGB	15
II. Singular Value Decomposition	16
1. Giới thiệu	16
2. Ý tưởng thuật toán	16
2.1) Quá trình nén.....	18
2.2) Quá trình giải nén.....	18
3. Áp dụng ý tưởng của thuật toán cho ảnh màu RGB	18
CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ	20
I. Dataset	20
II. Các độ đo sử dụng.....	20
1. Hiệu suất nén	21
1.1) Cách tính cho thuật toán SVD	21
1.2) Cách tính cho thuật toán DCT	21
2. Mean Squared Error	21
3. Root Mean Square Error	22
4. Peak Signal-to-Noise Ratio	22
III. Kết quả chạy thực nghiệm.....	22
1. Thuật toán DCT	22
2. Thuật toán SVD.....	23
IV. Nhận xét.....	24
CHƯƠNG IV: XÂY DỰNG ỨNG DỤNG WEB.....	25
I. Xây dựng giao diện	25
II. Công cụ hỗ trợ.....	26
TÀI LIỆU THAM KHẢO	27

LỜI MỞ ĐẦU

Trong những năm qua, việc truyền tải và lưu trữ các dữ liệu đa phương tiện có kích thước lớn như hình ảnh, âm thanh và video đã không còn là vấn đề nan giải với các chuyên gia công nghệ thông tin. Nhờ việc áp dụng các thuật toán nén và giải nén, chúng ta có thể làm giảm kích thước tệp và tiết kiệm không gian lưu trữ cũng như băng thông truyền tải. Chính vì tầm quan trọng ấy, việc ước tính hiệu suất của nén và giải nén dữ liệu được quan tâm trong các sản phẩm ứng dụng công nghệ nhằm đảm bảo trải nghiệm tốt nhất cho người dùng.

Thông qua môn học Tính toán đa phương tiện, nhóm chúng em muốn thực hiện đồ án về việc tìm hiểu hai thuật toán nén, giải nén cơ bản là DCT (Discrete Cosine Transform) và SVD (Singular Value Decomposition). Đây là hai thuật toán khá phổ biến và có nhiều ứng dụng trong thực tế. Việc thực hiện đồ án này giúp nghiên cứu chi tiết về ý tưởng hoạt động và hiểu rõ những ưu điểm, khuyết điểm cần cải thiện của hai thuật toán. Không chỉ dừng lại việc cài đặt lại thuật toán, nhóm chúng em còn xây dựng thêm một website nhỏ bằng công cụ Streamlit để minh họa rõ thuật toán và tạo môi trường cho mọi người trải nghiệm thử.

Nhóm xin gửi lời cảm ơn chân thành đến thầy Đỗ Văn Tiến đã tận tình giảng dạy, hướng dẫn chi tiết các lý thuyết cơ bản về tính toán đa phương tiện để nhóm có thể tự tin thực hiện đồ án này. Do kiến thức và thời gian môn học còn hạn chế, đồ án vẫn còn nhiều thiếu sót không mong muốn. Những nhận xét và góp ý của thầy và các bạn chính là động lực to lớn để đồ án được hoàn thiện hơn trong tương lai.

CHƯƠNG I: TỔNG QUAN

I. Tính toán đa phương tiện

- Tính toán đa phương tiện (Multimedia Computing) là một lĩnh vực trong khoa học máy tính và kỹ thuật điện tử liên quan đến xử lý và phân tích dữ liệu đa phương tiện như hình ảnh, âm thanh, video và các dữ liệu đa phương tiện khác. Nó tập trung vào việc phát triển các thuật toán, phương pháp và công nghệ để xử lý, mã hóa, nén, truyền tải, phân tích và trích xuất thông tin từ các dạng dữ liệu đa phương tiện.

- Tính toán đa phương tiện bao gồm nhiều ứng dụng, phổ biến như:

- Nén dữ liệu đa phương tiện.
- Xử lý hình ảnh.
- Xử lý âm thanh.
- Xử lý video.
- Truyền thông đa phương tiện.

II. Dữ liệu hình ảnh

- Dữ liệu dạng ảnh thường được biểu diễn dưới dạng ma trận số, trong đó mỗi phần tử trong ma trận đại diện cho một pixel trong hình ảnh. Cấu trúc và cách biểu diễn dữ liệu ảnh phụ thuộc vào định dạng file ảnh được sử dụng. Dưới đây là một số định dạng phổ biến trong dữ liệu ảnh:

- Định dạng ảnh raster: Đây là định dạng phổ biến nhất cho dữ liệu ảnh. Ảnh raster được biểu diễn bằng cách chia hình ảnh thành các điểm ảnh (pixels) và lưu trữ giá trị màu hoặc mức xám của từng pixel. Các định dạng raster phổ biến bao gồm BMP, JPEG, PNG, TIFF, GIF.
- Định dạng ảnh vector: Định dạng ảnh vector sử dụng các đối tượng hình học (như đường thẳng, đường cong, đa giác) để biểu diễn ảnh. Thay vì lưu trữ từng điểm ảnh như ảnh raster, định dạng vector chỉ lưu trữ các thông tin về các hình học và thuật toán để tái tạo ảnh khi cần thiết. Các định dạng vector phổ biến bao gồm SVG, EPS.
- Định dạng RAW: Định dạng RAW lưu trữ dữ liệu ảnh chưa qua xử lý hoặc nén từ máy ảnh kỹ thuật số. Dữ liệu RAW chứa thông tin đầy đủ về màu sắc và chi

tiết của ảnh, nhưng cần được xử lý sau để tạo ra ảnh chất lượng cao. Các định dạng RAW phổ biến bao gồm NEF (Nikon), CR2 (Canon), ARW (Sony).

III. Nén và giải nén

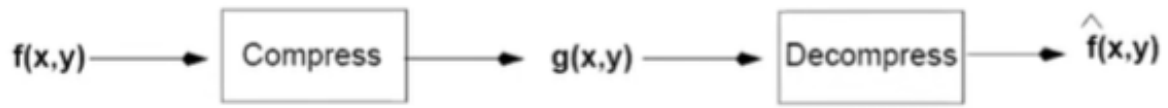
- Nén dữ liệu là quá trình giảm kích thước của tệp đa phương tiện bằng cách loại bỏ dữ liệu không cần thiết hoặc sử dụng các phương pháp nén để biểu diễn dữ liệu một cách hiệu quả hơn. Các thuật toán nén được áp dụng để giảm kích thước tệp mà vẫn giữ lại một lượng thông tin cần thiết. Kết quả của quá trình nén là một tệp nén có thể có kích thước nhỏ hơn tệp gốc.

- Giải nén dữ liệu là quá trình khôi phục dữ liệu từ tệp nén trở lại định dạng ban đầu hoặc gần giống với nó. Các thuật toán giải nén được sử dụng để khôi phục dữ liệu ban đầu từ tệp nén bằng cách sử dụng thông tin lưu trữ trong tệp nén. Kết quả của quá trình giải nén là một tệp có định dạng giống hoặc gần giống với tệp gốc, cho phép khôi phục lại dữ liệu ban đầu mà không mất đi nhiều thông tin quan trọng.

- Có ba loại chính của nén dữ liệu:

- **Nén mất mát (Lossy Compression):** Trong loại nén này, dữ liệu bị mất một phần thông tin không thể khôi phục lại chính xác khi giải nén. Quá trình nén dữ liệu này được thực hiện bằng cách loại bỏ hoặc giảm lượng dữ liệu không cần thiết hoặc không quan trọng để giảm kích thước tệp. Một số ví dụ về nén mất mát là JPEG (nén ảnh), MP3 (nén âm thanh) và MPEG (nén video). Quá trình giải nén chỉ tạo ra một phiên bản gần đúng của dữ liệu ban đầu.
- **Nén không mất mát (Lossless Compression):** Trong loại nén này, dữ liệu được nén mà không mất mát bất kỳ thông tin nào. Quá trình nén và giải nén hoàn toàn đảm bảo tính toàn vẹn của dữ liệu ban đầu. Một số ví dụ về nén không mất mát là ZIP (nén tệp tin), PNG (nén ảnh) và FLAC (nén âm thanh). Tuy nén không mất mát đảm bảo tính toàn vẹn của dữ liệu, nhưng hiệu suất nén thường không tối ưu như nén mất mát.
- **Nén bán mất mát (Lossy Compression with Partial Reconstruction):** Loại nén này kết hợp cả hai phương pháp nén mất mát và nén không mất mát. Nó cho phép mất mát thông tin một cách kiểm soát để giảm kích thước tệp, nhưng cũng cung cấp khả năng khôi phục một phần thông tin ban đầu. Một ví dụ phổ biến của nén

bán mất mát là nén ảnh theo tỷ lệ chất lượng, trong đó người dùng có thể điều chỉnh mức độ nén để thay đổi chất lượng hình ảnh và kích thước tệp.



Hình 1. Mô tả ý tưởng của nén và giải nén.

- Với $f(x,y)$ là ma trận ảnh ban đầu, khi thực hiện nén sẽ biến đổi ma trận $f(x,y)$ thành ma trận mới $g(x,y)$. Khi giải nén, mục tiêu của thuật toán nén có mất mát là biến ma trận $g(x,y)$ thành $\hat{f}(x,y)$ sao cho $\hat{f}(x,y) \approx f(x,y)$. Nếu $\hat{f}(x,y) = f(x,y)$ thì thuật toán là nén không mất mát.

CHƯƠNG II: THUẬT TOÁN

I. Discrete Cosine Transform

1. Giới thiệu

- Biến đổi cosin rời rạc DCT (Discrete Cosine Transform) được đưa ra bởi Ahmed và các đồng nghiệp vào năm 1974. Từ đó cho đến nay, nó được sử dụng rất phổ biến trong nhiều kỹ thuật xử lý ảnh số nói riêng và xử lý tín hiệu số nói chung. Trong các kỹ thuật thủy văn ảnh dựa trên phép biến đổi dữ liệu ảnh sang miền tần số thì phép biến đổi DCT cũng được sử dụng rất nhiều. Nó được sử dụng trong chuẩn nén JPEG để mã hóa ảnh tĩnh và chuẩn MPEG để mã hóa ảnh động...

- Phần lớn nội dung quan trọng của ảnh thay đổi tương đối chậm trên toàn bộ bức ảnh. Điều đó có nghĩa là giá trị cường độ ít khi thay đổi lên xuống nhiều lần trong một vùng nhỏ (block ảnh 8×8). Khi chuyển một bức ảnh sang miền tần số, các thành phần tần số thấp chứa nhiều thông tin hơn thành phần tần số cao do mắt người khó phát hiện ra các mất mát ở vùng tần số cao. Biến đổi DCT giúp tập trung năng lượng của tín hiệu vào các hệ số quan trọng. Chúng ta có thể giữ lại các hệ số ấy và bỏ qua những thành phần ít hữu ích, từ đó giảm kích thước dữ liệu mà vẫn giữ lại thông tin quan trọng.

2. Ý tưởng thuật toán

- Chia bức hình thành các khối nhỏ có kích thước đồng nhất (8×8 pixel).
- Áp dụng biến đổi DCT lần lượt lên từng khối dữ liệu đã chia ở trên.
- Thực hiện thao tác lượng tử hóa cho từng khối.
- Duyệt khối theo hình zigzag và lưu các giá trị bằng các thuật toán nén không tổn thất.
- Để giải nén và tái tạo lại dữ liệu gốc, ta áp dụng phép biến đổi nghịch DCT (IDCT).

2.1) Quá trình nén

Giả sử ta có ma trận của một khối 8×8 trong ảnh như sau:

$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix} \quad (1)$$

Bước 1: Thiết lập ma trận DCT cho khối 8x8

Ta có công thức DCT được biểu diễn như sau:

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

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

Với khối dữ liệu kích thước 8x8, ta có công thức tính DCT như sau:

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^7 \sum_{y=0}^7 p(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \cos\left[\frac{(2y+1)j\pi}{16}\right] \quad (3)$$

Từ công thức (2) (3), ta thiết lập công thức cho ma trận DCT:

$$T_{ij} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N}\right] & \text{if } i > 0 \end{cases} \quad (4)$$

Với khối 8x8, kết quả của ma trận DCT là:

$$T = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix} \quad (5)$$

Bước 2: Xử lý giá trị cho ma trận ảnh khối 8x8

Trước khi áp dụng công thức biến đổi DCT cho khối 8x8, ta tiến hành trừ đi giá trị 128 cho các giá trị trong ma trận ảnh. Do miền giá trị của cosin chạy từ 1 đến -1, việc làm trên giúp dịch chuyển và tập trung các giá trị xung quanh giá trị 0 sau khi biến đổi DCT. Ngoài ra, trong quá trình nén, các thành phần có giá trị gần 0 hoặc gần 128 có thể được biểu diễn và mã hóa một cách hiệu quả hơn, tạo ra các khối thông tin nén nhỏ hơn.

Ma trận khối 8x8 của ảnh sau khi trừ 128:

$$M = \begin{bmatrix} 26 & -5 & -5 & -5 & -5 & -5 & -5 & 8 \\ 64 & 52 & 8 & 26 & 26 & 26 & 8 & -18 \\ 126 & 70 & 26 & 26 & 52 & 26 & -5 & -5 \\ 111 & 52 & 8 & 52 & 52 & 38 & -5 & -5 \\ 52 & 26 & 8 & 39 & 38 & 21 & 8 & 8 \\ 0 & 8 & -5 & 8 & 26 & 52 & 70 & 26 \\ -5 & -23 & -18 & 21 & 8 & 8 & 52 & 38 \\ -18 & 8 & -5 & -5 & -5 & 8 & 26 & 8 \end{bmatrix} \quad (6)$$

Bước 3: Tiến hành nhân ma trận DCT cho ma trận ảnh khối 8x8

Bắt đầu biến đổi DCT theo công thức sau:

$$D = T M T' \quad (7)$$

Với T là ma trận DCT, T' là ma trận khả nghịch của T, M là ma trận khối 8x8 của ảnh.

Ta nhận được ma trận sau biến đổi có giá trị:

$$D = \begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix} \quad (8)$$

Bước 4: Thực hiện lượng tử hóa

Ta có công thức lượng tử hóa như sau:

$$C_{i,j} = \text{round} \left(\frac{D_{i,j}}{Q_{i,j}} \right) \quad (9)$$

Với $D(i,j)$ là ma trận DCT cho ma trận ảnh khối 8×8 được tính ở bước trên và $Q(i,j)$ là ma trận lượng tử có kích thước 8×8 . Ma trận lượng tử chất lượng nén là 50 (cân bằng giữa chất lượng hình ảnh và tỉ số nén) có giá trị như sau:

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

Lượng tử hóa chính là quá trình giúp dữ liệu được giảm kích thước, quá trình nén ảnh được xảy ra ở bước này. Trước hết, ta cần biết tại sao lại cần lượng tử hóa và tại sao nó lại giúp giảm kích thước tệp.

Số bit được sử dụng để biểu diễn một pixel trong ma trận ảnh phụ thuộc vào độ sâu bit (bit depth) của ảnh. Độ sâu bit xác định số lượng bit được sử dụng để lưu trữ thông tin màu sắc cho mỗi pixel. Trong trường hợp có ma trận ảnh (1), một pixel được biểu diễn bằng 8 bit, có thể có 256 giá trị màu khác nhau.

Ví dụ: Ta có một giá trị pixel là 45 tương ứng với 101101 (6 bits). Khi muốn biểu diễn số 45 với số bit ít hơn, ta không thể làm được. Tuy nhiên, ý tưởng này là có thể nếu biểu diễn nó bằng một số gần bằng thông qua việc cắt bỏ n bit phía sau (phần ít quan trọng và giá trị nhỏ).

+ Cắt bỏ 2 bits ở sau: $1011 = 11$ (Vì mất 2 bits nên ta lấy $11 \times 2^2 = 11 \times 4 = 44$).

+ Cắt bỏ 3 bits ở sau: $101 = 5$ (Vì mất 3 bits nên ta lấy $5 \times 2^3 = 5 \times 8 = 40$).

Trở về ý tưởng của lượng tử hóa, việc chia cho một số theo công thức (9) giúp ta giảm giá trị của nó, đồng nghĩa số bit đã bị cắt giảm. Sai số của giá trị lượng tử

(giá trị được chia càng lớn) thì tính chính xác càng giảm. Đây chính là nguyên nhân chính dẫn đến hệ quả nén có tổn thất của thuật toán.

Giá trị của các phần tử trong ma trận lượng tử là không cố định, ta có thể tùy chỉnh ma trận (10) theo nhu cầu nén theo công thức:

$$Q(n) = Q_{50} \times [(100 - n) / 50] \text{ với } n \geq 50, n \neq 100$$

$$Q(n) = Q_{50} \times (50 / n) \text{ với } n < 50$$

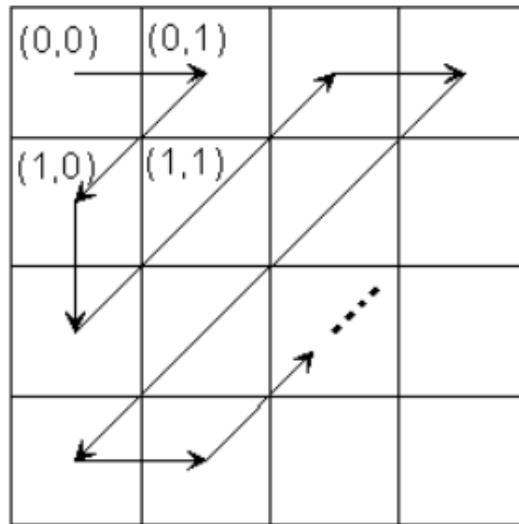
Với n là mức nén giữ lại mong muốn (hay còn gọi là level). Với level càng nhỏ thì chất lượng hình ảnh càng giảm và ngược lại.

Khi áp dụng chia ma trận ảnh sau biến đổi DCT cho ma trận lượng tử chất lượng nén là 50, ta có ma trận kết quả như sau:

$$C = \begin{bmatrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (11)$$

Bước 5: Duyệt ma trận kết quả theo hình zigzag

Ta thực hiện lưu các giá trị theo hình zigzag như hình bên dưới:



(12)

Hình 2. Minh họa cách duyệt ma trận theo hình zigzag

Do sau khi lượng tử hóa, các giá trị 0 xuất hiện với tần xuất cao hơn và tập trung nhiều ở phía dưới của ma trận kết quả. Duyệt ma trận theo hình zigzag giúp ta tiết kiệm không gian và dễ dàng lưu các giá trị 0 ở phía cuối các khi thực hiện các thuật toán nén không tổn thất như Huffman, RLE, LZW,... Cuối cùng, ma trận được biểu diễn với kích thước nhỏ hơn. Thuật toán nén kết thúc.

2.2) Quá trình giải nén

Để giải nén, ta biến đổi tệp nén về lại ma trận kết quả sau khi lượng tử (11). Tùy thuộc vào thuật toán nén khi thực hiện zigzag mà có cách đưa về ma trận kết quả khác nhau cho từng thuật toán nén không tổn thất.

Quá trình đọc lại ảnh thông qua các bước sau:

Bước 1: Nhân lại ma trận lượng tử cho ma trận kết quả (11).

Ta có công thức nhân ma trận:

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

Với $Q(i,j)$ là ma trận lượng tử đã thiết lập ở bước nén, $C(i,j)$ là ma trận kết quả. Ta có ma trận R như sau:

$$R = \begin{bmatrix} 160 & 44 & 20 & 80 & 24 & 0 & 0 & 0 \\ 36 & 108 & 14 & 38 & 26 & 0 & 0 & 0 \\ -98 & -65 & 16 & -48 & -40 & 0 & 0 & 0 \\ -42 & -85 & 0 & -29 & 0 & 0 & 0 & 0 \\ -36 & 22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (12)$$

Bước 2: Biến đổi nghịch DCT (IDCT)

Ta có công thức cho quá trình trên:

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

Sau khi nhân ma trận biến đổi nghịch DCT, ta phải cộng lại giá trị 128 cho từng giá trị pixel để khôi phục lại miền giá trị 0 – 255 cho ảnh màu. Ta có kết quả ma trận ảnh một khối 8x8 sau giải nén như sau:

$$Decompressed = \begin{bmatrix} 149 & 134 & 119 & 116 & 121 & 126 & 127 & 128 \\ 204 & 168 & 140 & 144 & 155 & 150 & 135 & 125 \\ 253 & 195 & 155 & 166 & 183 & 165 & 131 & 111 \\ 245 & 185 & 148 & 166 & 184 & 160 & 124 & 107 \\ 188 & 149 & 132 & 155 & 172 & 159 & 141 & 136 \\ 132 & 123 & 125 & 143 & 160 & 166 & 168 & 171 \\ 109 & 119 & 126 & 128 & 139 & 158 & 168 & 166 \\ 111 & 127 & 127 & 114 & 118 & 141 & 147 & 135 \end{bmatrix} \quad (13)$$

3. Áp dụng ý tưởng của thuật toán cho ảnh màu RGB**+ Nén ảnh:**

1. Đọc ảnh đầu vào bằng thư viện cv2.
2. Tách ảnh màu RGB thành ma trận 3 ma trận độc lập. Mỗi ma trận sẽ đại diện cho 1 kênh: kênh màu đỏ (R), kênh màu xanh lá (G), kênh màu xanh lam (B)
3. Lần lượt duyệt qua các kênh màu, lấy giá trị của từng pixel trừ đi 128.

4. Lần lượt duyệt qua các kênh màu, chia các khối 8x8 cho từng kênh, sau đó áp dụng biến đổi DCT cho các khối vừa chia trong từng kênh màu.
5. Lần lượt duyệt qua các kênh màu, áp dụng chia ma trận lượng tử cho các khối 8x8 trong từng kênh.
6. Sau đó, nén ảnh bằng cách lưu các giá trị trong từng khối của từng kênh màu theo hình zigzag.

+ Giải nén ảnh:

1. Giải mã kết quả nén để thu lại ma trận kết quả sau khi lượng tử hóa.
2. Lần lượt duyệt qua các kênh màu, áp dụng nhân ma trận lượng tử cho các khối 8x8 trong từng kênh.
3. Lần lượt duyệt qua các kênh màu, với từng khối 8x8 của mỗi kênh, ta áp dụng biến đổi nghịch DCT cho chúng.
4. Lần lượt duyệt qua các kênh màu, lấy giá trị của từng pixel cộng cho 128 để thu về đúng giá trị màu cho ảnh.
5. Gộp ba kênh màu lại thành một ma trận 3 chiều.
6. Đọc ảnh kết quả bằng thư viện cv2 và lưu lại hình sau giải nén.

II. Singular Value Decomposition

1. Giới thiệu

- SVD là một phương pháp phân tích ma trận, có nhiều ứng dụng như nén ảnh, pháp hiện ảnh giả mạo, thủy văn số, nhận diện khuôn mặt... Điểm đặc biệt của SVD là nó có thể áp dụng được trên bất kỳ ma trận thực nào. Thuật toán giúp phân tích một ma trận bất kỳ thành tích của ba ma trận thành phần. Thông qua đó nó có thể trích chọn các đặc trưng bền vững và quan trọng. Trong bài báo cáo này, nhóm sẽ nghiên cứu về phân tích SVD trong nén dữ liệu dạng ảnh.

2. Ý tưởng thuật toán

- Xét một ma trận A có m hàng và n cột (với $n \leq m$), có hạng là r và $r \leq n \leq m$, phép biến đổi SVD phân tích ma trận A thành ba ma trận U , S , V như sau:

$$A = USV^T$$

- Trong đó:

- U : Ma trận trực giao cấp m :

$$U = [u_1, u_2, \dots, u_r, u_{r+1}, \dots, u_n]$$

Với các vector u_i ($i = 1, 2, \dots, m$) tạo thành một tập trực chuẩn:

$$u_i u_j = \partial_{ij} = \{1, 2, \dots, i = j \mid 0 \dots i \neq j\}$$

- V : Ma trận trực giao cấp n :

$$V = [v_1, v_2, \dots, v_r, v_{r+1}, \dots, v_n]$$

Với các vector v_i ($i = 1, 2, \dots, n$) tạo thành một tập trực chuẩn:

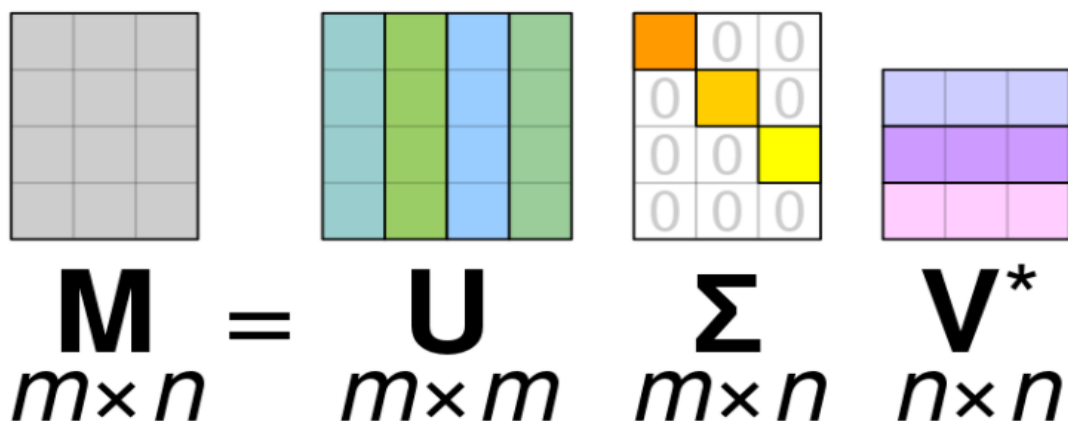
$$u_i u_j = \partial_{ij} = \{1, 2, \dots, i = j \mid 0 \dots i \neq j\}$$

- Như vậy, ma trận U là ma trận gồm các vector riêng trái của A , ma trận V là ma trận gồm các vector riêng phải của A .

- S là ma trận đường chéo, với mỗi phần tử trên đường chéo là một trị riêng của A . Ký hiệu các giá trị trên đường chéo chính của S là d , các giá trị này có độ lớn được sắp xếp theo thứ tự như sau:

$$\partial_1 \geq \partial_2 \geq \partial_3 \geq \dots \geq \partial_t \geq 0$$

- Các giá trị còn lại của ∂_i ($i = r + 1, \dots, n$) đều bằng 0. Khi áp dụng SVD vào ma trận điểm ảnh, “năng lượng” của ảnh được tập trung vào các phần tử ở hàng đầu, cột đầu của các ma trận U , V và các phần tử đầu tiên trên đường chéo của S . Các phần tử này có tính ổn định cao, vì vậy có thể sử dụng chúng để làm các đặc trưng bền vững, ứng dụng trong nhiều việc khác nhau.



Hình 3. Tính ma trận $M = USV^T$

2.1) Quá trình nén

1. Chuẩn bị ảnh: chuẩn bị ảnh muốn nén. Ảnh được biểu diễn dưới dạng ma trận pixel.
2. Áp dụng SVD: áp dụng phân tích SVD lên ma trận. SVD phân tích một ma trận thành ba ma trận U , S và V . Ma trận U và V chứa các singular vector, ma trận S là ma trận đường chéo chứa giá trị K .
3. Chọn k : để nén ảnh, ta cần phải chọn số k được sử dụng trong phép phân tích SVD. Bằng cách giữ lại một số lượng k nhất định trong ma trận đường chéo S , ta có thể giảm kích thước của ma trận.
4. Tạo lại ảnh nén: sử dụng các ma trận U , S và V đã được giới hạn bằng tham số k để tạo lại ảnh nén. Bằng cách nhân ma trận U với ma trận S và ma trận V , ta sẽ được một ước lượng nén của ma trận ảnh gốc.
5. Hiển thị và lưu ảnh nén: cuối cùng hiển thị ảnh nén lên màn hình và lưu về tệp tin.

2.2) Quá trình giải nén

1. Chuẩn bị dữ liệu: Dữ liệu là ảnh đã được nén bao gồm ma trận U , ma trận S và ma trận V từ quá trình nén SVD trên.
2. Tạo các ma trận: Sử dụng ma trận S , tạo ma trận giá trị k bằng cách đặt k trên đường chéo chính của ma trận. Đảm bảo rằng k đã được giới hạn và các giá trị k không quan trọng đã bị loại bỏ ở quá trình nén.
3. Tạo lại ma trận ảnh: Sử dụng ma trận U , ma trận S và ma trận V nhân các ma trận này với nhau để tạo lại ma trận gốc.
4. Hiển thị ảnh: cuối cùng, hiển thị ảnh đã được giải nén lên màn hình hoặc lưu vào tệp tin ảnh.

3. Áp dụng ý tưởng của thuật toán cho ảnh màu RGB

1. Chia ma trận hình ảnh RGB đầu vào thành 3 ma trận riêng lẻ, tương ứng với các kênh màu Red, Green, Blue





2. Sau khi áp dụng SVD để phân tích mỗi ma trận riêng lẻ thành 3 ma trận U , S và V và giới hạn k , sử dụng các ma trận U , S và V đó để tạo lại ma trận ảnh nén cho mỗi kênh màu Red, Green, Blue.

3. Kết hợp các ma trận kênh màu trên để tạo ảnh màu RGB.

CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ

I. Dataset

- Để đánh giá hai thuật toán DCT và SVD, nhóm sử dụng bốn bức ảnh có kích thước khác nhau nhằm đánh giá và so sánh một cách hiệu quả và chính xác. Thông tin các bức ảnh để test được trình bày trong bảng sau:

Tên	Ảnh	Kích thước
Ảnh 1		(256, 256, 3)
Ảnh 2		(512, 512, 3)
Ảnh 3		(400, 615, 3)
Ảnh 4		(1024, 1024, 3)

Bảng 2. Thông tin các ảnh sử dụng để đánh giá

II. Các độ đo sử dụng

1. Hiệu suất nén

- Là tỉ lệ giữa kích thước của ảnh ban đầu và sau khi giải nén được lưu vào bộ nhớ.

1.1) Cách tính cho thuật toán SVD

$$\text{Hiệu suất nén (\%)} = \frac{k \times (m + n + 1)}{2 \times m \times n} \times 100$$

➤ Trong đó:

- k là giá trị singular value ở ma trận đường chéo S.
- m, n lần lượt là chiều cao và chiều rộng của ảnh.

1.2) Cách tính cho thuật toán DCT

$$\text{Hiệu suất nén (\%)} = \frac{|I(i, j) - I'(i, j)|}{I(i, j)} \times 100$$

➤ Trong đó:

- I(i, j) là kích thước của ảnh gốc.
- I'(i, j) là kích thước của ảnh sau khi giải nén.

2. Mean Squared Error

- Mean Squared Error (MSE) tính toán sự khác biệt bình phương trung bình khoảng cách giữa các giá trị bit của ảnh gốc và giá trị bit của ảnh sau khi nén dữ liệu.

- Công thức tính như sau:

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

➤ Trong đó:

- I(i, j) là kích thước của ảnh gốc.
- I'(i, j) là kích thước của ảnh sau khi giải nén.
- m, n lần lượt là chiều cao và chiều rộng của ảnh.

3. Root Mean Square Error

- RMSE (Root Mean Square Error) được tính bằng căn của giá trị MSE.

- RMSE thấp cho thấy sự tương đồng cao giữa hai ảnh và chất lượng nén tốt.
- RMSE cao cho thấy sự khác biệt lớn giữa hai ảnh và chất lượng nén kém.

- Công thức tính như sau:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

4. Peak Signal-to-Noise Ratio

- 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 như sau:

$$\text{PSNR} = 10 \times \log_{10} \frac{\text{MAX}_I^2}{\text{MSE}}$$

➤ Trong đó:

- MAX_I là giá trị pixel lớn nhất của ảnh gốc $I(i, j)$ (thường là 255).

III. Kết quả chạy thực nghiệm

1. Thuật toán DCT

Ảnh	level	Thời gian nén và giải nén (s)	RMSE	PSNR	Hiệu suất nén (%)
Ảnh 1	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
Ảnh 2	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
Ảnh 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
Ảnh 4	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

Bảng 3. Kết quả chạy thực nghiệm thuật toán DCT trên bộ dataset với các level khác nhau.

2. Thuật toán SVD

Ảnh	k	Thời gian nén và giải nén (s)	RMSE	PSNR	Hiệu suất nén (%)
Ảnh 1	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
Ảnh 2	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
Ảnh 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
Ảnh 4	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

Bảng 4. Kết quả chạy thực nghiệm thuật toán SVD trên bộ dataset với các k khác nhau.

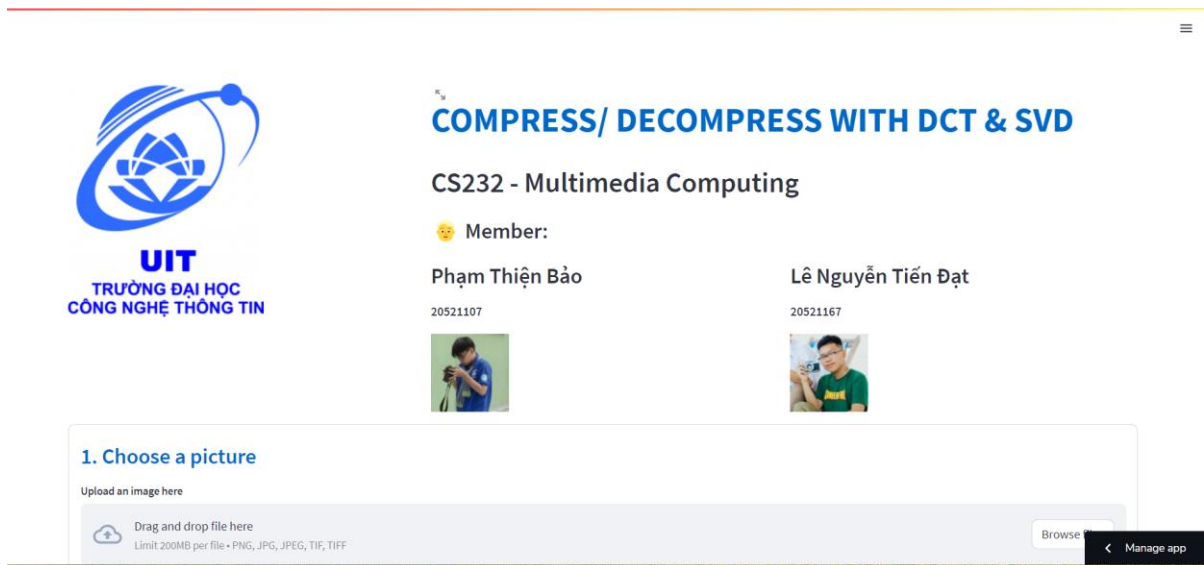
IV. Nhận xét

- Khi thay đổi level cho ma trận lượng tử, thời gian chạy của thuật toán DCT không có sự thay đổi nhiều. Vì thuật toán chia hình thành các khối 8×8 , sau đó lần lượt biến đổi DCT và chia ma trận lượng tử để nén cho từng khối. Giá trị level chỉ làm thay đổi giá trị của ma trận lượng tử để tăng hiệu suất nén, chứ không làm giảm số lần duyệt qua các khối 8×8 được chia. Do vậy, thuật toán vẫn phải chạy hết tất cả các khối 8×8 trong hình để thực hiện quá trình nén.
- Khi thay đổi giá trị k càng cao cho SVD thì thời gian chạy càng cao. Vì nếu k càng lớn thì ma trận S càng lớn, thuật toán phải thực hiện nhiều phép tính toán hơn dẫn tới việc thời gian nén và giải nén tăng.
- Khi chạy trên bộ dataset đánh giá, RMSE cho từng hình của thuật toán SVD có giá trị nhỏ với k càng lớn và giá trị RMSE cho từng hình của thuật toán DCT cũng nhỏ khi level cao. Điều này chứng tỏ kết quả thực nghiệm đúng với ý tưởng của thuật toán và cách tính phép đo.
- Khi chạy trên bộ dataset đánh giá, PSNR cho từng hình của thuật toán SVD có giá trị lớn với k càng lớn và giá trị PSNR cho từng hình của DCT cũng lớn khi level cao. Điều này chứng tỏ kết quả thực nghiệm đúng với ý tưởng của thuật toán và cách tính phép đo.
- Với mỗi bức hình đánh giá, thuật toán SVD có giá trị PSNR cao hơn hẳn DCT (xét hình 4 thì $PSNR = 83.91$). Có thể nói, trong thực nghiệm này, thuật toán SVD có chất lượng ảnh sau giải nén cao hơn thuật toán DCT. Bởi lẽ, việc chia từng giá trị pixel cho ma trận lượng tử của DCT đã dẫn đến mất mát nhiều thông tin. Tuy nhiên, về mặt thời gian DCT lại chiếm ưu thế cao tuyệt đối, giúp cho DCT được sử dụng phổ biến hơn trong ứng dụng nén và giải nén ảnh khi có thể cân bằng được yếu tố thời gian và chất lượng nén.

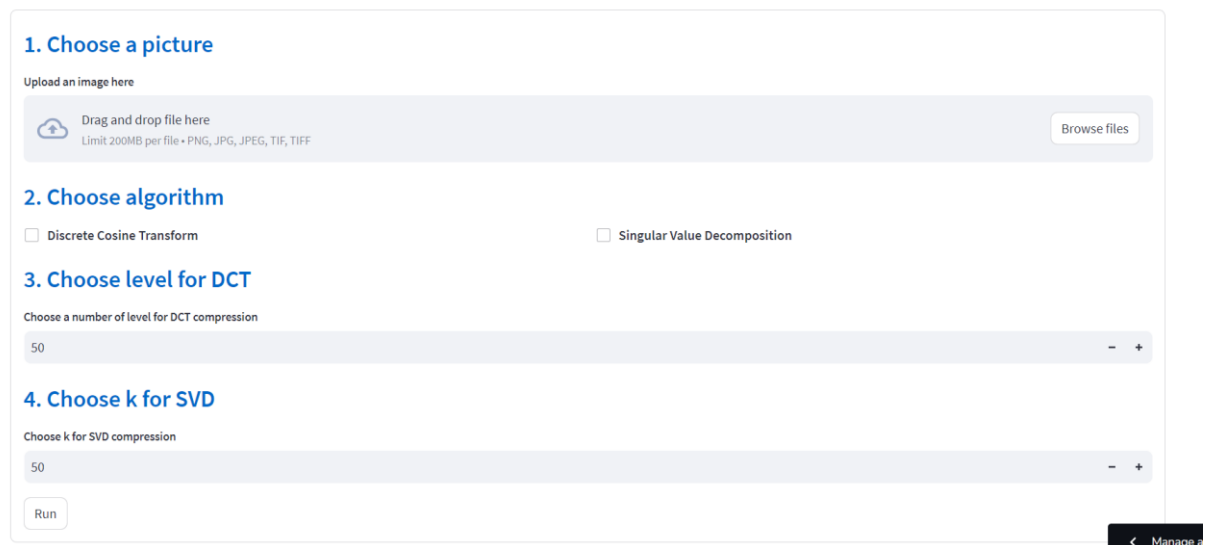
CHƯƠNG IV: XÂY DỰNG ỨNG DỤNG WEB

I. Xây dựng giao diện

- Link github: https://github.com/beetibao/CS232_MultimediaComputing
- Link website: <https://cs232demo.streamlit.app/>
- Website hỗ trợ trong việc hiển thị các hình ảnh trước khi nén và sau khi giải nén của hai thuật toán. Sau đó, website trả về bảng kết quả đánh giá chi tiết cho từng thuật toán. Sau đây là một số hình ảnh của website:



Hình 4. Giao diện chính.



Hình 5. Giao diện điều chỉnh các thông tin về thuật toán.

Image After:



Result_DCT:

	time(s)	compression_ratio(%)	RMSE	PSNR
0	0.8290	36	6.6800	31.5700

Hình 6. Giao diện kết quả khi nén ảnh bằng thuật toán DCT.

II. Công cụ hỗ trợ

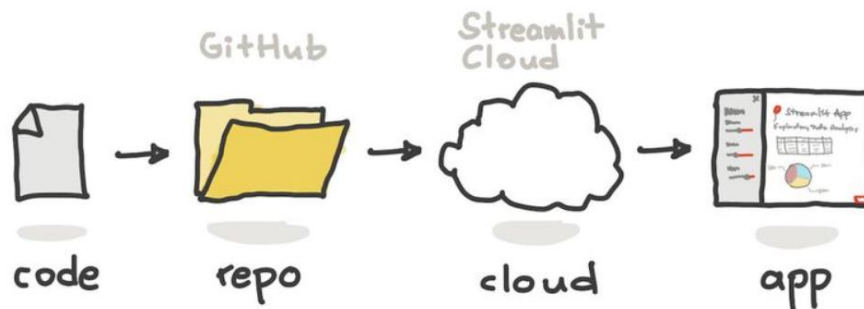
- Streamlit là một công cụ mã nguồn mở được sử dụng để xây dựng ứng dụng web nhanh chóng và dễ dàng bằng Python. Nó cho phép bạn tạo các ứng dụng web tương tác với giao diện người dùng trực quan để trình bày, trực quan hóa và tương tác với dữ liệu và mô hình.

- Streamlit Cloud là một dịch vụ đám mây được cung cấp bởi Streamlit, cho phép ta dễ dàng triển khai, chia sẻ và triển khai ứng dụng Streamlit của mình trực tuyến. Đây là một nền tảng đám mây được tối ưu hóa cho việc chạy các ứng dụng Streamlit mà không cần phải lo lắng về việc quản lý máy chủ và cấu hình.

- Với Streamlit Cloud, ta có thể:

- Triển khai ứng dụng Streamlit dễ dàng: triển khai ứng dụng Streamlit của mình chỉ bằng một vài dòng lệnh hoặc thông qua giao diện người dùng đơn giản. Streamlit Cloud sẽ tự động xử lý việc cài đặt, cấu hình và triển khai ứng dụng lên một môi trường đám mây.
- Chia sẻ ứng dụng trực tuyến: chia sẻ ứng dụng Streamlit của mình với người khác bằng cách cung cấp URL duy nhất của ứng dụng. Người dùng khác có thể truy cập và tương tác với ứng dụng của bạn thông qua trình duyệt web mà không cần phải cài đặt bất kỳ công cụ hay thư viện nào.
- Tự động cập nhật và theo dõi: Streamlit Cloud sẽ tự động theo dõi các thay đổi trong mã nguồn và cập nhật ứng dụng khi bạn lưu lại. Điều này giúp ta dễ dàng thấy kết quả của các thay đổi mới nhất mà không cần phải thực hiện các bước triển khai thủ công.

- Quản lý phiên bản và thống kê: cung cấp các tính năng quản lý phiên bản và thống kê giúp ta xem lại và phân tích các phiên bản và thông tin liên quan đến việc sử dụng ứng dụng của bạn.



Hình 7. Cách xây dựng ứng dụng web thông qua công cụ Streamlit.

TÀI LIỆU THAM KHẢO

- [1] Ken, Cabeen, and P. Gent. "Image compression and the discrete cosine transform." *College of the Redwoods, Tech. Rep* (1998).
- [2] Ahmed, Nasir, T_ Natarajan, and Kamisetty R. Rao. "Discrete cosine transform." *IEEE transactions on Computers* 100.1 (1974): 90-93.
- [3] Watson, Andrew B. "Image compression using the discrete cosine transform." *Mathematica journal* 4.1 (1994): 81.
- [4] Mounika, K., D. Sri Navya Lakshmi, and K. Alekya. "SVD based image compression." *International journal of engineering research and general science* 3.2 (2015): 1271-1278.
- [5] Prasantha, H. S., H. L. Shashidhara, and KN Balasubramanya Murthy. "Image compression using SVD." *International conference on computational intelligence and multimedia applications (ICCIMA 2007)*. Vol. 3. IEEE, 2007.