INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS FLORIANO
CURSO TADS

KAUAN BEZERRA BENVINDO

ANÁLISE DE UM ALGORITMO PARA A COMPRESSÃO DE IMAGEM UTILIZANDO A TÉCNICA SVD ALEATÓRIO

FLORIANO - PI

2025

KAUAN BEZERRA BENVINDO

ANÁLISE DE UM ALGORITMO PARA A COMPRESSÃO DE IMAGEM UTILIZANDO A TÉCNICA SVD ALEATÓRIO

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Floriano .

Orientador: Profª. Me. André Francisco Coêlho Castro.

FLORIANO - PI

2025

ANÁLISE DE UM ALGORITMO PARA A COMPRESSÃO DE IMAGEM UTILIZANDO A TÉCNICA SVD ALEATÓRIO

Kauan Bezerra Benvindo[1]
André Francisco Coêlho Castro[2]

RESUMO

Este trabalho apresenta a implementação e avaliação de um algoritmo de compressão de imagens baseado na Decomposição de Valores Singulares Aleatória (Randomized SVD). Desenvolvido em Python e aplicado a imagens em escala de cinza no formato BMP, o método busca reduzir o tamanho de armazenamento mantendo níveis aceitáveis de qualidade visual. Foram realizados testes variando o parâmetro de controle k, a fim de analisar sua influência no índice de compressão e no bitrate. Os resultados mostram que o algoritmo é capaz de gerar compressões significativas, embora com perda perceptível de detalhes, mantendo ainda assim uma qualidade suficiente para identificação do conteúdo. Observou-se também que o desempenho do método é insatisfatório para imagens previamente compactadas, como JPEG, e inaplicável a imagens coloridas e vídeos. Conclui-se que o Randomized SVD é viável para imagens em tons de cinza não compactadas, especialmente quando se busca redução acentuada de tamanho. Recomenda-se, em trabalhos futuros, adaptar o algoritmo para imagens coloridas e explorar sua aplicação em vídeos.

Palavras-chave: compressão de imagens; randomized SVD; decomposição de valores singulares; processamento de imagens.

ABSTRACT

This work presents the implementation and evaluation of an image compression algorithm based on Randomized Singular Value Decomposition (Randomized SVD). Developed in Python and applied to grayscale BMP images, the method aims to reduce storage size while maintaining acceptable visual quality. Tests varying the control parameter k were conducted to analyze its influence on compression ratio and bitrate. The results indicate that the algorithm achieves significant compression, though with a noticeable loss of detail, while still preserving sufficient quality for content recognition. It was also observed that the method performs poorly on already compressed images, such as JPEG, and is not applicable to color images or videos. The findings suggest that Randomized SVD is suitable for uncompressed grayscale images, particularly when substantial size reduction is desired. Future work should focus on adapting the algorithm for color images and investigating its potential use in video compression.

Keywords: image compression; randomized SVD; singular value decomposition; image processing.

Data de aprovação: 15/07/2025

---

[1] Graduando em Tecnologia e Análise e Desenvolvimento de Sistemas. Instituto Federal do Piauí - Campus Floriano. kauanbenvindo8@gmail.com

[2]Mestre em Modelagem Matemática e Computacional pela UFPB. Instituto Federal do Piauí - Campus Floriano. andrecastro@ifpi.edu.br.

1 INTRODUÇÃO

Um algoritmo para compressão de imagens tem como objetivo reduzir e redimensionar o tamanho de uma imagem, bem como o seu armazenamento de forma que não comprometa de forma total a qualidade da imagem original. Os primeiros algoritmos de compressão de imagem surgiram na década de 80 como uma tentativa de permitir a exibição de imagens em telas de computadores mais fracas utilizando da compressão para arquivos JPEG, formato esse que foi capaz de difundir a compressão de dados com perdas ao remover dados imperceptíveis aos olhos sem comprometer a qualidade da imagem (Adobe, s.d.).

Devido ao crescente número de tecnologias de captura de imagem, áudio e vídeo de alta definição, o emprego de tecnologias capazes de transportar estes dados de forma mais eficiente se tornaram necessárias, tornando-os em arquivos mais facilmente manipuláveis, disponibilizando uma melhor experiência para os usuários que estejam os acessando. Conforme Brito (1995), a compressão de dados e de imagens consolidou-se nesse período como uma alternativa indispensável para reduzir redundâncias e permitir maior eficiência no armazenamento e na transmissão de arquivos multimídia.

Entretanto, a compressão de imagens nem sempre se torna viável e fácil um dos maiores empecilhos se dá pela redução da imagem preservando a qualidade da mesma, estando o arquivo reduzido é impossível que este não sofra uma queda na qualidade, estes algoritmos buscam portanto balancear a compressão da imagem sem corromper a imagem final, além de buscar evitar distorções em imagens, geradas principalmente em imagens com alta definição (Felippe, s.d.).

Este estudo, portanto, explora a aplicação do método de Decomposição de Valores Singulares Aleatório (Randomized SVD) na compressão de imagens, uma abordagem eficiente que reduz o custo computacional e acelera o processo, especialmente para imagens de alta resolução (Burton; Kutz, 2019).

O objetivo deste trabalho é implementar e avaliar a eficácia de um algoritmo de compressão de imagens baseado no método Randomized SVD, investigando seu desempenho em termos de qualidade visual, taxa de bits e índice de compressão.

2 REFERENCIAL TEÓRICO

2.1 Algoritmo

Na área da programação, um algoritmo poderia ser definido como a execução de passos matemáticos e computacionais que determinam o funcionamento de um código, de forma análoga à receita e ao preparo de um bolo. Mas para se entender de fato o que seria um algoritmo se faz necessário saber o que seria lógica e seu papel na construção de um algoritmo.

O conceito de lógica dar-se-á no estabelecimento de um raciocínio padrão durante o desempenho de uma tarefa ou na organização de um pensamento. Como definido pelo professor Dr. Mortari (2001, p.2), "Lógica é a ciência que estuda princípios e métodos de inferência, tendo o objetivo principal de determinar em que condições certas coisas se seguem (são consequências), ou não, de outras."

Sob esta ótica, entende-se que a aplicabilidade da lógica é essencial no controle e desenvolvimento de aplicações, sendo explorada de forma intrínseca a programação. Por meio desta interação, surge a sequência dos processos lógicos que determinam o funcionamento de um algoritmo, esta série de processos executados em um computador determina-se como Lógica de Programação.

Portanto, o algoritmo será responsável por incorporar a lógica aplicada e estabelecer uma conversa entre os dados em transporte e a máquina, convertendo-os em valores numéricos que o computador é capaz de ler, armazenar e escrever.

2.2 Compressão de Dados e de Imagens

A compressão de dados se dá pela redução otimizada de uma informação não organizada, gerando uma versão reduzida dos dados, porém mantendo o sentido da mensagem inalterado. Tal operação se torna necessária mediante ao crescimento das informações disponíveis na rede, em visto que estes mesmos dados vêm se tornando cada vez mais complexos e redundantes. Para isto, a compressão de dados é capaz de facilitar tanto o armazenamento como o envio e captação destas informações.

As metodologias de compressão de dados se dividem dependendo do resultado gerado após a compressão. Caso a qualidade da imagem se mantenha

inalterada após sofrer a compressão ela é determinada sem perdas (*loseless*), entretanto nem todos os casos serão necessários manter a total qualidade da informação, para estes casos são determinados métodos de compressão com perdas (*lossy*). De acordo com Cloudinary (s.d.), a escolha do algoritmo de compressão deve considerar o equilíbrio entre qualidade visual e tamanho do arquivo, uma vez que diferentes técnicas apresentam vantagens em contextos distintos, como imagens estáticas ou transmissões em tempo real.

Por seguinte, o entendimento de imagens como parte destes dados a serem comprimidos resulta na aplicabilidade dos conceitos de compressão de dados para com as imagens. Dentro da computação, uma imagem é representada por um sistema de bits, entretanto este sistema depende da limitação de bits propostos pela quantidade de bits, um sistema monocromático será representado por um sistema de 1bit, sendo este capaz de estar representando até duas cores, por exemplo: preto e branco.

Para os demais sistemas de representação por bits, pode ser aplicada a seguinte equação:

$$2^x = y$$

Sendo x, o total de bits do sistema e y o total de cores possíveis a serem representados nos elementos da imagem. Além do sistema de bits, as imagens são agrupadas também em *pixels ou picture element* (elemento de imagem), a representação dos pixels são dispostas em matrizes gráficas de duas dimensões, demarcada pela largura e altura das sequências de *pixels* na tela. Dada uma Matriz A, com M representando a largura da matriz e N a altura dela, com estes últimos sendo dados pela quantidade de *pixels* presentes.

$$A_{MxN}$$

O estudo da compressão de imagens, portanto, abrangeria a aplicação de metodologias capazes de diminuir esta matriz de representação, assim como o espaço de armazenamento da imagem sem perder as informações presentes na imagem original. Dado este conceito, algumas técnicas de compressão de dados capazes de reduzir o armazenamento de uma imagem seriam métodos como o JPEG e o PNG.

2.2.1 JPEG ou JPG

No método JPEG ou JPG, é aplicado através de um algoritmo DCT (Transformada Discreta do Cosseno), tal método de compressão que busca diminuir a quantidade de dados irrelevantes ou redundantes na imagem, mas mantendo a qualidade da imagem bem próxima da original. (Jacobi; Silveira, 2013), por exemplo em uma faixa com variações de uma mesma cor, este algoritmo busca reduzi-la para um tamanho menor aproximando as cores para uma mais homogênea.

FIGURA 1 - Exemplo de compressão de cores

Fonte: Elaborado pelo autor (2024)

Com isto o valor resultante da compressão de imagem para arquivos JPEG, acaba por ser considerado um método com perdas, pois é priorizado o armazenamento da imagem de forma que a imagem não apresenta perda de informações perceptíveis aos olhos humanos.

2.2.2 PNG

A compressão de imagens em arquivos PNG, podem ser realizadas tanto de forma com perdas como de forma sem perdas, dependendo apenas de qual algoritmo se baseia na compressão do arquivo. Para comprimir sem perdas pode-se utilizar o algoritmo DEFLATE, este algoritmo aplica constituições de dois métodos de compressão, sendo eles o método LZ77 (Lempel-Ziv77) e a codificação de Huffman.

No método LZ77 ocorre a conversão sequências de *pixels* em séries mais compactas de *pixels*, funcionando por meio de uma janela que armazena

temporariamente uma janela de valores buscando por repetições de algum dos valores expressos na janela. De forma detalhada é analisada uma sequência de *n* entradas no qual algumas dessas entradas possuem repetições, os caracteres repetidos são substituídos por um valor que contém informações referentes a primeira ocorrência e atual, como a posição da primeira ocorrência de um valor repetido e o tamanho da sequência repetida.

Tenhamos como exemplo a sequência *ABTYAB,* o método LZ77 irá inicializar os primeiros valores (A B T Y) de forma literal, pois estes não se repetem, mas ao inicializar o próximo valor ocorre a repetição de um caractere, este irá ser convertido em uma tupla que armazena informações como a posição desta entrada com relação a primeira ocorrência e o tamanho desta, sendo inserida desta forma ABTY(4,1). Em seguida, é inserido o último valor, mas este também já possui uma ocorrência logo sua representação será feita da seguinte forma, ABTY(4,1)(4,1), sendo este o resultado final, porém como as duas últimas entradas são uma sequência que também está repetida a representação poderá ser realizada da seguinte maneira ABTY(4,2).

Após a realização da etapa LZ77 é aplicada a codificação de Huffman que transforma uma cadeia de valores em representações binárias, onde valores mais presentes recebem valores menores que outros menos frequentes. Sendo assim, uma sequência que possua uma informação muito repetida, esta será representada por um valor mais curto (1) enquanto um mais ausente apresentará um valor mais longo (111).

Na compressão de imagens, existem várias maneiras de se realizar a compressão, cada método varia conforme o resultado esperado e o objetivo da redução, caso a necessidade principal seja a maior redução do espaço da imagem, a compressão sem perdas será utilizada, mas em sistemas mais críticos, onde apenas a simplificação da imagem original sem que haja a perda total das informações da imagem original, um algoritmo com perda controlada de dados pode vir a ser empregado.

2.3 Decomposição de Valores Singulares (SVD)

Ao ser representada em uma matriz de cores, uma imagem tende a gerar um valor descontrolado que irá depender do número de elementos nela representados,

da escala de cores dela, assim como da quantidade de *pixels* nela apresentada. Para decomposição de matrizes como esta se torna inviável a aplicação de um método de redução como feito em matrizes quadráticas, para estes casos a decomposição de valores singulares (SVD) vem como uma opção de redução de uma matriz não quadrática.

O método SVD é um importante meio de fatoração para matrizes computacionais, sendo capaz de distinguir e descartar elementos redundantes ou pouco essenciais na imagem. A organização de uma imagem em meio computacional pode ser feita de duas formas, caso esta for representada em escala de tons cinza, variando de preto a branco, uma matriz A é gerada contém elementos que variam dependendo da intensidade do brilho, recebendo valores de 0 a 255 na escala de 8 bits. Caso a imagem seja colorida, geralmente são designados três canais de cores (RGB, por exemplo) no qual o algoritmo SVD é aplicado em cada um destes canais.

O funcionamento de um algoritmo SVD permite decompor ou representar as matrizes geradas através do produto de outras três matrizes:

$$A = U\Sigma V^T$$

Onde:

- U é uma matriz ortogonal (m x m)
- Σ sendo uma matriz diagonal (m x n) de valores singulares da matriz A
- V^T é uma matriz ortogonal (n x n), em que T representa a matriz transposta desta

A construção de cada uma destas matrizes se deriva dos resultados obtidos dos autovetores e autovalores das matrizes associadas AA^T e A^TA.

Os valores singulares da matriz diagonal Σ são obtidos através das raízes quadradas dos autovalores de AA^T ou A^TA, sendo estes organizados de forma decrescente na matriz.

A matriz V é obtida pelos autovetores de A^TA através da seguinte equação, $A^TAv_i = \lambda_i v_i$. Em que:

- v_i são os autovetores de A^TA
- E $\lambda_i = \sigma_i^2$ são os autovalores correspondentes

Por fim cada autovetor v_i corresponde a uma coluna da matriz V, portanto $V = [v_1\, v_2 \ldots v_n]$.

Para determinar a matriz U, utiliza-se os autovetores de AA^T, com a equação $AA^T u_i = \lambda_i u_i$, onde:

- u_i são os autovetores de AA^T
- E $\lambda_i = \sigma^2{}_i$ são os mesmos autovalores usados para a matriz V.

Em seguida, cada autovetor u_i representa uma coluna da matriz U, $U = [u_1\, u_2 \ldots u_n]$.

Mesmo que eficiente na compressão de imagens, algoritmos SVD em sua forma mais crua, podem apresentar quedas de desempenho em matrizes muito grandes, se tornando muito custosos computacionalmente, para evitar tal ocorrência aplicasse métodos derivados do SVD, sejam com métodos iterativos ou através do método *Randomized* SVD.

2.4 Randomized SVD

O método Randomized SVD é uma variante do modelo tradicional do SVD, surgiu como resposta a manipulação de matrizes maiores, no qual utiliza-se métodos probabilísticos e amostragens aleatórias em que a precisão do produto das matrizes $U\Sigma V^T$ resulta num valor aproximado da decomposição da matriz inicial.

O Randomized SVD permite que seja estabelecido um limite nos valores singulares através de um parâmetro (k), sendo este calculado por um valor que representa uma dimensão muito menor que a as dimensões da matriz inicial (A). Dado este contexto o produto das matrizes com o limite de valores resulta na seguinte relação:

$$A \approx U_k \Sigma_k V^T{}_k$$

Este modelo trabalha com a projeção aleatória de uma matriz Y de tamanho *m x (k+p)* com *p,* onde p representa uma espécie de controle na aproximação da decomposição dos valores. Para representação da matriz Y é apresentada uma matriz Ω de tamanho [n x (k +p)], sendo esta aplicada na relação: $Y = A\Omega$, para gerar uma projeção da matriz A em dimensão reduzida.

Por seguinte, a matriz Y é decomposta e duas matrizes, uma matriz ortonormal $Q_{m\,x\,(k+p)}$ e uma matriz triangular superior $R_{(k+p)x(k+p)}$. Após isto é gerada uma matriz B de tamanho reduzido na relação:

$$B_{(k+p)\,x\,n} = Q^T A$$

Após a criação da matriz B realizada uma decomposição SVD tradicional para busca exata de B, onde uma matriz U* de tamanho (k + p) x (k + p), junto de uma matriz Σ contendo os valores de k + p e de uma matriz transposta $V_{n\,x\,(k+p)}$ geram o produto desta decomposição. Por fim, em seguida da decomposição é feita uma reconstrução aproximada para uma matriz $U_{m\,x\,(k+p)}$, dada pela relação $U = QU^*$ e são separados cada um dos valores *k* e seus autovetores correspondentes em U, Σ, V .

Logo, a implementação do método randomized, permite uma maior escalabilidade para imagens e vídeos de matrizes maiores, permitindo ajustar a precisão do resultado final e diminuir o tempo total para a compressão, mantendo uma aproximação reduzida da matriz original sem que a perda de informações seja danosa a imagem resultante.

3 METODOLOGIA

Este estudo teve como objetivo implementar e avaliar o desempenho de um algoritmo de compressão de imagens baseado no método de SVD aleatório (Randomized SVD), considerando aspectos como qualidade, tempo de execução e tamanho da imagem comprimida. A técnica empregada é indicada especialmente para imagens de alta resolução, pois permite a compressão de forma mais rápida e com menor custo computacional.

Para os experimentos, foram utilizados um conjunto de dados genéricos composto por imagens digitais em formato de imagem Bitmap (.bmp), devido este formato de imagem não ter compactação de dados. O conjunto de dados incluiu imagens de alta e baixa resolução, com variações de complexidade visual, como fotografias naturais, padrões geométricos e texturas homogêneas. Também foram realizados testes com alta e com baixa taxa de compressão para melhor avaliação do algoritmo.

A implementação do algoritmo foi realizada em Python (3.12.3), devido à sua versatilidade e popularidade crescente na área de ciência de dados, além das bibliotecas robustas disponíveis para análise e manipulação de imagens. Algumas bibliotecas do Python foram utilizadas para o desenvolvimento do código, Pillow (10.3.0) para fornecer suporte à leitura de imagens, Random aplicada na geração de valores aleatórios fundamentais ao modelo de SVD aplicado, Math para operações matemáticas complexas e Os para acesso a informações da máquina. O algoritmo ficou dividido em funções para melhor documentação e para evitar o desperdício de leitura de variáveis que não serão utilizadas a todo momento na execução.

Foram implementadas algumas funções para a aplicação de operações matriciais básicas, como produto entre matrizes, transposição, produto escalar, norma vetorial, multiplicação escalar e subtração escalar. As funções mais importantes e que ficaram responsáveis por gerir a execução do código foram as funções: decomp_QR, random_SVD e compress_image.

A função decomp_QR, representada na Figura 2, executa a decomposição QR manualmente, garantindo que a base ortonormal seja obtida de forma eficiente para a reconstrução da imagem comprimida.

Figura 2 - Imagem da Função decomp_QR

```
def decomp_QR(A):
    n = len(A)
    m = len(A[0])
    Q = []
    for j in range(m):
        v = [A[i][j] for i in range(n)]
        for q in Q:
            proj = scalar_mult(dot(v, q), q)
            v = vector_sub(v, proj)
        norm_v = norm(v)
        if norm_v > 1e-10:
            q = [vi / norm_v for vi in v]
            Q.append(q)
    Q = transpose(Q)
    return Q
```

Fonte: Elaborada pelo autor (2025)

A função principal, random_SVD, realiza a decomposição da matriz principal da imagem original por meio do método de SVD aleatório. Essa função projeta a matriz da imagem original num subespaço de dimensão reduzida utilizando a matriz aleatória Ômega, para depois obter a base ortonormal pela decomposição QR e então reconstrói uma aproximação da matriz da imagem.

Figura 3 - Imagem da Função random_SVD

```
def random_SVD(A, k):
    m = len(A)
    n = len(A[0])
    Omega = random_mat(n, k)
    Y = matmul(A, Omega)
    Q = qr_decomposition(Y)
    B = matmul(transpose(Q), A)
    A_approx = matmul(Q, B)
    return A_approx
```

Fonte: Elaborada pelo Autor(2025)

Por fim a função *compress_image,* é responsável carregar e converter a imagem original em uma matriz de pixels, para em seguida aplicar a compressão pelo método *randomized SVD* e então salvar a versão comprimida.

Figura 4 - Imagem da Função compress_SVD

```
def compress_image(input_path, output_path, k=10):
    img = Image.open(input_path).convert("L")
    A = [[img.getpixel((x, y)) for x in range(img.width)] for y in range(img.height)]
    A_compressed = random_SVD(A, k)
    salva_img(A_compressed, output_path)
```

Fonte: Elaborada pelo autor (2025)

Para facilitar a demonstração do algoritmo, foi desenvolvida uma interface com auxílio do framework Flask (3.1.1) responsável por fazer a conexão do arquivo de compressão com a interface feita. A interface foi desenvolvida com a linguagem HTML e CSS, nela o usuário é capaz de inserir uma imagem e em seguida obter sua versão comprimida e os dados da compressão.

Após a devida implementação do código foram avaliadas as imagens comprimidas utilizando métricas de avaliação, por ser uma imagem a mais básica métrica de avaliação foi visual e em seguida, foram comparadas as imagens originais com suas versões pós-compressão.

Para isso, foram utilizados métodos avaliativos como o índice de *Bitrate*, sendo esta uma medida da taxa de dados que determina a qualidade de um arquivo (GOGONI, 2021). Quanto menor for o Bitrate da imagem, mais compactada a imagem está. O emprego da taxa de bits é dado por:

$$Bitrate = \frac{Tamanho\ do\ arquivo\ comprimido\ em\ bits}{Tamanho\ da\ imagem\ em\ pixels}$$

Outro método de avaliação aplicado como parâmetro de avaliação foi o índice de compressão (*Compression Ratio*), sendo esta baseada na medida que calcula o quanto o tamanho original da imagem foi reduzido para o tamanho compresso. Uma imagem cujo o CR deu 6 indica que esta está seis vezes menor que a imagem original.

$$CR = \frac{Tamanho\ original}{Tamanho\ comprimido}$$

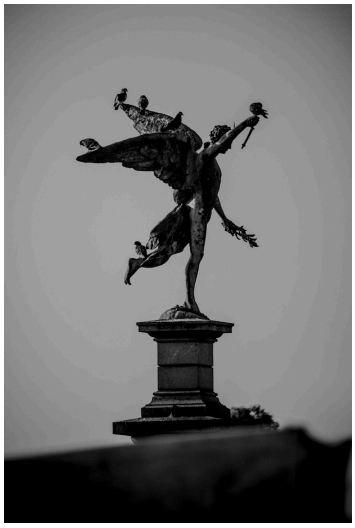
Através destas métricas foi possível mensurar o funcionamento qualitativo do algoritmo escolhido, mediante à avaliação destes foi determinada a viabilidade do SVD Aleatório como uma técnica eficiente e escalável para compressão de grandes volumes de dados, como imagens de alta resolução.

4 RESULTADOS

Nesta seção serão apresentados os principais resultados da implementação e avaliação do algoritmo de compressão de imagens pelo método Randomized SVD. Foi aplicado diferentes variáveis de controle (*k*) com intuito de analisar o impacto causado na compressão da imagem, bem como as métricas de *Bitrate* e *Compression Ratio* para avaliar a qualidade visual e o tamanho final dos arquivos.

O código aplicado trabalha com escala de cinza, então as imagens testadas passaram por um filtro antes de serem comprimidas, além disso foram aplicadas em formatos não compactados, como Bitmap, para que os valores resultantes não fossem comprometidos por um modelo já compactado. Ao total foram realizadas um total de 6 execuções do código, abaixo, serão apresentados os resultados mais relevantes.

Figura 5 - Imagem original de Estátua



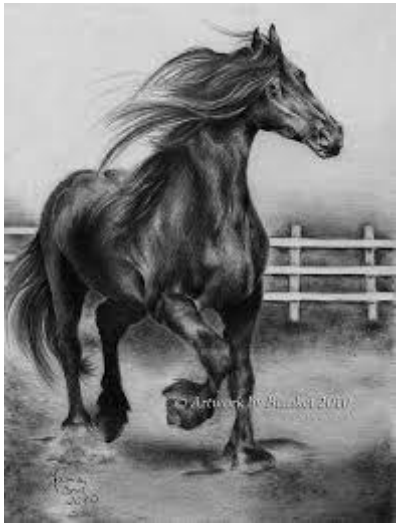
Fonte:Lopez(2025)

Figura 6 - Imagem de Estátua após a compressão, com k = 50

Fonte: Elaborada pelo autor (2025)

A Figura 5 representa a imagem em formato .bmp original é visivelmente mais nítida, enquanto a Figura 6 é versão comprimida possuí um tipo de “chiado”. O resultado do índice de compressão apresentou o valor 71.64, resultado extremamente alto, demonstrando que o tamanho da versão comprimida é 70 vezes menor que a original. O Bitrate da imagem comprimida retornou um valor abaixo de 0.5, o que demonstra uma enorme perda da qualidade original da imagem. Entretanto, ainda é possível distinguir alguns elementos da imagem mesmo após a compressão.

Figura 7 - Imagem original de um cavalo e uma cerca



Fonte: Sanagapati (2022)

Figura 8 - Imagem comprimida de um cavalo e uma cerca, com k = 50

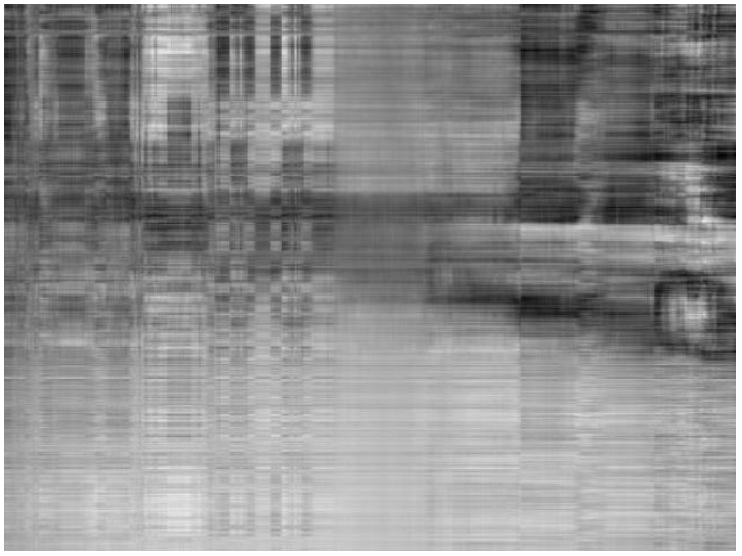
Fonte: Elaborada pelo autor (2025)

Neste segundo teste, a imagem original, representada na figura 7, possui uma resolução menor, a variável de controle se manteve em 50, porém os resultados foram mais positivos. O índice de compressão ficou em 13.74, o que ainda é uma redução muito grande da imagem original, porém a taxa de bitrate resultou em 1.76, o que demonstra que a imagem da Figura 8 manteve uma boa qualidade visual e possui uma compreensão razoável.

Figura 9 - Imagem com filtro cinza de um carro no quintal

Fonte: Sanagapati (2022)

Figura 10 - Imagem comprimida de um carro no quintal, com k = 10



Fonte: Elaborada pelo autor (2025)

O terceiro teste foi realizado com uma imagem colorida, para isso a imagem foi passada por um filtro cinza, entretanto a variável de controle aplicada possui o valor 10, o que manteve pouquíssimos componentes da imagem original, resultando numa imagem borrada e com baixa qualidade. O índice de compressão resultante foi de 22.25, enquanto o bitrate resultou em 1.08, muito próximo de uma imagem de baixa qualidade e com difícil compreensão. Vale ressaltar que as métricas são aplicadas na imagem após ser convertida para a escala de cinza, não imagem original colorida.

Após o fim da execução das compressões foram agrupados todos os dados de cada imagem, antes e depois da compressão. Os resultados obtidos foram organizados e apresentados na Tabela 1.

Tabela 1 — Resultados da compressão das imagens utilizando SVD

Imagem	**Tamanho Original (kb)**	**Tamanho Comprimido (kb)**	**Compression Ratio**	**Bitrate (bpp)**	**Valor de k**
Bicicleta	900	60	18.41	1.3	10
Estátua	39335	549	71.64	0.34	50
Cavalo	148	10.8	13.74	1.76	50
Carro	900	40.4	22.25	1.08	10
Coruja	23115	517	44.65	0.54	10
Gato	52900	1430	36.91	0.65	90

Fonte: Elaborada pelo autor (2025)

5 CONSIDERAÇÕES FINAIS

Verifica-se que o emprego Randomized SVD se aplicou viável para a compressão de imagens. O uso deste método se mostrou eficaz para a compressão de imagens em escala de cinza de forma não muito custosa. Entretanto, o uso deste demonstrou conseguir manter a qualidade somente em um nível razoável, de forma que o ser humano conseguisse entender o que está na imagem, mas minando a qualidade da imagem a troco de redução de espaço.

Em visto que a compressão de imagem pelo modelo de Randomized SVD foi realizada sem ocorrer a comparação com outro modelos de compressão, não foi possível verificar o desempenho do algoritmo perante condições semelhantes em outros métodos de compressão.

A aplicação deste método possui algumas limitações, caso este seja aplicado em uma imagem comprimida este gera uma imagem com bastante degradação visual, mas com o índice de compressão baixo e com alto Bitrate. Esta limitação, indica que o modelo não é recomendado em arquivos já compactados, como o JPEG, pois este não gerou redução significativa no tamanho do arquivo e acabou comprometendo demais o conteúdo da imagem.

Este modelo também não é aplicável para redução de imagens fora da escala de cinza, não sendo capaz de trabalhar com imagens coloridas em matrizes RGB e vídeos. Limitação esta que surge devido a complexidade de trabalhar com múltiplos canais de cor em imagens coloridas e com o fluxo contínuo de dados presentes em vídeos.

Sugere-se a extensão do algoritmo para suportar compressão de imagens coloridas aplicando o SVD em cada canal de cor e adaptar para o emprego do modelo para vídeos, o que irá exigir uma abordagem mais específica para manter a coerência entre os quadros do vídeo. Tais avanços no modelo poderiam aumentar o potencial do uso deste modelo em diferentes contexto, levando em consideração aplicações em arquivos multimídias e armazenamento de um grande volume de dados.

REFERÊNCIAS

ADOBE. **Saiba mais sobre os arquivos JPEG.** Disponível em: https://www.adobe.com/br/creativecloud/file-types/image/raster/jpeg-file.html. Acesso em: 6 jan. 2025.

BRITO, Rogério Theodoro de. **Compressão de Dados e de Imagens.** 1995. Disponível em: https://www.producao.usp.br/directbitstream/c3b93991-d6ed-4fbd-a7a6-15d6b9f7eb98/907664.pdf. Acesso em: 29 dez. 2024.

BURTON, Steven L.; KUTZ, J. Nathan. ***Data-Driven Science and Engineering: Machine learning, dynamical systems and control.*** Seattle: Cambridge University Press, 2019. Disponível em: http://databookuw.com/. Acesso em: 15 dez. 2024.

CLOUDINARY. ***Image Compression Algorithms.*** Disponível em: https://cloudinary.com/glossary/image-compression-algorithms. Acesso em: 6 jan. 2025.

FELIPPE, Asaphe. **O que é: Algoritmo de Compressão de Imagem.** Disponível em: https://aeroengenharia.com/glossario/o-que-e-algoritmo-de-compressao-de-imagem/. Acesso em: 6 jan. 2025.

GOGONI, Ronaldo. **O que é bitrate e como ele afeta qualidade do streaming?: Descubra o que é bitrate, qual a diferença dele para velocidade de conexão, e como sua redução afeta serviços de streaming.** Tecnoblog. Disponível em: https://tecnoblog.net/responde/o-que-e-bitrate-e-como-ele-afeta-qualidade-do-streaming/. Acesso em: 3 jan. 2025.

JACOBI, Otávio F.; SILVEIRA, Thiago L. T.. **Avaliação do impacto da ordem da DCT em compressão de imagens do tipo JPEG.** 2013. Disponível em: https://tltsilveira.github.io/public/Jacobi,%20OF%20et%20al%20-%20JAI2013.pdf. Acesso em: 27 dez. 2024.

LOPEZ, Gabi. **Escultura dramática de anjo no cemitério de Buenos Aires.** Pexels, 2025. Disponível em: https://www.pexels.com/pt-br/foto/escultura-dramatica-de-anjo-no-cemiterio-de-buenos-aires-32982058/. Acesso em: e9 jul. 2025.

MORTARI, Cezar A.. **Introdução à Lógica.** 3. ed. São Paulo: Unesp, 2001. 390 p. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=6iS4_QGZzHAC&oi=fnd&pg=PA1&dq=introdu%C3%A7%C3%A3o+a+l%C3%B3gica&ots=7Mhj0F_xUH&sig=DE5Dohsd8EEIckdE_HKxhmi-3m0#v=onepage&q&f=false. Acesso em: 12 dez. 2024.

SANAGAPATI, Pavan. ***Images Dataset.*** *Kaggle*. Disponível em: https://www.kaggle.com/datasets/pavansanagapati/images-dataset. Acesso em: 13 jul. 2025.

APÊNDICE A - CÓDIGO-FONTE DISPONÍVEL EM REPOSITÓRIO

O código-fonte completo utilizado neste trabalho está disponível no seguinte repositório:

https://github.com/Kuanbb/TCC_RandomizedSVD

O repositório contém os scripts de compressão de imagem, arquivos de configuração, instruções de uso e exemplos de execução.

O acesso é público e pode ser feito por meio de qualquer navegador com conexão à internet.