UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCELO HAIDER TORRES

Aproximando a Transformada Discreta do Cosseno por Redes Neurais Ternárias

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Thiago Lopes Trugillo da Silveira

Porto Alegre
2023

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Renato Torres e Viviane Torres, e a minha irmã Michelle Torres, a todo esforço e dedicação que me permitiram estar aqui neste momento, e a todo apoio incondicional que me deram ao longo da minha formação.

Agradeço também a todos meus professores, em especial ao professor Thiago Lopes Trugillo da Silveira, que ao longo deste trabalho apoiou-me em todos momentos, sabendo guiar-me impecavelmente à conclusão deste trabalho que sem ele não seria possível.

Por último, agradeço a minha namorada Laura Etchalus, a toda parceria e assistência nos melhores e piores momentos durante esta trajetória.

RESUMO

Transformadas discretas desempenham um papel fundamental em processamento de sinais. Em particular, a transformada discreta de cosseno (DCT) se destaca por ser uma importante ferramenta para compressão de imagens e vídeos. Algoritmos rápidos permitem a aplicação da DCT com complexidade reduzida mas requerem computações em ponto flutuante. Tais condições podem ser proibitivas em aplicações que demandam ação em tempo real ou *hardware* de baixo consumo energético. Diversos trabalhos formulam um problema de otimização em um espaço de busca discreto para aproximar o cômputo da DCT utilizando apenas operações de baixa complexidade aritmética. Geralmente, os coeficientes da transformada aproximada estão em (ou subconjunto de) $\{0, \pm\frac{1}{2}, \pm 1, \pm 2\}$. Este trabalho modela o problema de derivação de transformadas aproximadas através da otimização (treinamento) de uma rede neural artificial (ANN) do tipo *multilayer perceptron* (MLP). São utilizados mecanismos recentes para treinamento de ANNs ternárias de tal forma que aproximações ortogonais para a DCT de 8 pontos que requerem apenas adições/subtrações possam ser derivadas. Duas transformadas conhecidas na literatura são obtidas pela metodologia proposta. Para avaliação, são utilizadas métricas de eficiência de codificação e proximidade com relação à DCT exata. As aproximações para a DCT obtidas são submetidas a um experimento de compressão de imagens similar ao JPEG, onde são avaliadas métricas de qualidade de imagem. Os resultados indicam que a metodologia proposta permite derivar transformadas com alta eficiência de codificação e baixa complexidade aritmética. No futuro, a metodologia proposta será estendida de forma a contemplar outros tamanhos de transformada e outras transformadas discretas lineares.

Palavras-chave: Aproximações para DCT. Transformadas de baixa complexidade aritmética. Compressão de imagens. Redes Neurais Artificiais. *Multilayer perceptron*.

Aproximating the Discrete Cosine Transform by Ternary Neural Networks

ABSTRACT

Discrete transforms play a fundamental role in signal processing. In particular, the discrete cosine transform (DCT) is an essential tool for image and video compression. Fast algorithms allow the application of DCT with reduced complexity but require floating-point computations. Such conditions can be prohibitive in applications that demand real-time response or low-power hardware. Several works formulate an optimization problem in a discrete search space to approximate the computation of the DCT using only operations of low arithmetic complexity. Generally, approximate transform coefficients are in (or a subset of) $\{0, \pm\frac{1}{2}, \pm 1, \pm 2\}$. This work models the approximate transform proposition problem by optimizing (training) a multilayer perceptron (MLP) artificial neural network (ANN). Recent mechanisms for training ternary ANNs are used so that orthogonal 8-point DCT approximations that require only additions/subtractions can be derived. Two well-known transformations in the literature are obtained by the proposed methodology. For evaluation, metrics of coding efficiency and proximity to the exact DCT are used. The obtained DCT approximations are submitted to a JPEG-like image compression experiment, where image quality assessment is performed. The results indicate that the proposed methodology allows for deriving transforms with high coding efficiency and low arithmetic complexity. In the future, the proposed methodology will be extended in order to contemplate other transform sizes and other discrete linear transforms.

Keywords: DCT approximations, Low-complexity transforms, Image compression, Artificial Neural Networks, Multilayer perceptron.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DCT	Transformada Discreta do Cosseno
ANN	Rede Neural Artificial
MLP	*Multilayer Perceptron*
JPEG	*Joint Photographic Experts Group*
PSNR	Relação sinal-ruído de pico
SSIM	Índice de similaridade estrutural
IoT	*Internet of Thing*
HEVC	*High-Efficiency Video Coding*
VVC	*Versatile Video Coding*
DFT	Transformada Discreta de Fourier
ISO	*International Organization for Standardization*
SDCT	DCT Sinalizada
SCA	Abordagem para controle de esparsidade
WDR	Regularizador de discretização de pesos
SGD	*Stochastic Gradient Descent*
MSE	Erro quadrático médio

LISTA DE SÍMBOLOS

$\mathscr{B}$	Conjunto $\{\pm 1\}$
$\mathscr{T}$	Conjunto $\{0, \pm 1\}$
$\mathscr{P}$	Conjunto $\{0, \pm 1, \pm \frac{1}{2}, \pm 2\}$
$\mathbf{I}$	Imagem original
$\mathbf{A}$	Bloco disjunto de uma imagem
$\mathbf{C}$	Matriz de transformação da DCT
$\mathbf{B}$	Bloco transformado
$\oslash$	Divisão de matrizes elemento-a-elemento
$\mathbf{Q}$	Matriz de quantização
$\mathrm{round}(\cdot)$	Função de arredondamento para inteiro
$\mathbf{Q_0}$	Matriz de quantização para $QF = 50$
$\odot$	Multiplicação de matrizes elemento-a-elemento
$\hat{\mathbf{B}}$	Bloco quantizado de uma imagem
$\tilde{\mathbf{B}}$	Bloco dequantizado de uma imagem
$\tilde{\mathbf{A}}$	Bloco reconstruído de uma imagem
$\tilde{\mathbf{I}}$	Imagem reconstruída
QF	Fator de quantização
$\mathbf{T}_{\pm}$	Matriz de transformação de baixa complexidade da SDCT
$\widehat{\mathbf{C}}_{\pm}$	Matriz de transformação da SDCT
$\widehat{\mathbf{C}}$	Matriz de transformação de uma aproximação genérica para a DCT
$\arg\mathrm{opt}(\cdot)$	Argumento do valor ótimo
$g(\cdot, \cdot)$	Função que avalia a otimalidade de uma aproximação genérica para a DCT
$\mathbf{S}$	Matriz diagonal genérica de escala
$\mathbf{T}$	Matriz de transformação genérica de baixa complexidade

$\mathbf{\Omega}$	Matriz de pesos/parâmetros
b	Termo de viés/tendência (*bias*)
$a(\cdot)$	Função de ativação
$L(\cdot, \cdot, \cdot)$	Função de perda
$\tanh(\mathbf{\Theta})$	Matriz de pesos/parâmetros (idealmente) ternários
M	Tamanho do conjunto de dados para treinamento
(X, Y)	Conjunto de dados pareados para treinamento supervisionado
$\| \cdot \|_2$	Norma euclideana
$D(\cdot)$	Regularizador de discretização de pesos
$J(\cdot)$	Função objetivo
α	Controlador de esparsidade de pesos
λ_1	Ponderador do regularizador de discretização de pesos
m	Tamanho da amostra (*batch size*)
$\mathcal{N}(\cdot, \cdot)$	Distribuição Normal
$\boldsymbol{\mathcal{T}}$	Matriz de pesos (idealmente) ternários otimizada
$\mathbf{\Sigma}$	Matriz diagonal otimizada
$\mathrm{diag}(\cdot)$	Função que retorna a diagonal de uma matriz ou uma matriz diagonal
$O(\cdot)$	Regularizador de ortogonalizadade
$\| \cdot \|_F$	Norma de Frobenius
λ_2	Ponderador do regularizador de ortogonalidade
$\epsilon(\cdot, \cdot)$	Erro total de energia
$\mathrm{tr}(\cdot)$	Função traço
$\mathbf{C}_g(\cdot)$	Ganho de codificação de transformada
$\mathbf{R}_\mathbf{x}$	Matriz de Covariância
$\mathrm{su}(\cdot)$	Função de soma dos elementos de uma matriz
$\eta(\cdot)$	Eficiência da transformada

$\Delta(\cdot)$	Erro de trucamento para uma matriz
$\delta(\cdot,\cdot)$	Erro de truncamento entre 2 diagonais
$\mathrm{PSNR}(\cdot,\cdot)$	PSNR de uma imagem
$\mathrm{SSIM}(\cdot,\cdot)$	SSIM de um bloco de imagem
$\mu(\cdot)$	Média aritmética dos elementos de $(\cdot)$
$\sigma(\cdot)$	Desvio padrão dos elementos de $(\cdot)$
$\mathrm{cov}(\cdot,\cdot)$	Covariância entre duas matrizes
$\mathrm{bpp}(\cdot)$	*Bits* por pixel de uma imagem
$\mathrm{sz}(\cdot)$	Tamanho de uma imagem (em *bytes*)

SUMÁRIO

1 INTRODUÇÃO

Este capítulo motiva a proposta de transformadas aproximadas para compressão de imagens em aplicações em tempo real e de baixo consumo energético. Este capítulo também pontua os objetivos deste trabalho e apresenta a organização do restante do texto.

1.1 Motivação

Algoritmos para o processamento de sinais com baixa complexidade aritmética são fundamentais para aplicações emergentes como as de Internet das Coisas (IoT) (CAMPOBELLO et al., 2017; MA et al., 2015). Geralmente, dispositivos IoT atuam em tempo real, têm baixo poder computacional e requerem baixo consumo de energia (CALLEBAUT et al., 2021). Aplicações que armazenam ou transferem imagens em dispositivos com recursos computacionais restritos dependem de técnicas de compressão para serem eficientes (HAMZA; HASSAN; PATIL, 2019; KOUADRIA et al., 2013).

A transformada discreta do cosseno (DCT) (AHMED; NATARAJAN; RAO, 1974) é uma importante ferramenta para compressão de imagens. A DCT tem formulação independente dos dados e tem a capacidade de agrupar a energia do sinal em poucos coeficientes espectrais (LIANG; TRAN, 2000; HAWEEL; EL-KILANI; RAMADAN, 2014). A DCT é uma aproximação assintótica para a transformada de Karhunen-Loève (AHMED; NATARAJAN; RAO, 1974) quando o sinal pode ser modelado como um processo de Markov I altamente autocorrelacionado (BRITANAK P. YIP, 2007). Imagens naturais satisfazem essa propriedade. Padrões de compressão de imagens – como *Joint Photographic Experts Group* (JPEG) (PENNEBAKER; MITCHELL, 1992) – e vídeo – como o *High Efficiency Video Coding* (HEVC) (ROMA; SOUSA, 2007) e o *Versatile Video Coding* (VVC) (ZHAO et al., 2021) – utilizam a DCT como peça chave. A DCT de tamanho $N = 8$ é adotada no JPEG, enquanto transformadas de comprimento maior são adotadas em padrões de compressão de vídeo como HEVC e VVC. Aproximações inteiras para a DCT são utilizadas nesses dois últimos (ROMA; SOUSA, 2007; ZHAO et al., 2021).

Algoritmos rápidos para a DCT (FEIG; WINOGRAD, 1992; CHEN; SMITH; FRALICK, 1977, 1977; LOEFFLER; LIGTENBERG; MOSCHYTZ, 1989) reduzem o custo de transformação se comparado àquele de sua definição. Entretanto, ainda assim, a aplicação da DCT pode ser onerosa em situações onde há reduzido poder computacional e baixo consumo energético (COELHO et al., 2018; HAWEEL, 2001). Em aplicações como

as de IoT, frequentemente, algum tipo de imprecisão na computação é tolerável (ALARIFI et al., 2020). Nesse contexto, surgem aproximações para a DCT.

Aproximações para a DCT têm matrizes de transformação de baixa complexidade com coeficientes, geralmente, em $\mathscr{P} = \{0, \pm 1, \pm\frac{1}{2}, \pm 2\}$ ou um subconjunto de $\mathscr{P}$ ($\mathscr{B} = \{\pm 1\}$ ou $\mathscr{T} = \{0, \pm 1\}$). Essas aproximações são ditas livre de multiplicações e oferecem um compromisso entre custo aritmético e eficiência em codificação (BAYER; CINTRA, 2012; SILVEIRA et al., 2021; COELHO et al., 2018; CINTRA; BAYER, 2011). Alguns trabalhos usam funções como a do sinal ou a de arredondamento dos elementos da DCT para proposição de aproximações (HAWEEL, 2001; CINTRA; BAYER, 2011; BAYER; CINTRA, 2010). Relações entre a DCT e transformadas binárias também podem ser exploradas (BOUGUEZEL; AHMAD; SWAMY, 2013). Outros trabalhos buscam por matrizes de transformação "ótimas" sob algum critério a partir de um espaço de busca atrelado a $\mathscr{B}$, $\mathscr{T}$ ou $\mathscr{P}$ (SILVEIRA et al., 2021; CANTERLE et al., 2020; OLIVEIRA et al., 2018; COELHO et al., 2018). Embora factíveis, procedimentos de busca exaustiva não escalam conforme o tamanho N da transformada aumenta. Por exemplo, buscar exaustivamente uma matriz de transformação ótima com $N \times N$ coeficientes independentes em $\mathscr{T}$ é intratável com a tecnologia de hoje (BAYER et al., 2013).

Ao melhor do conhecimento dos autores, uma forma pouco explorada para proposição de transformadas é via otimização (treinamento) de uma rede neural artificial (ANN). Velik (2008) mostrou que é possível modelar uma transformada discreta – neste caso, a transformada discreta de Fourier (DFT) – como uma ANN do tipo *perceptron* (GOODFELLOW; BENGIO; COURVILLE, 2016, pág. 168). Velik (2008) atribui valores aos parâmetros (pesos) de uma *perceptron* de uma camada que mimetiza a DFT. O objetivo de Velik (2008) é realizar computações paralelas em várias entradas simultaneamente. Não há treinamento dessa ANN. Akhtar (2021) estende a ideia de Velik (2008) e treina uma ANN do tipo *multilayer perceptron* (MLP) (GOODFELLOW; BENGIO; COURVILLE, 2016, pág. 168) para realização da cômputo da DFT sobre vetores de dados reais.

O presente trabalho explora a relação existente entre transformadas discretas lineares e ANNs do tipo *perceptron* e MLP e utiliza uma abordagem recente para o treinamento de ANNs com pesos (idealmente) em $\mathscr{T}$ (DENG; ZHANG, 2022). Este trabalho investiga formulações para o treinamento da MLP visando a proposição de aproximações de baixo custo aritmético para a DCT de tamanho $N = 8$. Tal abordagem, em princípio, pode ser aplicada para qualquer tamanho N de transformada.

1.2 Objetivos

O objetivo geral desse trabalho é propor aproximações para a DCT com baixa complexidade aritmética se valendo da relação entre transformadas discretas lineares e MLPs. Para isso, o trabalho utiliza ferramentas modernas para treinamento supervisionado de MLPs com baixa complexidade aritmética.

Os objetivos específicos desse trabalho incluem:

1. Revisar aproximações de baixa complexidade aritmética para a DCT;
2. Identificar parametrizações para aproximações da DCT;
3. Identificar abordagens para treinamento de ANNs de baixa complexidade aritmética;
4. Modelar aproximações de baixa complexidade aritmética para a DCT como MLPs;
5. Treinar e avaliar MLPs com diferentes (hiper-)parametrizações.

1.3 Organização dos Capítulos

O restante deste documento está organizado como segue.

O Capítulo 2 discute aspectos fundamentais para o entendimento do escopo deste trabalho. Mais precisamente, a Seção 2.1 apresenta uma visão geral do papel da DCT de tamanho $N = 8$ no padrão JPEG. A Seção 2.2 lista brevemente abordagens para redução de complexidade de aplicação da DCT de forma exata.

O Capítulo 3 apresenta abordagens para redução do custo de implementação, por aproximação, de transformadas discretas e ANNs. Em particular, a Seção 3.1 revisa importantes aproximações de baixa complexidade aritmética para a DCT. A Seção 3.2 discute métodos para construção de ANNs com pesos binários ou ternários.

O Capítulo 4 apresenta modelagens da DCT e aproximações como uma ANN do tipo *perceptron* ou MLP. A Seção 4.1 mostra como otimizar uma ANN do tipo *perceptron* para mimetizar a DCT. A Seção 4.2 discute aspectos como mudança da arquitetura da ANN para duas camadas (MLP) e a escolha de uma função objetivo, com termos de perda e regularização, adequada para aproximar a DCT.

O Capítulo 5 apresenta os experimentos realizados e resultados obtidos neste trabalho. A Seção 5.1 especifica os recursos e ferramentas computacionais utilizados. A Seção 5.2 discute o mecanismo de treinamento da ANN adotado e a escolha dos hiper-

parâmetros da modelagem realizada. Aproximações para a DCT derivadas da modelagem introduzidas na Seção 4.2 são apresentadas e avaliadas na Seção 5.3. A Seção 5.4 descreve um experimento de compressão de imagens com as transformadas aproximadas obtidas e discute os resultados. Comparações com outras aproximações para a DCT são feitas nas Seções 5.3 e 5.4.

Por fim, o Capítulo 6 apresenta as conclusões e elenca possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo revisa a importância da DCT em compressão de imagens e identifica abordagens eficientes para sua computação. A Seção 2.1 explica brevemente o padrão JPEG, enquanto a Seção 2.2 lista algoritmos rápidos renomados para o cômputo da DCT.

2.1 O Padrão JPEG

O *Joint Photographic Experts Group* da *International Standards Organization* (ISO) estabeleceu, na década de 1990, o primeiro padrão de compressão com perdas para imagens de tom contínuo (WALLACE, 1991). O padrão JPEG, descrito brevemente nesta seção, é amplamente utilizado até hoje. Padrões de codificação de vídeo modernos – como o HEVC e o VVC – também utilizam a DCT como parte fundamental para descorrelação e compressão de dados (ROMA; SOUSA, 2007; ZHAO et al., 2021).

O padrão JPEG compreende um processo de codificação por transformada aplicada a blocos da imagem. A DCT é utilizada para transformar blocos da imagem do domínio espacial para o domínio espectral. Como a DCT agrupa a maior parte da energia do sinal nos primeiros coeficientes espectrais (AHMED; NATARAJAN; RAO, 1974), é possível descartar coeficientes de baixa magnitude sem que haja perda significativa da qualidade de imagem (WALLACE, 1991).

Sem perda de generalidade, uma imagem em tons de cinza[1] pode ser representada por uma matriz $\mathbf{I}$ de tamanho $L \times A$. Frequentemente, imagens têm *pixels* de 8 *bits*, o que permite armazenar valores inteiros no intervalo [0, 255]. A primeira etapa da compressão pelo padrão JPEG consiste na quebra da imagem $\mathbf{I}$ em $\frac{L \times A}{N^2}$ blocos disjuntos $\mathbf{A}$ de tamanho $N \times N$ *pixels* com $N = 8$. Aqui, por simplicidade, assume-se L e A como inteiros múltiplos de N.

Cada bloco é transformado conforme

$$\mathbf{B} = \mathbf{C} \cdot \mathbf{A} \cdot \mathbf{C}^{\top}, \tag{2.1}$$

em que $\mathbf{C}$ é a matriz de transformação da DCT de tamanho $N = 8$. Os (i, j)-ésimos

[1] Imagens coloridas são transformadas de forma similar, em que cada canal é submetido individualmente a um processo similar ao descrito neste texto.

elementos de $\mathbf{C}$ são dados por

$$c_{i,j} = u_i \cos\left(\frac{i(2j+1)\pi}{2N}\right), \tag{2.2}$$

em que $i, j = 0, \ldots, N-1$ e

$$u_i = \begin{cases} 1/\sqrt{N}, & \text{se} \quad i = 0 \\ \sqrt{2/N}, & \text{se} \quad i \neq 0 \end{cases}. \tag{2.3}$$

O processo em (2.1) não gera perda de informação. A etapa de quantização, por outro lado, induz a remoção de coeficientes de baixa magnitude e pode ser aplicada por

$$\hat{\mathbf{B}} = \text{round}(\mathbf{B} \oslash \mathbf{Q}), \tag{2.4}$$

em que $\oslash$ é a divisão matricial aplicada elemento a elemento e $\text{round}(\cdot)$ é a função de arredondamento para inteiro, também aplicada elemento a elemento (OLIVEIRA et al., 2017).

A matriz $\mathbf{Q}$ é chamada matriz de quantização e comumente é dada por (PENNEBAKER; MITCHELL, 1992)

$$\mathbf{Q} = \left\lfloor \frac{S \cdot \mathbf{Q_0} + 50}{100} \right\rfloor, \tag{2.5}$$

em que

$$\mathbf{Q_0} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}, \tag{2.6}$$

e

$$S = \begin{cases} 5000/QF, & \text{se}\, QF < 50 \\ 200 - 2QF, & \text{caso contrário} \end{cases}. \tag{2.7}$$

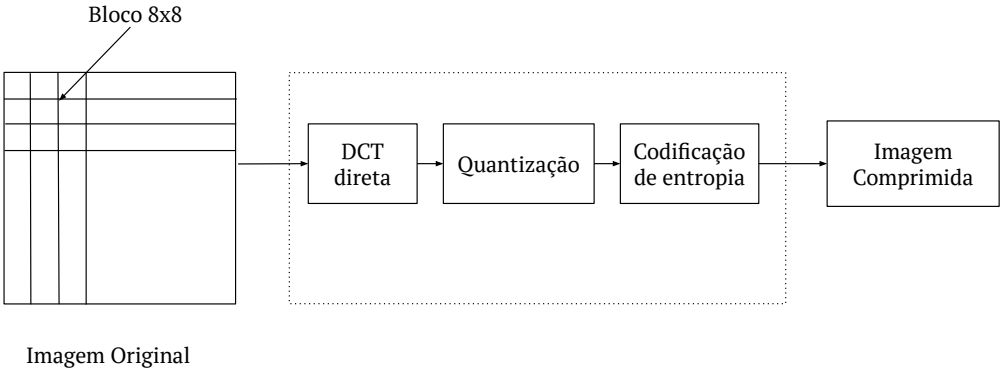


Figura 2.1 – Processo de compressão pelo Padrão JPEG. Ilustração inspirada em Wallace (1991).

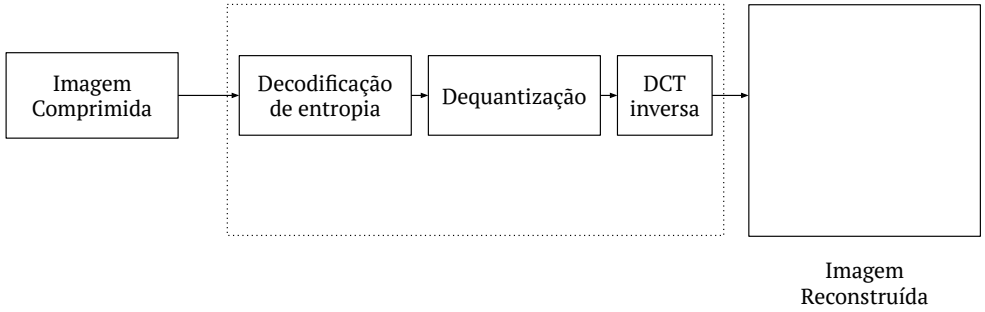


Figura 2.2 – Processo de descompressão pelo padrão JPEG. Ilustração inspirada em Wallace (1991).

O parâmetro QF corresponde a um fator de qualidade e varia no intervalo inteiro [0, 100].

Após quantizados, os blocos $\tilde{\mathbf{B}}$ são submetidos a uma etapa de codificação de entropia sem perdas, permitindo o armazenamento ou transmissão da imagem comprimida (PENNEBAKER; MITCHELL, 1992).

O processo inverso ao descrito anteriormente permite reconstruir uma aproximação para a imagem $\mathbf{I}$. O primeiro passo da descompressão pelo padrão JPEG consiste em decodificar os blocos $\hat{\mathbf{B}}$ e, posteriormente, dequantizá-los. A dequantização é dada por

$$\tilde{\mathbf{B}} = \mathbf{Q} \odot \hat{\mathbf{B}}, \tag{2.8}$$

em que $\odot$ é a multiplicação matricial aplicada elemento a elemento (WALLACE, 1991).

Por fim, aplica-se a transformação inversa que, devido à ortogonalidade da DCT

– isto é $\mathbf{C}^{-1} = \mathbf{C}^{\top}$ –, é dada por

$$\tilde{\mathbf{A}} = (\mathbf{C}^{\top} \cdot \tilde{\mathbf{B}} \cdot \mathbf{C}). \tag{2.9}$$

O rearranjo dos $\frac{L \times A}{N^2}$ blocos $\tilde{\mathbf{A}}$ gera uma imagem $\tilde{\mathbf{I}}$ que aproxima $\mathbf{I}$. A similaridade entre as imagens $\mathbf{I}$ e $\tilde{\mathbf{I}}$ depende de QF e pode ser mensurada por métricas de qualidade de imagem. Algumas dessas métricas, bem como sua aplicação na avaliação de compressão de imagens, serão discutidas na Seção 5.4.

Os processos de compressão e descompressão descritos nesta seção são ilustrados nas Figuras 2.1 e 2.2.

2.2 Algoritmos Rápidos para a DCT

Algoritmos rápidos para a DCT objetivam diminuir o número de operações aritméticas envolvidos em sua aplicação. Algoritmos rápidos para a DCT exploram sua estrutura e se valem de fatoração em matrizes esparsas para sua computação exata (CHEN; SMITH; FRALICK, 1977; ARAI; AGUI; NAKAJIMA, 1988; WANG, 1984; LEE, 1984; FEIG; WINOGRAD, 1992; LOEFFLER; LIGTENBERG; MOSCHYTZ, 1989).

A aplicação da DCT de 8 pontos sobre um vetor unidimensional, utilizando a matriz de transformação dada pela Equação (2.2), exige 56 adições e 64 multiplicações. Note que a matriz de transformação da DCT contém valores reais, e, portanto, as operações de adição e multiplicação necessárias para sua computação exigem aritmética em ponto flutuante. O algoritmo de Feig and Winograd (1992) é implementado com 28 adições e 22 multiplicações. O algoritmo rápido de Chen, Smith and Fralick (1977), por outro lado, requer 26 adições e 16 multiplicações. O algoritmo de Arai, Agui and Nakajima (1988) conta com 29 adições e 13 multiplicações. Já os algoritmos de Wang (1984) e de Lee (1984) são implementados com apenas 29 adições e 12 multiplicações. Por fim, o algoritmo de Loeffler, Ligtenberg and Moschytz (1989) requer 29 adições e 11 multiplicações.

O cômputo exato da DCT através de algoritmos rápidos é uma área madura e o algoritmo de Loeffler, Ligtenberg and Moschytz (1989) detém a complexidade multiplicativa mínima. Abordagens recentes avançam a área de computação aproximada, oferecendo um compromisso entre complexidade aritmética e desempenho de codificação (SILVEIRA et al., 2021; COELHO et al., 2019). Tais abordagens serão melhores vistas na Seção 3.1.

3 TRABALHOS RELACIONADOS

Este capítulo discute abordagens para a redução do custo de aplicação de transformadas discretas e ANNs por computação aproximada. Em especial, a Seção 3.1 revisa relevantes aproximações de baixo custo para a DCT. A Seção 3.2 discute técnicas para treinamento de ANNs com pesos binários ou ternários. A relação entre DCT e ANNs do tipo *perceptron* é indicada nesta seção e aprofundada no Capítulo 4.

3.1 Transformadas de Baixo Custo

Embora os algoritmos rápidos listados na Seção 2.2 reduzam o custo de aplicação da DCT, estes ainda requerem computações com valores em ponto flutuante. Nesse sentido, diferentes abordagens objetivam a redução de custo aritmético em troca de um cálculo aproximado da transformação dos dados (ALMURIB; KUMAR; LOMBARDI, 2018). Transformadas aproximadas, cujos coeficientes são valores inteiros, são amplamente adotadas em padrões de compressão de vídeo, como no HEVC e VVC (ROMA; SOUSA, 2007; ZHAO et al., 2021). De uma forma geral, as aproximações utilizadas nesses padrões ainda requerem multiplicações (inteiras). Multiplicações inteiras podem ser decompostas em somas e deslocamentos de *bits* (RADÜNZ et al., 2023; BRITANAK P. YIP, 2007).

Transformadas cujos coeficientes podem ser escritos como potências de dois têm baixíssima complexidade aritmética e são aplicáveis onde certas imprecisões podem ser toleradas, como em sistemas de baixo consumo energético, de tempo real ou ainda em dispositivos de IoT (ALARIFI et al., 2020; KOUADRIA et al., 2013). Em especial, matrizes de transformação cujos coeficientes estão em $\mathscr{B} = \{\pm 1\}$ ou $\mathscr{T} = \{0\} \cup \mathscr{B}$ podem ser implementadas com adições e subtrações apenas. Se esse conjunto é estendido para $\mathscr{P} = \{\pm\frac{1}{2}, \pm 2\} \cup \mathscr{T}$, então pode-se computar multiplicações e divisões por 2 utilizando deslocamentos de *bits* simples. O presente trabalho foca especialmente em transformadas de tamanho $N = 8$ com coeficientes em $\mathscr{B}$ ou $\mathscr{T}$.

A DCT sinalizada (SDCT) (HAWEEL, 2001) é uma importante aproximação para a DCT. A SDCT é construída através da aplicação da função sinal a cada um dos elementos da matriz de transformação da DCT:

$$\mathbf{T}_{\pm} = \operatorname{sign}(\mathbf{C}), \tag{3.1}$$

em que

$$\text{sign}(x) = \begin{cases} +1, & \text{se} \quad x > 0 \\ 0, & \text{se} \quad x = 0 \\ -1, & \text{se} \quad x < 0 \end{cases} . \tag{3.2}$$

A matriz de transformação $\mathbf{T}_{\pm}$ é livre de multiplicações e tem elementos em $\mathscr{B}$. A SDCT normalmente é escrita como

$$\widehat{\mathbf{C}}_{\pm} = s \cdot \mathbf{T}_{\pm}, \tag{3.3}$$

em que

$$s = \frac{1}{\sqrt{N}} \tag{3.4}$$

é um fator de normalização aplicado a cada uma de suas funções base. Em aplicações de compressão de imagens, o escalar s pode ser computado junto da etapa de quantização – o que faz com que a complexidade de aplicação da transformada $\widehat{\mathbf{C}}_{\pm}$ seja a mesma que de $\mathbf{T}_{\pm}$ (HAWEEL, 2001). Diferentemente da DCT, a SDCT é uma transformada não-ortogonal – o que dificulta o compartilhamento de implementações eficientes para transformações direta e inversa (JRIDI; ALFALOU; MEHER, 2015).

Assim como Haweel (2001) aplica a função sinal sobre os elementos da matriz de transformação da DCT, Cintra and Bayer (2011) aplicam a função de arredondamento. Uma série de transformadas de baixo custo é proposta por Bouguezel-Ahmad-Swamy (BAS) (BOUGUEZEL; AHMAD; SWAMY, 2013; BOUGUEZEL; AHMAD; SWAMY, 2011; BOUGUEZEL; AHMAD; SWAMY, 2008; BOUGUEZEL; AHMAD; SWAMY, 2009; BOUGUEZEL; AHMAD; SWAMY, 2010) as quais foram recentemente unificadas em uma família de transformadas (CANTERLE et al., 2020). Outras famílias de aproximações para a DCT relevantes incluem Tablada, Bayer and Cintra (2015), Silveira et al. (2021), Coelho et al. (2018).

Muitos trabalhos buscam por matrizes de transformação "ótimas" de acordo com algum critério (SILVEIRA et al., 2021; CANTERLE et al., 2020; OLIVEIRA et al., 2018; COELHO et al., 2018). Geralmente, otimiza-se

$$\widehat{\mathbf{C}}^{*} = \underset{\widehat{\mathbf{C}}}{\arg\text{opt}}\ g(\mathbf{C}, \widehat{\mathbf{C}}), \tag{3.5}$$

em que $g(\cdot, \cdot)$ é uma função que pode avaliar a similaridade da matriz de transformação candidata com a DCT (CINTRA; BAYER, 2011; BRITANAK P. YIP, 2007; FONG;

CHAM, 2012), sua capacidade de codificação (TAKALA; NIKARA, 2001; FONG; CHAM, 2012) e/ou sua complexidade aritmética. Tais figuras de mérito serão descritas no Capítulo 5.

Trabalhos que seguem essa abordagem consideram um espaço de busca factível onde as matrizes de transformação de baixa complexidade têm elementos em $\mathscr{B}$, $\mathscr{T}$ ou $\mathscr{P}$. Bayer et al. (2013) considera o espaço de busca de todas as matrizes de tamanho $N = 4$ em $\mathscr{T}$ e obtêm uma transformada ortogonal ótima de acordo com uma métrica de erro (CINTRA; BAYER, 2011). Tal abordagem considera busca exaustiva, o que não escala conforme o tamanho da transformada aumenta (por exemplo, $N = 8$). Em outros trabalhos (SILVEIRA et al., 2021; CANTERLE et al., 2020; OLIVEIRA et al., 2018; COELHO et al., 2018), o espaço de busca é dado em função de $\mathscr{B}$, $\mathscr{T}$ ou $\mathscr{P}$ e uma reparametrização da DCT em $K \ll N \times N$ valores.

Frequentemente, essas e outras aproximações para a DCT podem ser escritas como

$$\widehat{\mathbf{C}} = \mathbf{S} \cdot \mathbf{T}, \tag{3.6}$$

em que $\mathbf{T}$ é uma matriz ortogonal de baixa complexidade aritmética e $\mathbf{S}$ é uma matriz diagonal com elementos reais. Os elementos de $\mathbf{T}$ estão em $\mathscr{B}$, $\mathscr{T}$ ou $\mathscr{P}$ e a matriz $\mathbf{S}$ pode ser computada por (SILVEIRA et al., 2021; CANTERLE et al., 2020)

$$\mathbf{S} = \sqrt{[\mathbf{T} \cdot \mathbf{T}^\top]^{-1}}, \tag{3.7}$$

em que $\sqrt{\cdot}$ calcula a raiz quadrada de cada elemento separadamente da matriz. Assim como o escalar s em (3.4), em aplicações de compressão de imagens, a matriz diagonal $\mathbf{S}$ pode ser computada durante a quantização (BAYER; CINTRA, 2012; JRIDI; ALFALOU; MEHER, 2015; SILVEIRA et al., 2021; RADÜNZ et al., 2023; OLIVEIRA et al., 2017).

3.2 Redes Neurais Artificiais de Baixo Custo

Velik (2008) propõe modelar a DFT como uma ANN do tipo *perceptron* e atribuir os valores desejados aos parâmetros. O objetivo de Velik (2008) não é treinar a ANN, mas sim realizar computações de várias entradas simultaneamente. Akhtar (2021) propõe modelar a DFT como uma ANN do tipo MLP de duas camadas. Esta modelagem conta com uma camada oculta que separa e otimiza as partes real e imaginária da DFT.

O metodo proposto por Akhtar (2021) faz uso função de ativação sigmóide e conta com passos de normalização e denormalização dos dados durante o treinamento. Ao melhor do conhecimento dos autores, treinar uma *perceptron* ou MLP *de baixa complexidade* que aproxima a DCT é uma abordagem ainda não explorada na literatura.

ANNs compõem o estado da arte em diversas aplicações envolvendo reconhecimento de padrões, processamento de imagens e visão computacional (CLAY et al., 2021; KRONER et al., 2020; REDMON et al., 2016). Atualmente, há um grande interesse na implementação de ANNs em dispositivos portáteis e de baixo consumo energético (DENG; ZHANG, 2022). Uma possível abordagem consiste em construir ANNs com pesos em $\mathscr{B}$ ou em $\mathscr{T}$. A inclusão de pesos com valor 0 permite que ANNs se tornem mais esparsas e menos complexas (DENG; ZHANG, 2022).

Há duas principais frentes de pesquisa para "compressão de ANNs": quantização e poda. A quantização de ANNs visa treinar uma rede com pesos ou funções de ativação com poucos *bits* e comumente utilizam fatores de escala em cada uma de suas camadas (DENG; ZHANG, 2022). Muitas abordagens exploram quantização binária de ANNs (SOUDRY; HUBARA; MEIR, 2014; COURBARIAUX; BENGIO; DAVID, 2015; HUBARA et al., 2016; COURBARIAUX; BENGIO; DAVID, 2015; LIN; ZHAO; PAN, 2017; TANG; HUA; WANG, 2017) enquanto outras permitem quantização ternária (ALEMDAR et al., 2017), 10.1007/978-3-319-46493-$0_3 2, 8579080$.

Por outro lado, a poda objetiva a construção de ANNs eficientes fazendo com que pesos no domínio real se tornem mais esparsos (DENG; ZHANG, 2022). De uma forma ou de outra, a ideia principal se relaciona com a construção de algoritmos rápidos para transformadas. Algumas abordagens consideram a decomposição de camadas da ANN em matrizes esparsas (DENIL et al., 2013; KIM et al., 2015; REN et al., 2018).

Algumas técnicas para treinamento de ANNs (DENG; ZHANG, 2022; TANG; HUA; WANG, 2017; DARABI et al., 2020) incentivam a otimização de parâmetros binários ou ternários através de um termo de regularização. A vantagem de utilizar tais abordagens é que as funções de ativação e o funcionamento básico da retropropagação da ANN não são alterados. Deng and Zhang (2022) propõem uma abordagem para controle de esparsidade (SCA) da ANN que limita o domínio dos pesos em $(-1, 1)$ através da parametrização destes pela tangente hiperbólica. A SCA é capaz de controlar a esparsidade dos pesos por meio dessa parametrização e de um regularizador de discretização dos pesos. A SCA usa poda e quantização para que camadas de uma ANN possam ser descritas apenas com valores ternários (DENG; ZHANG, 2022).

O presente trabalho considera a SCA, a qual é discutida em maiores e detalhes na Seção 4.2, para a proposição de aproximações de baixa complexidade aritmérica para a DCT.

4 METODOLOGIA PROPOSTA

Este capítulo apresenta a metodologia proposta. A Seção 4.1 mostra como otimizar uma ANN do tipo *perceptron* para mimetizar uma transformada discreta linear – em especial a DCT. A Seção 4.2 discute como ampliar a arquitetura da ANN para duas camadas (MLP) e a escolha de uma função objetivo, com termos de perda e regularização, adequada para aproximar a DCT.

4.1 Computando a DCT com uma ANN do tipo *perceptron*

Como argumentado por Velik (2008), pode-se modelar uma transformada discreta linear – como a DFT – como uma ANN do tipo *perceptron* de uma única camada. Ressalta-se que Velik (2008) não treina a *perceptron* para mimetizar a DFT, mas sim atribui os coeficientes da DFT aos parâmetros dessa ANN. O objetivo de Velik (2008) é explorar ferramentas de aprendizado de máquina para processamento de múltiplas entradas simultaneamente. Akhtar (2021) treina uma MLP – com uma camada oculta – que otimiza as partes real e imaginária da DFT separadamente. O texto a seguir relaciona uma ANN do tipo *perceptron* com uma transformada discreta linear, como a DFT e a DCT.

Sem perda de generalidade, uma ANN do tipo *perceptron* pode ser vista como uma função f que multiplica uma matriz de pesos $\mathbf{\Omega}$ de tamanho $N \times N$[1] aprendidos por uma entrada $\mathbf{x}$ de tamanho N e soma ao resultado um termo de tendência ou viés (*bias*) b. O resultado dessas operações é submetido a uma função de ativação $a(\cdot)$ e produz uma saída $\bar{\mathbf{y}}$ também de tamanho N:

$$\bar{\mathbf{y}} = f(\mathbf{\Omega}, b, \mathbf{x}) = a(\mathbf{\Omega} \cdot \mathbf{x} + b). \tag{4.1}$$

Note que se a função de ativação é a identidade, $a(x) = x$, e o termo de tendência b é nulo, $b = 0$, então f é uma transformação linear $T : \mathbb{R}^N \to \mathbb{R}^N$ que mapeia $\mathbf{x}$ para $\bar{\mathbf{y}}$ conforme $T : \bar{\mathbf{y}} \mapsto \mathbf{\Omega} \cdot \mathbf{x}$. Dadas as suposições acima, pode-se escrever simplesmente

$$\bar{\mathbf{y}} = f(\mathbf{\Omega}, \mathbf{x}) = \mathbf{\Omega} \cdot \mathbf{x}. \tag{4.2}$$

Os parâmetros $\mathbf{\Omega}$ da Equação (4.2) podem ser otimizados pelo método do gradi-

[1] Em geral, o número de entradas N_1 e o número de saídas N_2 de uma camada pode ser diferente de N. Nesse caso, $\mathbf{\Omega}$ tem tamanho $N_2 \times N_1$.

ente (CHOLLET, 2018, pág. 48) dada uma função objetivo $L(\cdot)$ que relaciona $\mathbf{\Omega}$ a pares de dados (X, Y), em que $X = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{M-1})$ e $Y = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{M-1})$ e M é o tamanho da base de dados de treinamento. Aqui, segue-se o conceito de aprendizado de máquina supervisionado (GOODFELLOW; BENGIO; COURVILLE, 2016).

Geralmente, em problemas de regressão supervisionada, o objetivo é determinar $\mathbf{\Omega}$ de tal forma que um erro envolvendo $\mathbf{y}_k$ e $\bar{\mathbf{y}}_k$, para $k = 0, 1, \ldots, M-1$, seja minimizado. Aqui, $\mathbf{y}_k$ é obtido de $\mathbf{x}_k$ por um processo desconhecido pela ANN e $\bar{\mathbf{y}}_k = \mathbf{\Omega} \cdot \mathbf{x}_k$.

Uma escolha natural para a função de perda (*loss function*) em problemas de regressão é o erro quadrático médio (GOODFELLOW; BENGIO; COURVILLE, 2016). Para uma ANN do tipo *perceptron* com pesos $\mathbf{\Omega}$, pode-se computar o erro quadrático médio por

$$L(\mathbf{\Omega}, X, Y) = \frac{1}{M} \sum_{k=0}^{M-1} \|\mathbf{y}_k - (\mathbf{\Omega} \cdot \mathbf{x}_k)\|_2^2. \tag{4.3}$$

O erro quadrático médio é diferenciável e quantifica a diferença entre $\mathbf{y}_k$ e $\bar{\mathbf{y}}_k$. Portanto, o erro quadrático médio é adotado como função de perda nesse trabalho.

Note que, se os hiper-parâmetros de treinamento são especificados de forma adequada e o conjunto de treinamento (X, Y) é construído de tal forma que

$$\mathbf{y}_k = \mathbf{C} \cdot \mathbf{x}_k, \tag{4.4}$$

para $k = 0, 1, \ldots, M-1$, então $\mathbf{\Omega} \to \mathbf{C}$. Salienta-se que a Equação (4.4) é a transformação direta de um sinal unidimensional $\mathbf{x}_k$ realizada pela DCT.

4.2 Derivando Aproximações para a DCT com Baixa Complexidade Aritmética com uma MLP

Ao usar o formalismo da Seção 4.1, desprezando erros de representação em ponto flutuante, obtém-se $\mathbf{\Omega} = \mathbf{C}$. A presente seção extrapola essa abordagem para obter pesos $\mathbf{\Omega}$ da ANN com elementos em $\mathscr{T}$ que melhor aproxima $\mathbf{C}$ sob a métrica de erro adotada.

A SCA de Deng and Zhang (2022) permite treinar uma ANN ternária com esparsidade ajustável pela reparametrização de seus pesos e uso de um regularizador de pesos discretos (WDR). A seguir, descreve-se brevemente a SCA, adequando-a às especificidades deste trabalho.

A SCA parametriza uma camada da ANN com

$$\mathbf{\Omega} = \tanh(\mathbf{\Theta}), \tag{4.5}$$

em que $\tanh(\cdot)$ aplica a tangente hiperbólica a cada elemento de sua matriz argumento e $\mathbf{\Theta}$ é a matriz de pesos da ANN. Deng and Zhang (2022) se valem de que $\tanh(\cdot)$ tem domínio em $(-1, 1)$ e mostram que a parametrização em (4.5) auxilia o treinamento de uma ANN ternária. Visto que $\tanh(\mathbf{\Theta})$ tem valores reais, seu uso permite a retropropagação do gradiente e a minimização da função de perda (DENG; ZHANG, 2022).

Neste trabalho, podemos substituir (4.5) em (4.3), obtendo

$$L(\tanh(\mathbf{\Theta}), X, Y) = \frac{1}{M} \sum_{k=0}^{M-1} \|\mathbf{y}_k - [\tanh(\mathbf{\Theta}) \cdot \mathbf{x}_k]\|_2^2. \tag{4.6}$$

A SCA ainda incorpora um regularizador, o WDR, para incentivar que $\tanh(\mathbf{\Theta})$ tenha elementos ternários. O WDR é dado por

$$D(\tanh(\mathbf{\Theta})) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [\alpha - \tanh^2(\theta_{i,j})] \tanh^2(\theta_{i,j}), \tag{4.7}$$

em que $\theta_{i,j}$ é o (i, j)-ésimo elemento da matriz de pesos $\mathbf{\Theta}$ de tamanho $N \times N$. O parâmetro α, provido pelo usuário, controla a esparsidade de $\tanh(\mathbf{\Theta})$. Deng and Zhang (2022) mostram que a função $D(\tanh(\mathbf{\Theta}))$ tem três pontos de mínimo – -1, 0 e $+1$[2] – quando $\alpha \in [0, 2]$.

Considerando os termos de perda em (4.6) e de regularização em (4.7), pode-se compor uma função objetivo conforme a SCA para uma ANN do tipo *perceptron* (idealmente) ternária como

$$J(\tanh(\mathbf{\Theta}), X, Y) = L(\tanh(\mathbf{\Theta}), X, Y) + \lambda_1 D(\tanh(\mathbf{\Theta})), \tag{4.8}$$

em que λ_1 controla a importância do regularizador de pesos discretos. Se λ_1 é muito pequeno, eventualmente, $\tanh(\mathbf{\Theta})$ não é ternário. Na prática, com a seleção adequada de hiper-parâmetros, $\text{round}[\tanh(\mathbf{\Theta})] \approx \tanh(\mathbf{\Theta})$ de forma que aplicar a função $\text{round}(\cdot)$ não impacta na computação efetuada pela ANN (DENG; ZHANG, 2022).

Ao otimizar a Equação (4.8) com $\alpha = 0$ e $\lambda_1 = 1$ e uma base de dados de treinamento (X, Y) como aquela discutida na Seção 4.1, obtemos, desprezados erros de

[2] O limite da função $\tanh(x)$, quando $x \to -\infty$, é -1. O limite dessa função é $+1$ quando $x \to +\infty$.

representação, $\mathbf{\Omega} = \tanh(\mathbf{\Theta}) = \mathbf{T}_{\pm}$. O incremento de α faz com que o valor da função de perda em (4.6) seja maior (eventualmente $\tanh(\mathbf{\Theta})$ degenera em uma matriz de zeros) e alterar λ_1, nesse caso, faz com que a convergência para pesos discretos seja mais ou menos rápida. Dessa forma, treinar uma ANN do tipo *perceptron* com uma única camada se mostra uma abordagem limitada.

Conforme descrito na Seção 3.1, diferentes trabalhos propõem aproximações para a DCT na forma $\hat{\mathbf{C}} = \mathbf{S} \cdot \mathbf{T}$ (veja a Equação (3.6)). Este trabalho, portanto, propõe a definição e treinamento de uma ANN do tipo *perceptron* com duas camadas (MLP), sem termos de tendência e sem função de ativação, ao invés de uma[3].

A MLP proposta, portanto, otimiza $\boldsymbol{\mathcal{T}} = \tanh(\mathbf{\Theta})$, que tem $N \times N$ elementos (idealmente) em $\mathscr{T}$, e uma matriz $\mathbf{\Sigma} = \mathrm{diag}(\boldsymbol{\varsigma})$, que tem N elementos diferentes de zero, $\boldsymbol{\varsigma} = [\varsigma_0, \varsigma_1, \ldots, \varsigma_{N-1}]^\top$, em ponto flutuante. Para fins de notação, neste texto, quando $\mathrm{diag}(\cdot)$ recebe um vetor unidimensional, essa função retorna uma matriz diagonal cujos elementos não-nulos são seu argumento. Se $\mathrm{diag}(\cdot)$ recebe uma matriz, tal função retorna um vetor com os elementos da diagonal principal desse argumento.

Com isso, finalmente, reescrevemos a função de perda de (4.6) para a MLP conforme

$$L(\{\tanh(\mathbf{\Theta}), \boldsymbol{\varsigma}\}X, Y) = \frac{1}{M} \sum_{k=0}^{M-1} \|\mathbf{y}_k - \{[\mathrm{diag}(\boldsymbol{\varsigma}) \cdot \tanh(\mathbf{\Theta})] \cdot \mathbf{x}_k\}\|_2^2. \tag{4.9}$$

Note que a MLP otimiza N parâmetros em ponto flutuante referentes a $\boldsymbol{\varsigma}$ (que se relaciona à matriz $\mathbf{S}$) e $N \times N$ parâmetros (idealmente) ternários referentes a $\tanh(\mathbf{\Theta})$ (que se relaciona à matriz $\mathbf{T}$). Ao fim,

$$\mathrm{diag}(\boldsymbol{\varsigma}) \cdot \tanh(\mathbf{\Theta}) \approx \mathrm{diag}(\boldsymbol{\varsigma}) \cdot \mathrm{round}[\tanh(\mathbf{\Theta})] \approx \mathbf{C}. \tag{4.10}$$

É importante lembrar que $\mathbf{S}$ pode ser derivada de $\mathbf{T}$ através da Equação (3.7). Entretanto, durante o treinamento da MLP, eventualmente $\boldsymbol{\mathcal{T}} \cdot \boldsymbol{\mathcal{T}}^\top$ é singular, de tal forma que $\sqrt{\left[\boldsymbol{\mathcal{T}} \cdot \boldsymbol{\mathcal{T}}^\top\right]^{-1}}$ não pode ser computado. Por isso, $\mathrm{diag}(\boldsymbol{\varsigma})$ é otimizado pela MLP.

Note que a ANN do tipo *perceptron* de uma camada descrita anteriormente faz com que $\tanh(\mathbf{\Theta}) = \mathbf{T}_{\pm}$. A matriz de transformação de baixa complexidade da SDCT, embora uma boa aproximação para a DCT, não é ortogonal. A ortogonalidade da transformada aproximada é uma propriedade geralmente almejada em trabalhos relacionados,

[3]Na prática, pode-se modelar uma transformada discreta linear como uma MLP com um número arbitrário de camadas.

visto que implementações eficientes da transformação direta e inversa podem ser compartilhadas (COELHO et al., 2018; SILVEIRA et al., 2021).

A fim de incentivar a proposição de matrizes de transformação ortogonais, adiciona-se outro termo de regularização à função objetivo. Esse novo termo, baseado no desvio de diagonalidade (FLURY; GAUTSCHI, 1986; COELHO et al., 2018), é dado por:

$$O(\tanh(\boldsymbol{\Theta})) = 1 - \frac{\| \operatorname{diag}(\tanh(\boldsymbol{\Theta}) \cdot \tanh(\boldsymbol{\Theta})^\top)\|_F^2}{\|(\tanh(\boldsymbol{\Theta}) \cdot \tanh(\boldsymbol{\Theta})^\top)\|_F^2} \tag{4.11}$$

em que $\|\cdot\|_F$ é a norma de Frobenius (WATKINS, 2004, pág. 115). A função $O(\tanh(\boldsymbol{\Theta}))$ retorna um valor em $[0, 1]$; e, em especial, o valor zero caso a matriz argumento seja ortogonal.

Por fim, a função objetivo adotada neste trabalho para otimização da MLP é dada por

$$J(\{\tanh(\boldsymbol{\Theta}), \boldsymbol{\varsigma}\}X, Y) = L(\{\tanh(\boldsymbol{\Theta}), \boldsymbol{\varsigma}\}, X, Y) + \lambda_1 D(\tanh(\boldsymbol{\Theta})) + \lambda_2 O(\tanh(\boldsymbol{\Theta})), \tag{4.12}$$

em que λ_2 pondera a importância do regularizador de ortogonalidade.

A definição dos hiper-parâmetros α, λ_1 eλ_2 e a discussão dos resultados é feita no Capítulo 5.

5 RESULTADOS

Este capítulo apresenta os experimentos realizados e resultados obtidos neste trabalho. A Seção 5.1 especifica os recursos e ferramentas computacionais utilizados. A Seção 5.2 discute a escolha dos parâmetros de treinamento e dos hiper-parâmetros da MLP. Aproximações para a DCT derivadas pela abordagem proposta são apresentadas e avaliadas na Seção 5.3. Essas transformadas são submetidas a um experimento de compressão de imagens, conforme apresentado e discutido na Seção 5.4.

5.1 Recursos e Ferramentas Computacionais

Neste trabalho, o ambiente virtual de computação Google Colab é utilizado. As configurações da máquina fornecida nesse ambiente são: Intel Xeon CPU @2.20 GHz, 13 GB RAM, GPU NVIDIA Tesla K80 com 12 GB GDDR5 VRAM.

Os experimentos são implementados usando a linguagem Python (versão 3.9.16). A biblioteca Numpy (versão 1.22.4) é considerada para realização de cálculos matriciais, e a ferramenta Matplotlib (versão 3.7.1) para plotagem gráfica. A biblioteca Skimage (versão 0.19.3) é adotada para leitura e escrita de imagens e computação de métricas de qualidade de imagem. Para modelagem e treinamento das ANNs, usa-se o *framework* TensorFlow (versão 2.11.0).

5.2 Escolha de Parâmetros e Hiper-parâmetros

A MLP descrita na Seção 4.2 é treinada através do otimizador SGD (*Stochastic Gradient Descent*) (DEISENROTH; FAISAL; ONG, 2020, pág. 231) com uma taxa de aprendizado variável começando em 10^{-2} e decaindo exponencialmente conforme o processo de aprendizado avança (ZHANG et al., 2020, pág. 474).

O conjunto de dados para treinamento (X, Y) é construído de tal forma que $\mathbf{x}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_8)$, em que $\mathbf{0}$ é o vetor nulo de tamanho $N = 8$ e $\mathbf{I}_8$ é a matriz identidade de ordem $N = 8$, e $\mathbf{y}_k = \mathbf{C} \cdot \mathbf{x}_k, k = 0, 1, \ldots, M - 1$. Os vetores $\mathbf{x}_k$ e $\mathbf{y}_k$ têm tamanho $N = 8$. Adota-se $M = 3072$. Considera-se um tamanho de amostra (*batch size*) $m = 32$ para treinamento das épocas. Utiliza-se $E = 150$ épocas para o treinamento das MLPs. Outros $W = 1024$ pares de dados, construídos da mesma forma que os dados de treinamento,

são utilizados para validação. Não há sobreajuste (*overfitting*) (ZHANG et al., 2020, pág. 144).

Os $N \times N + N$ pesos da MLP são inicializados com valores em $\mathcal{N}(0, 0.05)$. Para a otimização da MLP, é necessário atribuir valores para os dois hiper-parâmetros da função objetivo em (4.12), λ_1 e λ_2. Além destes, o WDR da (4.7) também tem um hiper-parâmetro α. Deng and Zhang (2022) mostram que α e λ_1, hiper-parâmetros da SCA, têm uma relação complexa e dependente de arquitetura de ANN.

Avaliar todas as possíveis combinações de α, λ_1 e λ_2 é impraticável para este trabalho. Aqui, opta-se por pré-estabelecer um valor para λ_2 e variar α no intervalo $[0, 2]$, adequando λ_1 de forma a maximizar a proximidade dos elementos de $\tanh(\boldsymbol{\Theta})$ com elementos de $\mathscr{T}$. Em outras palavras, faz-se uma busca em grade (*grid search*) (GOODFELLOW; BENGIO; COURVILLE, 2016, pág. 432) com

- $\alpha \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0{,}025, 0{,}05, 0{,}075, 0{,}1, 0{,}125, 0{,}15, 0{,}175, 0{,}2, 0, 225, 0{,}25, 0{,}275, 0{,}3, 0{,}5, 1, 1{,}5, 2\}$ e
- $\lambda_1 \in \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0{,}1, 0{,}2, 0{,}5, 1\}$.

O parâmetro $\lambda_2 = 0{,}125$ é atribuído empiricamente com base em simulações piloto. Durante a busca, são consideradas apenas matrizes de transformação ortogonais.

5.3 Aproximações para a DCT Derivadas

Conforme discutido na Seção 4.2, a MLP proposta otimiza uma matriz diagonal $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\varsigma})$ e uma matriz (idealmente) ternária $\boldsymbol{\mathcal{T}} = \tanh(\boldsymbol{\Theta})$. As matrizes $\boldsymbol{\Sigma}$ e $\boldsymbol{\mathcal{T}}$ são relacionadas, respectivamente, às matrizes $\mathbf{S}$ e $\mathbf{T}$ da Equação (3.6). Como $\boldsymbol{\varsigma}$ pode assumir valores reais, consideramos

$$\boldsymbol{\Sigma}' = \mathrm{sign}(\boldsymbol{\Sigma}) \cdot \boldsymbol{\Sigma} \tag{5.1}$$

e

$$\boldsymbol{\mathcal{T}}' = \mathrm{sign}(\boldsymbol{\Sigma}) \cdot \boldsymbol{\mathcal{T}}. \tag{5.2}$$

Note que $\boldsymbol{\Sigma} \cdot \boldsymbol{\mathcal{T}} \equiv \boldsymbol{\Sigma}' \cdot \boldsymbol{\mathcal{T}}'$.

Este trabalho considera aproximações ortogonais para a DCT ($\hat{\mathbf{C}}$) dadas por

$$\widehat{\mathbf{C}} = \widehat{\boldsymbol{\Sigma}}' \cdot \widehat{\boldsymbol{\mathcal{T}}}', \tag{5.3}$$

em que

$$\widehat{\boldsymbol{\Sigma}}' = \sqrt{\left[\widehat{\boldsymbol{\mathcal{T}}}' \cdot \widehat{\boldsymbol{\mathcal{T}}}'^{\top}\right]^{-1}}, \tag{5.4}$$

e

$$\widehat{\boldsymbol{\mathcal{T}}}' = \mathrm{round}(\boldsymbol{\mathcal{T}}'). \tag{5.5}$$

Considere os valores de α, λ_1 e λ_2 discutidos na Seção 5.2. Obtém-se, para cada valor de α, a transformada $\widehat{\boldsymbol{\mathcal{T}}}'$ que mais se ajusta ao domínio $\mathscr{T}$, desde que ortogonal, dada a gama de valores λ_1 e o valor pré-fixado λ_2.

Precisamente, este processo resulta em duas transformadas de baixa complexidade:

$$\widehat{\boldsymbol{\mathcal{T}}}'_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 1 & -1 & 1 & -1 & 1 & 0 \end{bmatrix}, \tag{5.6}$$

$$\widehat{\boldsymbol{\mathcal{T}}}'_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 & 1 & -1 & 1 & 0 \end{bmatrix}. \tag{5.7}$$

De acordo com (5.4), as matrizes de baixa complexidade $\widehat{\boldsymbol{\mathcal{T}}}'_1$ e $\widehat{\boldsymbol{\mathcal{T}}}'_2$ podem ser escaladas, respectivamente, por

$$\widehat{\boldsymbol{\Sigma}}'_1 = \mathrm{diag}\left(\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}\right), \tag{5.8}$$

e

$$\widehat{\mathbf{\Sigma}}_2' = \mathrm{diag}\left(\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{4}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{4}}, \frac{1}{\sqrt{6}}\right). \tag{5.9}$$

As transformadas escaladas por (5.3), $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$, são aproximações ortonormais de baixa complexidade para a DCT.

A transformada $\widehat{\mathbf{C}}_1$ foi inicialmente apresentada em Cintra, Bayer and Tablada (2014). A aproximação $\widehat{\mathbf{C}}_2$ coincide com a DCT arredondada (RDCT), introduzida em Cintra and Bayer (2011), Bayer and Cintra (2010), e compõe o estado da arte em aproximações de baixa complexidade com elementos em $\mathscr{T}$.

5.3.1 Eficiência de Transformada e Similaridade com a DCT

Aproximações visam reduzir a complexidade aritmética de uma transformada, como a DCT, e, ao mesmo tempo, preservar suas características fundamentais (BAYER et al., 2013; BOUGUEZEL; AHMAD; SWAMY, 2010). Deve-se, portanto, avaliar o desempenho de uma transformada aproximada perante sua contraparte exata (computada por sua definição).

O erro total de energia (CINTRA; BAYER, 2011) e o erro quadrático médio[1] (BRITANAK P. YIP, 2007) são métricas amplamente utilizadas que avaliam a similaridade entre uma aproximação $\hat{\mathbf{C}}$ e a DCT $\mathbf{C}$ (COELHO et al., 2018; TABLADA; BAYER; CINTRA, 2015).

O erro total de energia (CINTRA; BAYER, 2011) é dado simplesmente por

$$\epsilon(\widehat{\mathbf{C}}, \mathbf{C}) = \pi \cdot \| \mathbf{C} - \widehat{\mathbf{C}} \|_F^2. \tag{5.10}$$

Por outro lado, o erro quadrático médio pode ser calculado por (BRITANAK P. YIP, 2007)

$$\mathrm{MSE}(\widehat{\mathbf{C}}, \mathbf{C}) = \frac{1}{N} \cdot \mathrm{tr}\left[(\mathbf{C} - \widehat{\mathbf{C}}) \cdot \mathbf{R}_\mathbf{x} \cdot (\mathbf{C} - \widehat{\mathbf{C}})^\top\right], \tag{5.11}$$

em que $\mathrm{tr}(\cdot)$ é a função traço e $\mathbf{R}_\mathbf{x}$ é a matriz de covariância com $\rho = 0.95$ cujos elementos $r_{i,j}$ são dados por

$$r_{i,j} = \rho^{|i-j|}, \quad i, j = 0, \ldots, N-1. \tag{5.12}$$

Pode-se computar o ganho de codificação (JAYANT, 1974) e a eficiência (TA-

[1] Não confundir essa métrica com a função de perda discutida na Seção 4.1

Tabela 5.1 – Hiper-parâmetros que geram aproximações ortogonais para a DCT, suas eficiência e similaridade com a DCT e adequação à modelagem da MLP. Assume-se $\lambda_2 = 0{,}125$.

$\widehat{\mathbf{C}}_k$	α	λ_1	$\epsilon(\cdot,\cdot)$	$\mathrm{MSE}(\cdot,\cdot)$	$\mathrm{C_g}(\cdot)$	$\eta(\cdot)$	$\Delta(\cdot)$	$\delta(\cdot,\cdot)$
$k=1$	0,0750	1,0000	1,7945	0,0092	8.1834	87,1566	0,0274	0,0019
$k=2$	0,1250	1,0000	1,7945	0,0092	8.1826	87,4297	0,0630	0.0018

KALA; NIKARA, 2001) de uma transformada a fim de quantificar seu poder de descorrelação de dados.

O ganho de codificação de uma transformada $\widehat{\mathbf{C}}$ é dado por

$$\mathrm{C_g}(\widehat{\mathbf{C}}) = 10 \cdot \log_{10}\left[\prod_{k=1}^{N} \frac{1}{(\mathbf{A}_k \cdot \mathbf{B}_k)^{1/N}}\right], \tag{5.13}$$

em que $\mathbf{A}_k = \mathrm{su}[(\mathbf{h}_k^\top \cdot \mathbf{h}_k) \odot \mathbf{R}_x]$, $\mathrm{su}(\cdot)$ é a uma função que retorna a soma dos elementos de uma matriz e $\mathbf{B}_k = \|\mathbf{g}_k\|_2^2$. Aqui, $\mathbf{h}_k$ e $\mathbf{g}_k$ são a k-ésima linha de $\widehat{\mathbf{C}}$ e $\widehat{\mathbf{C}}^\top$, respectivamente.

Por fim, a eficiência da transformada (CHAM, 1989) $\widehat{\mathbf{C}}$ é dada por

$$\eta(\widehat{\mathbf{C}}) = \frac{\sum_{i=1}^{N} |\bar{r}_{ii}|}{\sum_{i=1}^{N}\sum_{j=1}^{N} |\bar{r}_{ij}|} 100 \tag{5.14}$$

em que $\bar{r}_{i,j}$ é o $\{i,j\}$-ésimo elemento de $\widehat{\mathbf{C}} \cdot \mathbf{R}_\mathbf{x} \cdot \widehat{\mathbf{C}}^\top$.

Além das métricas descritas acima, computa-se a proximidade de $\boldsymbol{\mathcal{T}}$ com $\mathrm{round}(\boldsymbol{\mathcal{T}})$ por

$$\Delta(\boldsymbol{\mathcal{T}}) = \|\boldsymbol{\mathcal{T}} - \mathrm{round}(\boldsymbol{\mathcal{T}})\|_F^2. \tag{5.15}$$

Também, considera-se a variação entre $\mathrm{diag}(\boldsymbol{\Sigma}')$ e $\mathrm{diag}(\widehat{\boldsymbol{\Sigma}}')$, dada por

$$\delta(\boldsymbol{\Sigma}', \widehat{\boldsymbol{\Sigma}}') = \|\,\mathrm{diag}(\boldsymbol{\Sigma}') - \mathrm{diag}(\widehat{\boldsymbol{\Sigma}}')\|_2^2. \tag{5.16}$$

As métricas em (5.15) e (5.16) representam a inconformidade dos parâmetros aprendidos pela MLP com a modelagem dada pelas Equações (5.5) e (5.4), respectivamente.

Os resultados de todas essas métricas, bem como os valores de α, λ_1 e λ_2, associados às aproximações para a DCT derivadas são reportadas na Tabela 5.1. Ao variar α e λ_1, obtemos $\widehat{\boldsymbol{\mathcal{T}}}_1'$ e $\widehat{\boldsymbol{\mathcal{T}}}_2'$. Oito pares de α e λ_1 levam à otimização de $\widehat{\boldsymbol{\mathcal{T}}}_1'$ e outros dois pares desses hiper-parâmetros geram $\widehat{\boldsymbol{\mathcal{T}}}_2'$. Esses pares de valores para α e λ_1 são listados na Tabela 5.2. Nenhuma transformada ortogonal é obtida para os demais valores de α. Note que a relação entre α e λ_1 não é trivial, isto é, λ_1 não parece ser relacionado a α através de uma função simples.

Tabela 5.2 – Valores de α e λ_1 que levam à otimização da transformada $\widehat{\mathbf{C}}_k$.

$\widehat{\mathbf{C}}_k$	α	λ_1	$\Delta(\cdot)$	$\delta(\cdot,\cdot)$
$k=1$	0,0000	0,1000	0,7208	0,0007
$k=1$	10^{-5}	0,1000	0,5876	0,0005
$k=1$	0,0001	0,1000	0,5930	0,0007
$k=1$	0,0010	0,1000	0,5895	0,0008
$k=1$	0,0100	0,1000	0,5132	0,0006
$k=1$	0,0500	1,0000	0,0545	0,0019
$k=1$	0,0750	1,0000	0,0218	0,0019
$k=1$	0,1000	0,5000	0,0402	0,0018
$k=2$	0,1250	1,0000	0,0630	0,0018
$k=2$	0,2000	0,2000	0,5325	0,0007

5.3.2 Complexidade Aritmética

Uma maneira de medir o custo de computação de uma transformada é através da complexidade aritmética (COELHO et al., 2019; BOUGUEZEL; AHMAD; SWAMY, 2008; BAYER; CINTRA, 2012; CINTRA; BAYER; TABLADA, 2014). Esse tipo de avaliação é independente de implementação, seja em *hardware* ou *software*, e provê uma métrica justa de comparação (BLAHUT, 2010). Multiplicações são as operações mais custosas de serem realizadas, seguidas de adições e deslocamentos de *bits* (OLIVEIRA et al., 2017). Dessa forma, ao propor uma aproximação para uma transformada, procura-se reduzir sua complexidade multiplicativa, aditiva e quantidade de deslocamentos de *bits* (WANG, 1984).

Como dito na Seção 2.2, se computada por sua definição matricial, a DCT de tamanho $N = 8$ requer 64 multiplicações e 56 adições, ambas em aritmética de ponto-flutuante, para ser aplicada a um vetor unidimensional. As transformações dadas por $\widehat{\boldsymbol{\mathcal{T}}}'_1$ e $\widehat{\boldsymbol{\mathcal{T}}}'_2$, conforme apresentadas em (5.6) e (5.7), requerem apenas 48 e 40 adições/subtrações inteiras, respectivamente. Em aplicações de compressão de imagens, as complexidades aritméticas de $\widehat{\boldsymbol{\mathcal{T}}}'_1$ e $\widehat{\boldsymbol{\mathcal{T}}}'_2$ coincidem com as de $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$.

Algoritmos rápidos para a DCT reduzem a complexidade de sua aplicação para 11 multiplicações e 29 adições reais (LOEFFLER; LIGTENBERG; MOSCHYTZ, 1989). As transformadas $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ podem ser computadas, via algoritmos rápidos, utilizando apenas 24 e 22 adições inteiras, respectivamente. Detalhes podem ser encontrados em (CINTRA; BAYER; TABLADA, 2014; CINTRA; BAYER, 2011; BAYER; CINTRA, 2010).

5.4 Experimento em Compressão de Imagens

As transformadas $\widehat{\mathrm{C}}_1$ e $\widehat{\mathrm{C}}_2$ são submetidas a um experimento de compressão de imagens do tipo JPEG. Para comparação com as transformadas $\widehat{\mathrm{C}}_1$ e $\widehat{\mathrm{C}}_2$, consideram-se $\widetilde{\mathrm{C}}_1$ (BAYER; CINTRA, 2012) e $\widetilde{\mathrm{C}}_2$ (SILVEIRA et al., 2021). Ambas são aproximações ternárias para a DCT. A primeira é tida como uma aproximação para a DCT com baixíssima complexidade aritmética, demandando apenas 14 adições para sua implementação. A segunda se mostra como uma forte concorrente para $\widehat{\mathrm{C}}_1$ e $\widehat{\mathrm{C}}_2$, requerendo 22 adições.

Neste experimento, aplica-se a metodologia descrita na Seção 2.1, substituindo C pelas aproximações $\widehat{\mathbf{C}}_k$ e $\widetilde{\mathbf{C}}_k, k = 1, 2$, nas Equações (2.1) e (2.9). Neste experimento, $QF \in \{5, 15, 25, 35, 45, 55, 65, 75, 85, 95\}$. Para o experimento, consideram-se 45 imagens de 8 *bits*, em escala de cinza, com tamanho $L \times A$, $L = A = 512$, disponíveis publicamente em (USC-SIPI, 2011). Esse conjunto de imagens é utilizado em diversos trabalhos da área (CANTERLE et al., 2020; COELHO et al., 2018; BAYER et al., 2013; SILVEIRA et al., 2021; TABLADA; BAYER; CINTRA, 2015).

São computados a relação sinal-ruído de pico (PSNR) e o índice de similaridade estrutural (SSIM) (WANG et al., 2004). Estas métricas permitem avaliar o comportamento das transformadas, de forma indireta, através da qualidade de reconstrução das imagens. Também é computada quantidade de *bits* por *pixel* (bpp).

O PSNR de uma imagem $\tilde{\mathbf{I}}$ com relação a uma imagem $\mathbf{I}$, ambas de 8 *bits*, é dado por

$$\mathrm{PSNR}(\mathbf{I}, \tilde{\mathbf{I}}) = \ 10 \cdot \log_{10} \left(\frac{255^2}{\mathrm{MSE}(\mathbf{I}, \tilde{\mathbf{I}})} \right), \tag{5.17}$$

em que

$$\mathrm{MSE}(\mathbf{I}, \tilde{\mathbf{I}}) = \frac{\|\mathbf{I} - \tilde{\mathbf{I}}\|_F^2}{L \times A} \tag{5.18}$$

e $L \times A$ é a dimensão das imagens $\mathbf{I}$ e $\tilde{\mathbf{I}}$.

O SSIM quantifica a percepção da visão humana de forma mais adequada que o PSNR, levando em consideração a interdependência espacial dos *pixels*, luminância e contraste (WANG et al., 2004). O SSIM de uma imagem $\tilde{\mathbf{I}}$ com relação a $\mathbf{I}$ é dado por

$$\overline{\mathrm{SSIM}}(\mathbf{I}, \tilde{\mathbf{I}}) = \frac{1}{\mathrm{Q}} \sum_{\mathrm{j}=1}^{\mathrm{Q}} \mathrm{SSIM}(\mathbf{I}_j, \tilde{\mathbf{I}}_j), \tag{5.19}$$

em que $\mathbf{I}_j$ e $\tilde{\mathbf{I}}_j$ são os j-ésimos blocos de $\mathbf{I}$ e $\tilde{\mathbf{I}}$, respectivamente, e

$$\mathrm{SSIM}(\mathbf{X},\mathbf{Y}) = \frac{(2\mu(\mathbf{X})\mu(\mathbf{Y}) + \mathrm{C}_1)(2\,\mathrm{cov}(\mathbf{X},\mathbf{Y}) + \mathrm{C}_2)}{(\mu(\mathbf{X})^2 + \mu(\mathbf{Y})^2 + \mathrm{C}_1)(\sigma(\mathbf{X})^2 + \sigma(\mathbf{Y})^2 + \mathrm{C}_2)}\,. \tag{5.20}$$

Considera-se $C1 = 6{,}5025$ e $C2 = 58{,}5225$ (WANG et al., 2004),

$$\mu(\mathbf{Z}) = \frac{1}{\mathrm{D}}\sum_{\mathrm{i}=1}^{\mathrm{D}} z_i, \tag{5.21}$$

$$\sigma(\mathbf{Z}) = \frac{1}{D-1}\sum_{i=1}^{D}\left[z_i - \mu(\mathbf{Z})\right]^2 \tag{5.22}$$

e

$$\mathrm{cov}(\mathbf{Z},\mathbf{Y}) = \frac{1}{D-1}\sum_{i=1}^{D}\left[z_i - \mu(\mathbf{Z})\right)(y_i - \mu(\mathbf{Y})]\,, \tag{5.23}$$

em que z_i e y_i são os i-ésimos elementos de $\mathbf{Z}$ e $\mathbf{Y}$, respectivamente, e D é o tamanho de $\mathbf{Z}$ e $\mathbf{Y}$.

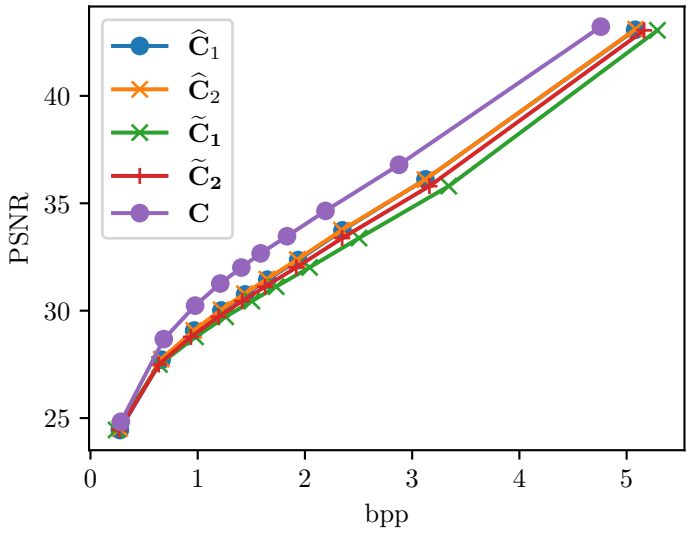
O bpp é uma medida absoluta que representa a quantidade média de pixel necessarios para representar uma imagem. Para compressão de imagens, considera-se o bpp de uma imagem $\tilde{\mathbf{I}}$ como

$$\mathrm{bpp}(\tilde{\mathbf{I}}) = 8 \times \frac{\mathrm{sz}(\tilde{\mathbf{I}})}{A \times L}, \tag{5.24}$$

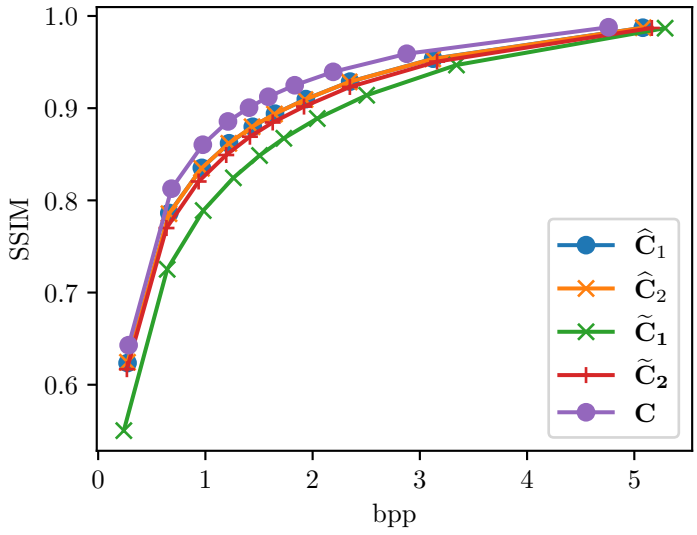
em que $\mathrm{sz}(\cdot)$ retorna o tamanho (em *bytes*) da imagem argumento. Assume-se, como tamanho, a quantidade de valores não-zero após a quantização necessários para armazenamento da imagem comprimida $\tilde{\mathbf{I}}$.

A Figura 5.1 apresenta os resultados médios de PSNR e SSIM para diferentes bpps. Os diferentes valores de bpp são resultantes de diferentes valores de QF, indicados com marcações nas curvas. Para melhor visualização, a Figura 5.2 mostra o erro absoluto percentual (EAP) – em termos de PSNR e SSIM – dessas transformadas com relação à DCT. Percebe-se que as transformadas $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ têm desempenho muito similar e melhor que das transformadas $\widetilde{\mathbf{C}}_1$ e $\widetilde{\mathbf{C}}_2$. Especialmente, $\widehat{\mathbf{C}}_1$ se sobressai a $\widehat{\mathbf{C}}_2$ para valores de bpp maiores que 2. Nota-se também que $\widehat{\mathbf{C}}_1$ gera imagens com menor bpp a uma qualidade similar as demais quando $QF = 95$ (última marcação das curvas).

As Figuras 5.3, 5.4 e 5.5 mostram exemplos de compressão para diferentes imagens e valores de QF. A compressão feita pelas aproximações $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ são virtualmente idênticas àquela realizada pela DCT. A Figura 5.3 mostra resultados para $QF = 10$. As transformadas $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ apresentam melhor qualidade mas atingem mais altos valores de

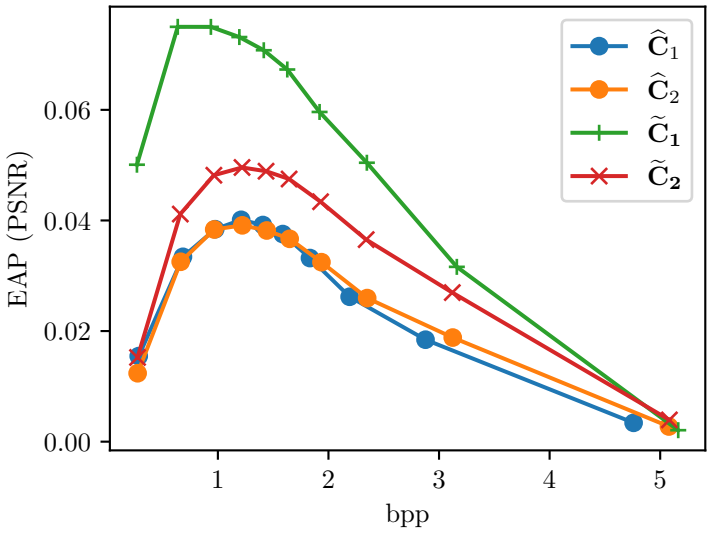


(a)

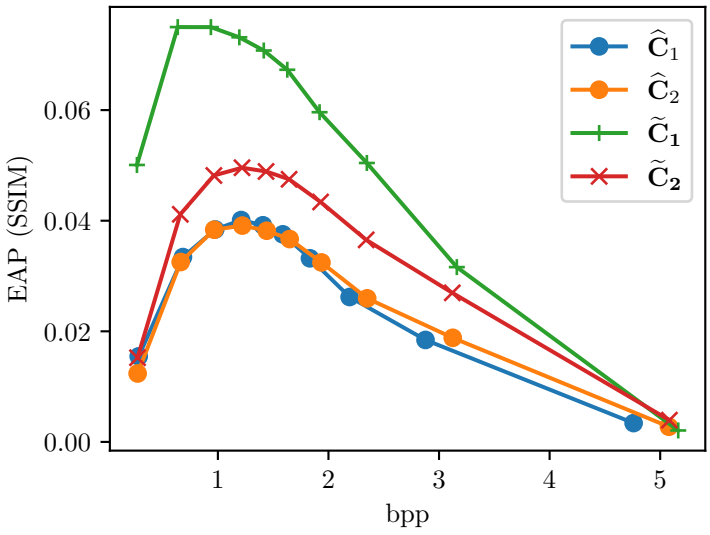


(b)

Figura 5.1 – Valores médios de (a) PSNR (em dB) e (b) SSIM *versus* bpp.



(a)

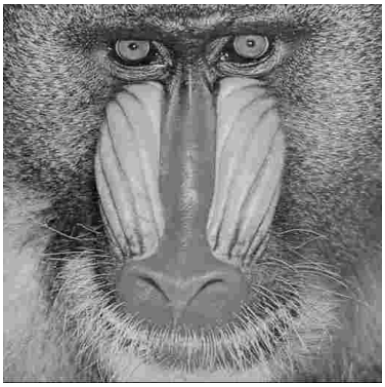


(b)

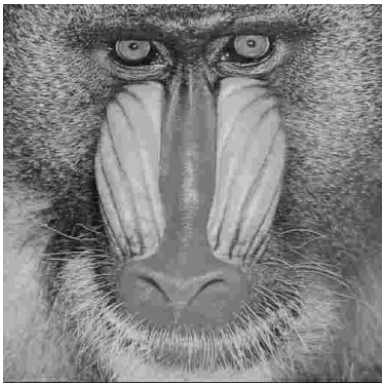
Figura 5.2 – Erro absoluto percentual médio (a) PSNR e (b) SSIM *versus* bpp.

bpp se comparadas a $\widetilde{\mathbf{C}}_{\mathbf{1}}$ e $\widetilde{\mathbf{C}}_{\mathbf{2}}$. A Figura 5.4 ilustra o caso em que $QF = 50$. Vê-se que as transformadas $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ têm desempenho muito próximo a de $\widetilde{\mathbf{C}}_{\mathbf{1}}$ a um bpp resultante maior. $\widehat{\mathbf{C}}_1$ e $\widehat{\mathbf{C}}_2$ conseguem melhores resultados que $\widetilde{\mathbf{C}}_{\mathbf{2}}$. Um exemplo para $QF = 90$ é ilustrado na Figura 5.5. Nota-se que, para esse caso, $\widetilde{\mathbf{C}}_{\mathbf{1}}$ leva a melhores resultados e menor tamanho de arquivo. Esse resultado não é representativo no caso médio, como mostram as Figuras 5.1 e 5.2.

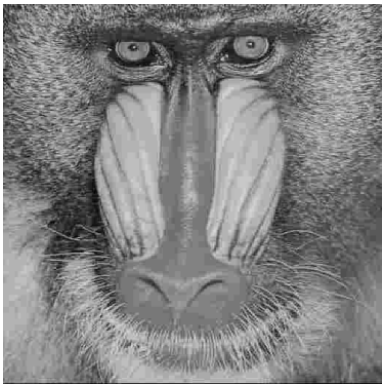
Além dos resultados de compressão de imagens apresentados, deve-se considerar as complexidades dessas transformadas. $\mathbf{C}$ pode ser computada com 11 multiplicações e 29 adições (LOEFFLER; LIGTENBERG; MOSCHYTZ, 1989). As transformadas aproximadas $\widehat{\mathbf{C}}_{\mathbf{1}}$ e $\widehat{\mathbf{C}}_{\mathbf{2}}$ requerem 24 e 22 adições, respectivamente (CINTRA; BAYER; TABLADA, 2014; CINTRA; BAYER, 2011; BAYER; CINTRA, 2010). As transformadas utilizadas para comparação, $\widetilde{\mathbf{C}}_{\mathbf{1}}$ e $\widetilde{\mathbf{C}}_{\mathbf{2}}$, requerem 14 e 22 adições, respectivamente (SILVEIRA et al., 2021; BAYER; CINTRA, 2012).



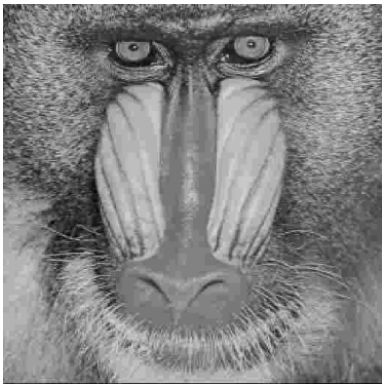
(a) PSNR: 23,4198 dB - SSIM: 0,7048 - bpp: 0,6621



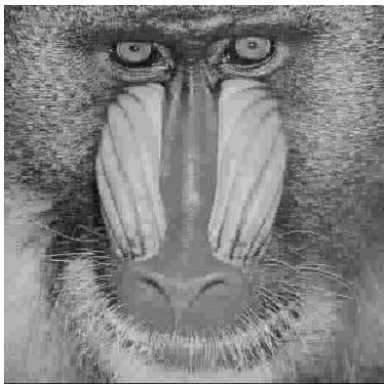
(b) PSNR: 23,0446 dB - SSIM: 0,6784 - bpp: 0,6436



(c) PSNR: 22,9872 dB - SSIM: 0,7970 - bpp: 0,6383



(d) PSNR: 22,6004 dB - SSIM: 0,65123 - bpp: 0,6049



(e) PSNR: 21,93927 dB - SSIM: 0,5935 - bpp: 0,5780

Figura 5.3 – Imagem *Baboo* comprimida com $QF = 10$. As transformadas (a) $\mathbf{C}$, (b) $\widehat{\mathbf{C}}_1$, (c) $\widehat{\mathbf{C}}_2$, (d) $\widetilde{\mathbf{C}}_\mathbf{1}$ e (e) $\widetilde{\mathbf{C}}_\mathbf{2}$ são usadas para compressão.

(a) PSNR: 35,8088 dB - SSIM: 0,9245 - bpp: 0,8707

(b) PSNR: 33,6895 dB - SSIM: 0,9026 - bpp: 0,9314

(c) PSNR: 33,7031 dB- SSIM: 0,9043 - bpp: 0,9304

(d) PSNR: 33,6933 dB - SSIM: 0,9036 - bpp: 0,9127

(e) PSNR: 32,3763 dB - SSIM: 0,8800 - bpp: 0,9983

Figura 5.4 – Imagem *Lenna* comprimida com $QF = 50$. As transformadas (a) $\mathbf{C}$, (b) $\widehat{\mathbf{C}}_1$, (c) $\widehat{\mathbf{C}}_2$, (d) $\widetilde{\mathbf{C}}_\mathbf{1}$ e (e) $\widetilde{\mathbf{C}}_\mathbf{2}$ são usadas para compressão.

(a) PSNR: 38,8555 dB - SSIM: 0,9466 - bpp: 2,5963

(b) PSNR: 38,5142 dB - SSIM: 0,9438 - bpp: 2,8689

(c) PSNR: 38,5778 dB - SSIM: 0,9447 - bpp: 2,8821

(d) PSNR: 38,6627 dB - SSIM: 0,9462 - 2,8533

(e) PSNR: 38,4722 dB - SSIM: 0,9437 - bpp: 3,0136

Figura 5.5 – Imagem *Peppers* comprimida com $QF = 90$. As transformadas (a) $\mathbf{C}$, (b) $\widehat{\mathbf{C}}_1$, (c) $\widehat{\mathbf{C}}_2$, (d) $\widetilde{\mathbf{C}}_\mathbf{1}$ e (e) $\widetilde{\mathbf{C}}_\mathbf{2}$ são usadas para compressão.

6 CONCLUSÃO

Este trabalho apresentou arquiteturas para ANNs do tipo *perceptron* com uma e duas camadas (MLP) para a computação da DCT e aproximações. As parametrizações dos pesos, a função de perda e regularizadores para discretização e ortogonalidade foram determinados para otimização das ANNs. Testes variando os hiper-parâmetros α e λ_1 da MLP possibilitaram derivar duas transformadas ortogonais. Essas transformadas são conhecidas na literatura. Tais transformadas são avaliadas de acordo com métricas de codificação e proximidade com relação à DCT. A complexidade aritmética dessas aproximações para a DCT também é apresentada. Por fim, as transformadas derivadas foram submetidas a um experimento de compressão de imagens.

Uma limitação da abordagem proposta é a difícil determinação dos valores dos hiper-parâmetros dos regularizadores. Há um relação não trivial envolvendo α, λ_1 e λ_2.

Entretanto, uma série de experimentos futuros devem comprovar o potencial da metodologia apresentada neste texto. Os trabalhos futuros vislumbrados neste documento são vastos e incluem:

1. Explorar a relação entre os hiper-parâmetros α, λ_1 e λ_2;
2. Modelar MLPs para a DCT e aproximações de outros comprimentos N;
3. Modelar MLPs para outras transformadas discretas lineares;
4. Modelar MLPs com várias camadas para derivar diretamente um algoritmo rápido;
5. Avaliar outras funções objetivo, inclusive com regularizadores que imponham pesos em outros domínios que não $\mathscr{T}$;
6. Realizar experimentos em compressão de vídeo;
7. Realizar implementações em *hardware*.

REFERÊNCIAS

AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete Cosine Transform. **IEEE Transactions on Computers**, C-23, n. 1, p. 90–93, jan 1974.

AKHTAR, J. Discrete fourier transform with neural networks. **IEEE Vehicular Technology Conference (VTC)**, p. 1–5, 2021.

ALARIFI, A. et al. A novel hybrid cryptosystem for secure streaming of high efficiency H.265 compressed videos in IoT multimedia applications. **IEEE Access**, v. 8, p. 128548–128573, 2020.

ALEMDAR, H. et al. Ternary neural networks for resource-efficient ai applications. **International Joint Conference on Neural Networks (IJCNN)**, 2017.

ALMURIB, H. A.; KUMAR, T. N.; LOMBARDI, F. Approximate DCT image compression using inexact computing. **IEEE Transactions on Computers**, v. 67, n. 2, p. 149–159, 2018.

ARAI, Y.; AGUI, T.; NAKAJIMA, M. A fast DCT-SQ scheme for images. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, v. 71, p. 1095–1097, 1988.

BAYER, F.; CINTRA, R. DCT-like transform for image compression requires 14 additions only. **Electronics Letters**, v. 48, n. 15, p. 919, 2012.

BAYER, F. et al. Multiplierless approximate 4-point DCT VLSI architectures for transform block coding. **Electronics Letters**, v. 49, n. 24, p. 1532–1534, nov. 2013.

BAYER, F. M.; CINTRA, R. J. Image compression via a fast DCT approximation. **IEEE Latin America Transactions**, v. 8, n. 6, p. 708–713, 2010.

BLAHUT, R. E. **Fast Algorithms for Signal Processing**. 2010.

BOUGUEZEL, S.; AHMAD, M.; SWAMY, M. Low-complexity 8×8 transform for image compression. **Electronics Letters**, v. 44, n. 21, p. 1249, 2008.

BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. A novel transform for image compression. **IEEE International Midwest Symposium on Circuits and Systems**, p. 509–512, 2010.

BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. A low-complexity parametric transform for image compression. **IEEE International Symposium on Circuits and Systems (ISCAS)**, 2011.

BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. N. S. A fast 8×8 transform for image compression. **International Conference on Microelectronics - ICM**, p. 74–77, 2009.

BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. N. S. Binary discrete cosine and hartley transforms. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 60, n. 4, p. 989–1002, 2013.

BRITANAK P. YIP, K. R. R. V. Discrete cosine and sine transforms. **Academic Press**, 2007.

CALLEBAUT, G. et al. The art of designing remote IoT devices—technologies and strategies for a long battery life. **Sensors**, MDPI AG, v. 21, n. 3, p. 913, jan. 2021.

CAMPOBELLO, G. et al. RAKE: A simple and efficient lossless compression algorithm for the internet of things. **European Signal Processing Conference (EUSIPCO)**, aug. 2017.

CANTERLE, D. R. et al. A multiparametric class of low-complexity transforms for image and video coding. **Signal Processing**, Elsevier BV, v. 176, p. 107685, nov. 2020.

CHAM, W.-K. Development of integer cosine transforms by the principle of dyadic symmetry. **IEE Proceedings I Communications, Speech and Vision**, Institution of Engineering and Technology (IET), v. 136, n. 4, p. 276, 1989.

CHEN, W.-H.; SMITH, C.; FRALICK, S. A fast computational algorithm for the discrete cosine transform. **IEEE Transactions on Communications**, v. 25, n. 9, p. 1004–1009, 1977.

CHOLLET, F. **Deep learning with Python**. 2018.

CINTRA, R.; BAYER, F.; TABLADA, C. Low-complexity 8-point DCT approximations based on integer functions. **Signal Processing**, Elsevier BV, v. 99, p. 201–214, jun. 2014.

CINTRA, R. J.; BAYER, F. M. A dct approximation for image compression. **IEEE Signal Processing Letters**, v. 18, n. 10, p. 579–582, 2011.

CLAY, V. et al. Learning sparse and meaningful representations through embodiment. **Neural Networks**, v. 134, p. 23–41, 2021. ISSN 0893-6080.

COELHO, D. F. G. et al. Low-complexity loeffler DCT approximations for image and video coding. **Journal of Low Power Electronics and Applications**, MDPI AG, v. 8, n. 4, p. 46, 2018.

COELHO, D. F. G. et al. Discrete transform approximations for video coding. **VLSI Architectures for Future Video Coding**, p. 257–297, aug. 2019.

COURBARIAUX, M.; BENGIO, Y.; DAVID, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. **Advances in neural information processing systems**, v. 28, 2015.

DARABI, S. et al. Regularized binary network training. **NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing**, 2020.

DEISENROTH, M. P.; FAISAL, A. A.; ONG, C. S. **Mathematics For Machine Learning**. 2020.

DENG, X.; ZHANG, Z. Sparsity-control ternary weight networks. **Neural Networks**, v. 145, p. 221–232, jan. 2022.

DENIL, M. et al. Predicting parameters in deep learning. **Advances in neural information processing systems**, v. 26, 2013.

FEIG, E.; WINOGRAD, S. Fast algorithms for the discrete cosine transform. **IEEE Transactions on Signal processing**, IEEE, v. 40, n. 9, p. 2174–2193, 1992.

FLURY, B. N.; GAUTSCHI, W. An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. **SIAM Journal on Scientific and Statistical Computing**, v. 7, n. 1, p. 169–184, 1986.

FONG, C.-K.; CHAM, W.-K. Llm integer cosine transform and its fast algorithm. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 6, p. 844–854, 2012.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. **MIT press**, p. 1, 2016. ISSN 1548-7091.

HAMZA, R.; HASSAN, A.; PATIL, A. S. A lightweight secure IoT surveillance framework based on DCT-DFRT algorithms. **Machine Learning for Cyber Security**, p. 271–278, 2019.

HAWEEL, R. T.; EL-KILANI, W. S.; RAMADAN, H. H. A fast modified signed discrete cosine transform for image compression. **International Conference on Computer Engineering Systems (ICCES)**, p. 56–61, 2014.

HAWEEL, T. I. A new square wave transform based on the dct. **Signal Processing**, v. 81, n. 11, p. 2309–2319, 2001.

HUBARA, I. et al. Binarized neural networks. **Advances in neural information processing systems**, v. 29, 2016.

JAYANT, N. Digital coding of speech waveforms: Pcm, dpcm, and dm quantizers. **Proceedings of the IEEE**, v. 62, n. 5, p. 611–632, 1974.

JRIDI, M.; ALFALOU, A.; MEHER, P. K. A generalized algorithm and reconfigurable architecture for efficient and scalable orthogonal approximation of DCT. **IEEE Transactions on Circuits and Systems I: Regular Papers**, 2015.

KIM, Y.-D. et al. Compression of deep convolutional neural networks for fast and low power mobile applications. **International Conference on Learning Representations (ICLR)**, 2015.

KOUADRIA, N. et al. Low complexity DCT for image compression in wireless visual sensor networks. **Electronics Letters**, Institution of Engineering and Technology (IET), v. 49, n. 24, p. 1531–1532, 2013.

KRONER, A. et al. Contextual encoder–decoder network for visual saliency prediction. **Neural Networks**, v. 129, p. 261–270, 2020.

LEE, B. A new algorithm to compute the discrete cosine transform. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 32, n. 6, p. 1243–1245, 1984.

LIANG, J.; TRAN, T. Approximating the DCT with the lifting scheme: systematic design and applications. **Fourth Asilomar Conference on Signals, Systems and Computers**, v. 1, p. 192–196, 2000.

LIN, X.; ZHAO, C.; PAN, W. Towards accurate binary convolutional neural network. **Advances in neural information processing systems**, v. 30, 2017.

LOEFFLER, C.; LIGTENBERG, A.; MOSCHYTZ, G. Practical fast 1-d dct algorithms with 11 multiplications. **International Conference on Acoustics, Speech, and Signal Processing,**, p. 988–991 vol.2, 1989.

MA, Z. et al. A detection and relative direction estimation method for UAV in sense-and-avoid. **IEEE International Conference on Information and Automation**, aug. 2015.

OLIVEIRA, P. A. M. et al. JPEG quantisation requires bit-shifts only. **Electronics Letters**, v. 53, n. 9, p. 588–590, 2017.

OLIVEIRA, R. S. et al. Low-complexity 8-point DCT approximation based on angle similarity for image and video coding. **Multidimensional Systems and Signal Processing**, 2018.

PENNEBAKER, W. B.; MITCHELL, J. L. **JPEG Still Image Data Compression Standard**. 1992.

RADÜNZ, A. P. et al. Extensions on low-complexity DCT approximations for larger blocklengths based on minimal angle similarity. **Journal of Signal Processing Systems**, Springer Science and Business Media LLC, 2023.

REDMON, J. et al. You only look once: Unified, real-time object detection. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 779–788, 2016.

REN, W. et al. Deep non-blind deconvolution via generalized low-rank approximation. **Advances in Neural Information Processing Systems**, v. 31, 2018.

ROMA, N.; SOUSA, L. Efficient hybrid dct-domain algorithm for video spatial downscaling. **EURASIP Journal on Advances in Signal Processing**, Springer, p. 1–16, 2007.

SILVEIRA, T. L. T. da et al. A class of low-complexity DCT-like transforms for image and video coding. **IEEE Transactions on Circuits and Systems for Video Technology**, 2021.

SOUDRY, D.; HUBARA, I.; MEIR, R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. **Advances in neural information processing systems**, v. 27, 2014.

TABLADA, C.; BAYER, F.; CINTRA, R. A class of DCT approximations based on the Feig–Winograd algorithm. **Signal Processing**, v. 113, p. 38–51, aug 2015.

TAKALA, J.; NIKARA, J. Unified pipeline architecture for discrete sine and cosine transforms of type IV. **Proceedings of the International Conference on Information, Communications, and Signal Processing**, p. 1–5, 2001.

TANG, W.; HUA, G.; WANG, L. How to train a compact binary neural network with high accuracy? **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 31, n. 1, 2017.

USC-SIPI. **The USC-SIPI Image Database**. http://sipi.usc.edu/database/, 2011. University of Southern California, Signal and Image Processing Institute. Available from Internet: <http://sipi.usc.edu/database/>.

VELIK, R. Discrete Fourier Transform computation using neural networks. **International Conference on Computational Intelligence and Security**, v. 1, p. 120–123, 2008.

WALLACE, G. K. The jpeg still picture compression standard. **Communications of the ACM**, AcM New York, NY, USA, v. 34, n. 4, p. 30–44, 1991.

WANG, Z. Fast algorithms for the discrete w transform and for the discrete fourier transform. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 32, n. 4, p. 803–816, 1984.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, 2004.

WATKINS, D. S. **Fundamentals of matrix computations**. 2. ed. Newy York: Wiley-Interscience, 2004. (Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts).

ZHANG, A. et al. **Dive Into Deep Learning**. 2020.

ZHAO, X. et al. Transform coding in the VVC standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 31, n. 10, p. 3878–3890, 2021.