UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

Carolina Rodrigues Menezes

Um Estudo de Redes Neurais Convolucionais de Valor Vetorial para Classificação de Imagens Coloridas

Campinas
25/06/2025

Carolina Rodrigues Menezes

Um Estudo de Redes Neurais Convolucionais de Valor Vetorial para Classificação de Imagens Coloridas*

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. Dr. Marcos Eduardo Ribeiro do Valle Mesquita.

*Este trabalho foi financiado Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) através do Programa Institucional de Bolsas de Iniciação Científica (PIBIC) da Universidade Estadual de Campinas (Unicamp), quota 2024-2025.

Resumo

Redes neurais têm ganhado destaque significante nos últimos anos desde o desenvolvimento de aprendizado de máquinas profundo. Redes neurais de valor vetorial (V-nets), incluindo as redes de valor hipercomplexo, são apropriadas para o processamento de dados multidimensionais, como imagens coloridas e hiperespectrais. Neste projeto, apresentaremos a base matemática para o desenvolvimento de V-nets. Implementaremos e aplicaremos esses modelos em um problema de classificação de imagens coloridas, especificamente para detectar leucemia linfoblástica aguda em imagens de esfregaço de sangue.

Abstract

Neural networks have gained significant attention in recent years since the development of deep learning. Vector-valued neural networks (V-nets), including hypercomplex-valued neural networks, are appropriate for processing multidimensional data, such as color and hyperspectral images. In this article, we present the mathematical foundation for the development of V-nets. We implement and apply these models for a color image classification task, specifically for detecting acute lymphoblastic leukemia in blood smear images.

Conteúdo

1 Introdução

Nos últimos anos, o aprendizado de máquina influenciou a forma como resolvemos uma variedade de problemas do mundo real (Géron, 2019; Goodfellow et al., 2016). De fato, as redes neurais (NN, do inglês *neural networks*) superaram muitas abordagens de última geração em várias aplicações com o desenvolvimento de arquiteturas de redes neurais profundas (DNN, do inglês *deep neural networks*). As redes neurais profundas se tornaram populares, em partes, devido a um aumento significativo no poder computacional nas últimas décadas. As redes neurais convolucionais (CNN, do inglês *convolutional neural networks*) são exemplos de rede neurais profundas aplicadas no processamento de sinais e imagens. As CNNs contém um tipo particular de operação baseada na convolução de filtros, sendo bem adequadas para aprender e representar padrões locais. Consequentemente, as CNNs são frequentemente consideradas como base dos modelos de ponta em processamento de imagens e reconhecimento de padrões.

Neste relatório, consideraremos redes neurais convolucionais em que as operações aritméticas são definidas numa álgebra não-associativa, ou seja, um espaço vetorial equipado com uma operação de produto (Schafer, 1961). Diferente dos modelos tradicionais de redes neurais profundas, redes neurais de valor vetorial (V-nets) utilizam vetores em vez de números reais para representar as entradas, saídas e os pesos sinápticos (Valle, 2024). Portanto, as V-nets são adequadas para processar dados multidimensionais como imagens coloridas e hiperespectrais. Além disso, as V-nets incluem as redes de valor hipercomplexos como as redes neurais baseadas nas álgebras de Clifford como casos particulares (Buchholz and Sommer, 2001, 2008; Vallejo and Bayro-Corrochano, 2008; Bayro-Corrochano et al., 2005). Consequentemente, as V-nets podem se beneficiar das propriedades geométricas da álgebra utilizada. Vários trabalhos demonstram a vantagem das V-nets, especificamente das redes hipercomplexas, sobre modelos equivalentes com valores reais, especialmente em tarefas envolvendo dados de vários canais, como o processamento de imagens (Breuils et al., 2022; Parcollet et al., 2020; Vieira and Valle, 2022a). Trabalhos recentes também mostram que redes em álgebras hipercomplexas se destacam na redução da complexidade computacional, ao mesmo tempo que fornecem desempenho semelhante ou superior em comparação com modelos com valores reais (Grassucci et al., 2023, 2022; Vieira and Valle,

2022b).

Nesse trabalho, revisaremos os conceitos básicos das álgebras não-associativas e estudaremos modelos tradicionais de redes neurais convolucionais. Posteriormente, estudaremos a implementação das V-nets usando uma biblioteca especifica para aprendizado de máquinas como `Keras`. Finalmente, investigaremos aplicações das V-nets em problemas de classificação de imagens como, por exemplo, no diagnóstico de leucemia linfoblástica aguda (Granero et al., 2021; Vieira and Valle, 2022b).

2 Álgebras Não-Associativas

Uma álgebra não-associativa é um espaço vetorial com uma multiplicação de vetores (Schafer, 1961). Formalmente, uma álgebra não-associativa $\mathbb{V}$ é um espaço vetorial sobre um corpo $\mathbb{F}$ com uma operação bilinear adicional, chamada multiplicação ou produto, que não necessariamente é associativa. Como uma operação bilinear, a multiplicação de $x, y \in \mathbb{V}$, denotada por xy, satisfaz

$$\begin{aligned} (x+y)z = x + yz, \quad z(x+y) = zx + zy, \quad \forall x, y, z \in \mathbb{V} \\ \text{e} \quad \alpha(xy) = (\alpha x)y = x(\alpha y), \quad \forall \alpha \in \mathbb{F} \text{ e } \forall x, y \in \mathbb{V}. \end{aligned} \tag{1}$$

Para implementar essas operações computacionalmente, simplificamos ao considerar $\mathbb{F} = \mathbb{R}$. Além disso, consideraremos apenas espaços vetoriais de dimensão finita.

Dada uma base ordenada $\mathcal{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$ em $\mathbb{V}$ e $x \in \mathbb{V}$, existe uma única tupla $(x_1, \ldots, x_n) \in \mathbb{R}^n$ tal que

$$x = \sum_{i=1}^{n} x_i \boldsymbol{e}_i \tag{2}$$

onde $x_1, \ldots x_n$ são as coordenadas x relacionadas à base ordenada $\mathcal{E}$. Apesar de serem equivalentes, podemos distinguir $\mathbb{V}$ de $\mathbb{R}^n$ ao introduzir a aplicação $\varphi : \mathbb{V} \to \mathbb{R}^n$, dada por

$$\varphi(x) = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \quad \forall x \in \mathbb{V}. \tag{3}$$

A aplicação φ é um isomorfismo entre $\mathbb{V}$ e $\mathbb{R}^n$. Assim, $\mathbb{V}$ herda a topologia e a métrica de

$\mathbb{R}^n$.

A multiplicação em $\mathbb{V}$ é completamente caracterizada pelo produto dos elementos da base $\mathcal{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$. Precisamente, seja

$$\boldsymbol{e}_i \boldsymbol{e}_j = \sum_{k=1}^{n} p_{ijk} \boldsymbol{e}_k, \quad \forall i, j = 1, \ldots, n. \tag{4}$$

o produto entre dois elemento de $\mathcal{E}$. Esse produto pode ser organizado na chamada tabela de multiplicação mostrada na Tabela 1. A tabela de multiplicação caracteriza completamente a álgebra não-associativa. Ademais, é possível inferir as propriedades de uma álgebra não-associativa. Por exemplo, uma álgebra é comutativa se e somente se $\boldsymbol{e}_i \boldsymbol{e}_j = \boldsymbol{e}_j \boldsymbol{e}_i$, para todo $i, j = 1, \ldots, n$. Equivalentemente, temos $p_{ijk} = p_{jik}$ para todo $i, j, k = 1, \ldots, n$, o que significa que a tabela de multiplicação é simétrica.

Tabela 1: Tabela de multiplicação genérica.

	e_j
	$\vdots$
e_i	$\cdots \quad \sum_{k=1}^{n} p_{ijk} e_k \quad \cdots$
	$\vdots$

2.1 Álgebra Hipercomplexa

Uma álgebra hipercomplexa $\mathbb{H}$ é uma álgebra não-associativa de dimensão finita onde o produto tem uma identidade pelos dois lados (Kantor and Solodovnikov, 1989; Valle, 2024). Relembrando, $\boldsymbol{e}_0 \in \mathbb{V}$ é uma identidade de dois lados se

$$x\boldsymbol{e}_0 = \boldsymbol{e}_0 x, \quad \forall x \in \mathbb{H}. \tag{5}$$

A identidade $\boldsymbol{e}_0$, também denotada por $\mathbf{1}$, é geralemnte o primeiro elemento de uma base ordenada $\mathcal{E}$ de uma álgebra hipercomplexa. De acordo com isso, a base canônica de uma álgebra hipercomplexa $\mathbb{H}$ de dimensão $dim(\mathbb{H}) = n+1$ é $\tau = \{\mathbf{1}, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_n\}$, e um número hipercomplexo é dado por $x = x_0 + x_1\boldsymbol{i}_1 + \ldots + x_n\boldsymbol{i}_n$. A figura 1 mostra a tabela de

multiplicação de números complexos, quatérnios e octônios em relação à base canônica $\tau_{\mathbb{C}} = \{\mathbf{1}, \boldsymbol{i_1}\}$, $\tau_{\mathbb{Q}} = \{\mathbf{1}, \boldsymbol{i_1}, \boldsymbol{i_2}, \boldsymbol{i_3}\}$, e $\tau_{\mathbb{O}} = \{\mathbf{1}, \boldsymbol{i_1}, \boldsymbol{i_2}, \boldsymbol{i_3}, \boldsymbol{i_4}, \boldsymbol{i_5}, \boldsymbol{i_6}, \boldsymbol{i_7}\}$, respectivamente.

Complexos Quatérnios Octônios

1	i_1	i_2	i_3	i_4	i_5	i_6	i_7
i_1	-1	i_3	$-i_2$	i_5	$-i_4$	$-i_7$	i_6
i_2	$-i_3$	-1	i_1	i_6	i_7	$-i_4$	$-i_5$
i_3	i_2	$-i_1$	-1	i_7	$-i_6$	i_5	$-i_4$
i_4	$-i_5$	$-i_6$	$-i_7$	-1	i_1	i_2	i_3
i_5	i_4	$-i_7$	i_6	$-i_1$	-1	$-i_3$	i_2
i_6	i_7	i_4	$-i_5$	$-i_2$	i_3	-1	$-i_1$
i_7	$-i_6$	i_5	i_4	$-i_3$	$-i_2$	i_3	-1

Figura 1: Tabela de multiplicação de números complexos, quatérnios e octônios em relação à base canônica.

2.2 Operação Produto

Usando a distributividade e a tabela de multiplicação, o produto de $x, y \in \mathbb{V}$ é dado por

$$xy = \left(\sum_{i=1}^{n} x_i \boldsymbol{e}_i\right)\left(\sum_{j=1}^{n} y_j \boldsymbol{e}_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_i (\boldsymbol{e}_i \boldsymbol{e}_j) = \sum_{k=1}^{n}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_j p_{ijk}\right)\boldsymbol{e}_k. \tag{6}$$

Seja $\mathcal{B}_k : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ dado por

$$\mathcal{B}_k(x, y) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_j p_{ijk}, \quad \forall k = 1, .., n. \tag{7}$$

temos $xy = \sum_{k=1}^{n} \mathcal{B}_k(x,y)\boldsymbol{e}_k$. Além disso, a função $\mathcal{B}_k$ é um mapeamento bilinear cuja representação matricial na base ordenada $\mathcal{E}$ é

$$B_k = \begin{bmatrix} p_{11k} & p_{12k} & \dots & p_{1nk} \\ p_{21k} & p_{22k} & \dots & p_{2nk} \\ p_{n1k} & p_{n2k} & \dots & p_{nnk} \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad \forall k = 1, \dots, n. \tag{8}$$

Assim, $\mathcal{B}_k(x,y) = \varphi(x)^T B_k \varphi(y)$.

Alternativamente, o produto pode ser obtido por uma operação entre uma matriz e um vetor. Dado o vetor $a = \sum_{i=1}^{n} a_i \boldsymbol{e}_i \in \mathbb{V}$, a multiplicação de x pela esquerda por a gera um operador linear $\mathcal{A}_L : \mathbb{V} \to \mathbb{V}$ dado por $\mathcal{A}_L = ax$, para todo $x \in \mathbb{V}$. Assim, a representação matricial de $\mathcal{A}_L$ em relação à base $\mathcal{E}$ gera a aplicação $M_L : \mathbb{V} \to \mathbb{R}^{n\times n}$ dado por

$$\begin{aligned} M_L(a) &= \begin{bmatrix} | & | & & | \\ \varphi(a\boldsymbol{e}_1) & \varphi(a\boldsymbol{e}_2) & \dots & \varphi(a\boldsymbol{e}_n) \\ | & | & & | \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^{n} a_i p_{i11} & \sum_{i=1}^{n} a_i p_{i11} & \dots & \sum_{i=1}^{n} a_i p_{i11} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} a_i p_{i11} & \sum_{i=1}^{n} a_i p_{i11} & \dots & \sum_{i=1}^{n} a_i p_{i11} \end{bmatrix} \\ &= \sum_{i=1}^{n} a_i P_i^T, \quad \text{onde} \quad P_i^T = \begin{bmatrix} p_{i11} & \dots & p_{in1} \\ \vdots & \ddots & \vdots \\ p_{i1n} & \dots & p_{inn} \end{bmatrix}. \end{aligned} \tag{9}$$

Resumidamente, $M_L : \mathbb{V} \to \mathbb{R}^{n\times n}$ mapeia $a \in \mathbb{V}$ à sua representação matricial em sua multiplicação pela esquerda em relação à base $\mathcal{E} = \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_n\}$. Usando a representação matricial, o produto ax é dado por

$$\varphi(ax) = M_L(a)\varphi(x) = \sum_{i=1}^{n} a_i P_i^T \varphi(x), \quad \forall a = \sum_{i=1}^{n} a_i \boldsymbol{e}_i, x \in \mathbb{V}. \tag{10}$$

Example 1 *O produto entre os quatérnios* $a = a_0 + a_1\boldsymbol{i_1} + a_2\boldsymbol{i_2} + a_3\boldsymbol{i_3}$ *e* $x = x_0 + x_1\boldsymbol{i_1} +$

$x_2\boldsymbol{i_2} + x_3\boldsymbol{i_3}$ *é dado por*

$$\begin{aligned} ax =&(a_0x_0 - a_1x_1 - a_2x_2 - a_3x_3) + (a_0x_1 + a_1x_0 + a_2x_3 - a_3x_2)\boldsymbol{i_1} \\ &+ (a_0x_2 - a_1x_3 + a_2x_0 + a_3x_1)\boldsymbol{i_2} + (a_0x_3 + a_1x_2 - a_2x_1 + a_3x_0)\boldsymbol{i_3}. \end{aligned} \tag{11}$$

Usando o isomorfismo φ *dado por* (3)*, obtemos*

$$\varphi(ax) = \begin{bmatrix} a_0x_0 - a_1x_1 - a_2x_2 - a_3x_3 \\ a_0x_1 + a_1x_0 + a_2x_3 - a_3x_2 \\ a_0x_2 - a_1x_3 + a_2x_0 + a_3x_1 \\ a_0x_3 + a_1x_2 - a_2x_1 + a_3x_0 \end{bmatrix}, \tag{12}$$

que pode ser reescrito como $\varphi(ax) = \mathcal{M}_L(a)\varphi(x)$ *com*

$$M_L(a) = \begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix} \quad and \quad \varphi(x) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{13}$$

3 Computação Matricial de Valor Vetorial

Similarmente à álgebra matricial tradicional, o produto entre duas matrizes de valor vetorial $A \in \mathbb{V}^{M\times L}$ e $B \in \mathbb{V}^{L\times N}$ é uma nova matriz $C \in \mathbb{V}^{M\times N}$ com entradas dadas por

$$c_{ij} = \sum_{l=1}^{L} a_{il}b_{lj}, \quad \forall i = 1, \ldots, M \quad e \quad j = 1, \ldots, N. \tag{14}$$

Para podermos usar softwares de computação científica rápidos, a operação acima é computada por operações matriciais de valor real. Usando o isomorfismo $\varphi : \mathbb{V} \to \mathbb{R}^n$ e a aplicação $M_L : \mathbb{V} \to \mathbb{R}^{n\times n}$, obtemos

$$\varphi(c_{ij}) = \sum_{l=1}^{L} \varphi(a_{il}b_{lj}) = \sum_{l=1}^{L} M_L(a_{il})\varphi(b_{lj}) \tag{15}$$

que é equivalente à operação matricial de valor real

$$\varphi(C) = M_L(A)\varphi(B), \tag{16}$$

onde

$$M_L(A) = \begin{bmatrix} M_L(a_{11}) & \dots & M_L(a_{1L}) \\ \vdots & \ddots & \vdots \\ M_L(a_{M1}) & \dots & M_L(a_{ML}) \end{bmatrix} \in \mathbb{R}^{nM \times nL}, \tag{17}$$

e

$$\varphi(B) = \begin{bmatrix} \varphi(b_{11}) & \dots & \varphi(b_{1N}) \\ \vdots & \ddots & \vdots \\ \varphi(b_{L1}) & \dots & \varphi(b_{LN}) \end{bmatrix} \in \mathbb{R}^{nL \times N}. \tag{18}$$

Então, temos $C = \varphi^{-1}(M_L(A)\varphi(B))$, o que permite que a computação das operações matriciais de valor vetorial sejam feitas a partir da álgebra linear de valor real disponível nos atuais softwares de computação científica.

3.1 O produto de Kronecker

Para reduzir ainda mais o tempo de computação, a matriz $M_L(A) \in \mathbb{R}^{nM \times nL}$ pode ser computada usando o produto de Kronecker (Loan, 2000).

O produto de Kronecker de duas matrizes de valor real $A = (a_{ij}) \in \mathbb{R}^{N \times M}$ e $B = (b_{ij}) \in \mathbb{R}^{P \times Q}$ é dado por

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1M}B \\ \vdots & \ddots & \vdots \\ a_{N1}B & \dots & a_{NM}B \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \tag{19}$$

Usando o produto de Kronecker, temos

$$M_L(A) = \sum_{k=1}^{n} \begin{bmatrix} a_{11k}P_k^T & \dots & a_{1Lk}P_k^T \\ \vdots & \ddots & \vdots \\ a_{M1k}P_k^T & \dots & a_{MLk}P_k^T \end{bmatrix} = \sum_{k=1}^{n} A_k \otimes P_i^T, \tag{20}$$

onde $A = \sum_{k=1}^{n} A_k \boldsymbol{e}_k$, com as matrizes de valor real $A_k \in \mathbb{R}^{M \times L}$ para todo $k = 1, \dots, n$.

Consequentemente, o produto matricial de valor vetorial $C = AB$ pode ser computado por

$$C = \varphi^{-1}\left(\left[\sum_{k=1}^{n} A_k \otimes P_k^T\right] \varphi(B)\right). \tag{21}$$

4 Redes Neurais de Valor Vetorial

Em redes neurais tradicionais, conhecidas como redes neurais de valor real, os dados são representados como vetores multidimensionais de números reais. Em contrapartida, redes neurais de valor vetorial (V-nets) representam os dados como "vetores de vetores". Essa representação permite que V-nets naturalmente levem em conta a intercorrelação entre canais de características usando álgebra vetorial, tornando estes modelos menos propensos a pararem em mínimos locais durante o treinamento. Assim, V-nets geralmente têm o treinamento mais robusto em comparação a redes tradicionais e tipicamente têm menos parâmetros. Quando trabalhando com dados de valor vetorial, como sinais de áudio ou imagens coloridas em RGB, modelos construídos especificamente para lidar com "vetores de vetores" podem performar melhor do que modelos tradicionais com vetores de números reais. A seção a seguir revisa as camadas densa e convolucional usadas em redes de valor vetorial, como descrito em (Valle, 2024).

4.1 Camadas densas

Em redes neurais, camadas densas consistem em múltiplos neurônions paralelos, cada um recebendo entradas através de conexões sinápticas. Cada neurônio computa a combinação linear de suas entradas, com os pesos sendo parâmetros treináveis chamados de pesos sinápticos. Um termo escalar de viés é adicionado a esta combinação. Além disso, a saída de cada neurônio pode ser transformada usando uma função de ativação não linear. Matematicamente, a saída do i-ésimo neurônio de valor vetorial em uma camada densa pode ser expressa como:

$$y_i = \psi(s_i + b_i), \quad \text{onde} \quad s_i = \sum_{j=1}^{N} \omega_{ij} x_j, \quad \forall i = 1, \ldots, M, \tag{22}$$

onde $x_1, \ldots, x_N \in \mathbb{V}$ representa as entradas de valor vetorial, ω_{ij} são os pesos sinápticos conectando a entrada j ao neurônio i, $b_i \in \mathbb{V}$ é o termo de viés para o neurônio i, e $\psi : \mathbb{V} \to \mathbb{V}$ denota a função de ativação de valor vetorial.

Vale lembrar que camadas densas tradicionais e de valor vetorial são equivalentes quando a dimensão do espaço vetorial $\mathbb{V}$ é igual a 1, ou seja, quando $dim(\mathbb{V}) = 1$.

4.2 Camadas convolucionais

Neurônios de camadas convolucionais, diferentemente dos de camadas densas, são conectados apenas a uma seção do mapa de características da camada anterior através de um filtro. Além de reduzirem significantemente o número de parâmetros, camadas convolucionais adicionam certa invariância espacial ao modelo e são eficazes na detecção de padrões locais.

Seja x a imagem de entrada com C canais de características de valor vetorial. Denotamos como $x(p, c) \in \mathbb{V}$ o conteúdo do c-ésimo canal de característica localizado no pixel $p \in D_x$, onde D_x é o domínio de x. Os pesos de uma camada convolucional com K filtros são arranjados em um vetor W tal que $W(q, c, k) \in \mathbb{V}$ é o peso do k-ésimo filtro no c-ésimo mapa de características em $q \in D$, onde D é o domínio do filtro. Assim, a convolução de valor vetorial de W e x é dada por

$$(W * x)(p, k) = \sum_{c=1}^{C} \sum_{q \in D} W(q, c, k) x(p + S(q), c), \ p \in D_x, \ \forall k = 1, \ldots, K \tag{23}$$

onde $p + S(q)$ é uma translação que pode contemplar *strides*. A saída de uma camada convolucional é obtida ao adicionar um termo de viés e aplicar uma função de ativação da seguinte forma:

$$y = \psi(W * x + b). \tag{24}$$

Usando o isomorfismo $\varphi : \mathbb{V} \to \mathbb{R}^n$, o produto de Kronecker e linearidade, podemos expressar a operação de convolução (23) através da equação

$$\varphi(W * x) = \left(\sum_{l=1}^{n} W_l \otimes P_{l:}^T \right) * \varphi(x) = \sum_{l=1}^{n} M_l * \varphi(x), \tag{25}$$

onde $W = W_1\boldsymbol{e}_1 + \ldots + W_n\boldsymbol{e}_n$ é uma representação dos filtros em relação à base ordenada $\mathcal{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, $M_l = W_l \otimes P_{i:}^T$ são filtros de valor real obtidos usando o produto de Kronecker e $\varphi(x)$ é obtido concatenando as componentes de $x = x_1\boldsymbol{e}_1 + \ldots + x_n\boldsymbol{e}_n$ no eixo das características.

4.3 Combinando camadas tradicionais e de valor vetorial

CNNs tradicionais incluem, além de camadas densas e convolucionais, camadas de *pooling* (Lecun et al., 2015). Enquanto há estudos atualmente buscando a criação de camadas de *pooling* de valor vetorial, podemos usar um método que combina sua versão tradicional com camadas convolucionais e densas de valor vetorial. Esse método aplica de forma eficaz as camadas de *pooling* componente a componente (Valle, 2024). Além disso, V-nets podem incorporar uma camada de saída densa tradicional para problemas de classificação. Uma camada densa de valor real é equivalente a tomar a parte real de uma camada de valor hipercomplexo (Valle, 2024).

5 Experimento Computacional

Esta seção demonstra a implementação de uma rede neural convolucional de valor vetorial (V-CNN), utilizando diversas álgebras hipercomplexas, especificamente números complexos, quatérnios e octônios. Além disso, discutiremos a aplicação de V-CNNs em um problema de classificação de imagens associadas ao diagnóstico de leucemia linfoblástica aguda.

5.1 O problema de classificação

Consideramos o banco de dados "*ALL-IDB: Acute Lymphoblastic Leukemia Image Database for Image Processing*", desenvolvido por Fabio Scotti e colaboradores no *Department of Information Technology, Università degli Studi di Milano*. Este banco de dados contém 260 imagens no total, com 130 imagens classificadas como células saudáveis (classificação 0) e 130 imagens classificadas como linfoblastos (classificação 1). As imagens foram redimensionadas para 128×128 usando as cores RGB. O principal objetivo do nosso

modelo é prever a classificação de uma dada imagem, determinando se ela é de uma célula saudável ou de um linfoblasto.

5.2 Codificação Hipercomplexa: Complexos, Quatérnios e Octônios

Para desenvolver os modelos de valor complexo, quaterniônico e octoniônico, precisamos primeiro codificar estas entradas de valor real como valores hipercomplexos. As entradas de valor real são representadas por vetores (ou tensores) de tamanho $(128, 128, 3)$, onde as duas primeiras entradas correspondem à largura e à altura em pixels e a terceira entrada contém os valores reescalados das entradas RGB da imagem.

Seja $x_{\mathbb{R}} = [\mathtt{R}, \mathtt{G}, \mathtt{B}]$ os valores de RGB da imagem colorida. As seguintes estratégias de codificação foram usadas na codificação da imagem RGB em imagens de valor complexo e quaterniônico:

- **Codificação Complexa**: O pixel de cores RGB é codificado em dois números complexos pela equação:

$$x_{\mathbb{C}}^{(1)} = [\mathtt{R}\boldsymbol{i}, \mathtt{G} + \mathtt{B}\boldsymbol{i}]. \tag{26}$$

  Note que a primeira componente é um número complexo imaginário puro, ou seja, a sua parte real é nula.

- **Codificação Quaterniônica**: O pixel de cores RGB é codificado como um quatérnio pela equação:

$$x_{\mathbb{Q}}^{(1)} = \mathtt{R}\boldsymbol{i} + \mathtt{G}\boldsymbol{j} + \mathtt{B}\boldsymbol{k}. \tag{27}$$

Além da codificação dada por (26) e (27), consideramos também uma codificação parametrizada, dada pelas expressões:

$$\begin{aligned} x_{\mathbb{C}}^{(2)} = [&(\alpha_{1,0}\mathtt{R} + \alpha_{2,0}\mathtt{G} + \alpha_{3,0}\mathtt{B} + \beta_0) + (\alpha_{1,1}\mathtt{R} + \alpha_{2,1}\mathtt{G} + \alpha_{3,1}\mathtt{B} + \beta_1)\boldsymbol{i}, \\ &(\alpha_{1,2}\mathtt{R} + \alpha_{2,2}\mathtt{G} + \alpha_{3,2}\mathtt{B} + \beta_2) + (\alpha_{1,3}\mathtt{R} + \alpha_{2,3}\mathtt{G} + \alpha_{3,3}\mathtt{B} + \beta_3)\boldsymbol{i}] \end{aligned} \tag{28}$$

$$\begin{aligned} x_{\mathbb{Q}}^{(2)} = &(\alpha_{1,0}\mathtt{R} + \alpha_{2,0}\mathtt{G} + \alpha_{3,0}\mathtt{B} + \beta_0) + (\alpha_{1,1}\mathtt{R} + \alpha_{2,1}\mathtt{G} + \alpha_{3,1}\mathtt{B} + \beta_1)\boldsymbol{i} \\ &+ (\alpha_{1,2}\mathtt{R} + \alpha_{2,2}\mathtt{G} + \alpha_{3,2}\mathtt{B} + \beta_2)\boldsymbol{j} + (\alpha_{1,3}\mathtt{R} + \alpha_{2,3}\mathtt{G} + \alpha_{3,3}\mathtt{B} + \beta_3)\boldsymbol{k} \end{aligned} \tag{29}$$

Note que, como $x_{\mathbb{C}}^{(1)}$, a imagem de valor complexo $x_{\mathbb{C}}^{(2)}$ tem dois canais. De forma similar,

Tabela 2: Arquiteturas CNNs de valor real e hipercomplexo.

	Real		Complexo		Quatérnio		Octônio	
	filtros	parâmetros	filtros	parâmetros	filtros	parâmetros	filtros	parâmetros
Conv#1	32	896	16	608	8	320	4	320
MaxPool#1								
Conv#2	64	18.496	32	9.280	16	4.672	8	2.368
MaxPool#2								
Conv#3	128	73,856	64	36.992	32	18.560	16	9.344
MaxPool#3								
Conv#4	128	147.584	64	73.856	32	36.992	16	18.560
MaxPool#4								
Densa	2	9.218	2	16.386	2	16.386	2	16.386
# Total de parâmetros		250.050		137.138		76.946		47.010

a imagem RGB é codificada em um imagem de valor octoniônico pela equação

$$x_{\mathbb{O}} = (\alpha_{1,0}\mathtt{R} + \alpha_{2,0}\mathtt{G} + \alpha_{3,0}\mathtt{B} + \beta_0) + \sum_{k=1}^{7}(\alpha_{1,k}\mathtt{R} + \alpha_{2,k}\mathtt{G} + \alpha_{3,k}\mathtt{B} + \beta_k)\boldsymbol{i}_k. \tag{30}$$

Os parâmetros $\alpha_{i,k}$ e β_k são ajustados durante o treinamento dos modelos V-CNN.

5.3 Arquiteturas das Redes Neurais Convolucionais de Valor Vetorial

A Tabela 2 apresenta as arquiteturas dos modelos real e hipercomplexos, excluindo as camadas de codificação, *data augmentation* e *dropout*. Cada camada convolucional tem filtros de tamanho 3×3, com o argumento *padding* configurado para "SAME" e utiliza a função de ativação ReLU. Para o modelo hipercomplexo, utilizamos a camada `V_Conv2D` disponível em `https://github.com/mevalle/v-nets`, que usa álgebras hipercomplexas. A camada de saída é uma camada densa de valor real com dois neurônios—um para cada classe—e usa a função de ativação softmax. Como de costume, incluímos a camada *flatten* do Keras antes da última camada densa. Finalmente, treinamos 10 vezes cada um dos nossos 6 modelos, definidos por:

- **Modelo R:** A CNN de valor real;
- **Modelos C1 e C2:** As CNNs de valor complexo com codificações dadas por (26) e (28), respectivamente;
- **Modelos Q1 e Q2**: As CNNs de valor quaterniônico com codificações dadas por (27) e (29), respectivamente;

- **Modelo O:** A CNN de valor octoniônico com codificação dada por (30).

É válido lembrar que as arquiteturas CNN exibem um número total de parâmetros decrescente. Especificamente, a CNN de valor real tem 250.050 parâmetros, enquanto o modelo de valor octoniônico tem apenas 47.010 parâmetros (contados como números de ponto flutuante).

5.4 O treinamento

Durante o treinamento, utilizamos camadas de *data augmentation* e técnicas de regularização *dropout*. As camadas de *data augmentation* incluem espelhamento aleatório, rotações aleatórias (com ângulo máximo de 0,3 radianos), zooms aleatórios (com fator máximo de 0,2) e translações aleatórias (com translação máxima de (0,1, 0,1). A taxa de *dropout* é configurada para 0,5.

Utilizamos o otimizador Adam e usamos como função perda a entropia cruzada, com *batches* de tamanho 32 e um total de 200 épocas. O banco de dados original é dividido em conjuntos de treino (64%), validação (16%), e teste (20%), com o argumento "`random_state`" configurado para 42. Vale destacar que utilizamos a codificação *one-hot* para as classes. O processo de treinamento para cada modelo é repetido 10 vezes e os valores de acurácia e da função perda são salvos para uma análise comparativa.

5.5 Resultados e Análise

A Tabela 3 mostra a média e desvio padrão da acurácia de cada modelo no conjunto de teste. Para ajudar na interpretação dos resultados do experimento computacional, a Figura 2 ilustra o gráfico *boxplot* da acurácia atingida por cada modelo no conjunto de teste. Além disso, a Figura 3 apresenta um diagrama de Hasse, onde os modelos de melhor performance são posicionados no topo. Nesse diagrama, uma linha indica que o modelo acima performou melhor que o modelo abaixo, determinado por um teste de hipótese pareado com 95% de nível de significância.

Os modelos C1, C2 e Q2 exibiram as maiores varições, com os modelos C1 e Q2 tendo também os menores valores de acurácia. Por outro lado, os modelos R, C2, Q1 e O alcançaram as maiores acurácias e demonstraram equivalência estatística.

Tabela 3: A média e desvio padrão para a acurácia no conjunto de teste.

	Modelo R	Modelo C1	Modelo C2	Modelo Q1	Modelo Q2	Modelo O
média	0.967308	0.863462	0.934615	0.973077	0.926923	0.971154
desvio padrão	0.015832	0.196629	0.090291	0.013446	0.055736	0.018689

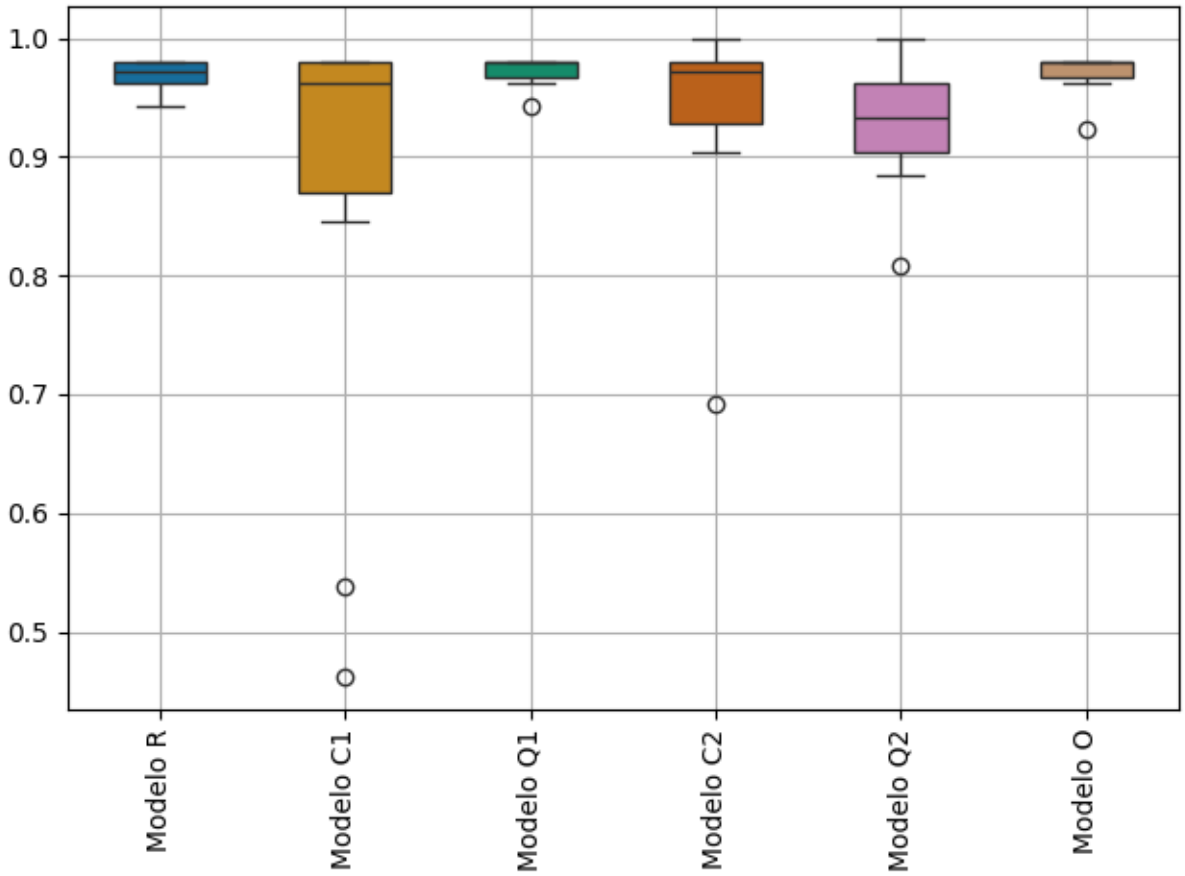

Figura 2: *Boxplot* com a acurácia dos modelos CNN no conjunto de teste.

Podemos concluir que V-Nets, especialmente as de valor hipercomplexo discutidas neste relatório, performam tão bem quanto ou mesmo ligeiramente melhor que suas versões de valor real. Isso é conquistado ao mesmo tempo que vemos uma redução significante do número de parâmetros, e, consequentemente, dos recursos computacionais necessários para armazenar estes modelos.

6 Conclusões

Neste relatório, revisamos as bases algébricas de redes neurais de valor vetorial e hipercomplexo. Demonstramos que modelos baseados nos números complexos,

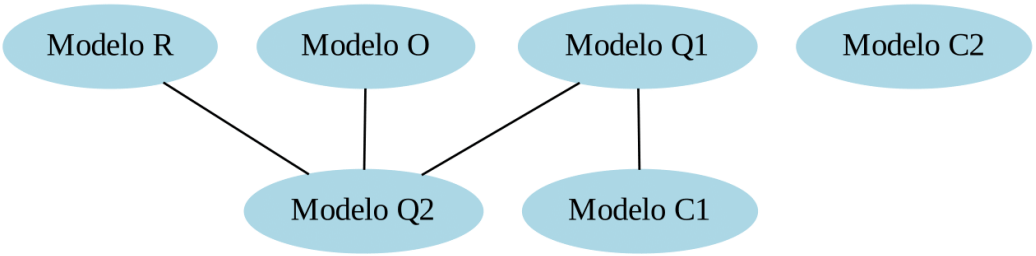

Figura 3: Diagrama Hasse comparando os modelos CNN no conjunto de teste.

quatérnios e octônios podem atuar na classificação de imagens coloridas de forma eficaz. Especificamente, avaliamos a performance de redes neurais de valor hipercomplexo usando o banco de dados *ALL-IDB* (Labati et al., 2011). Nossos resultados indicam que modelos de valor vetorial conseguem reduzir o número de parâmetros sem comprometer a acurácia. Essa conclusão é especialmente significativa no contexto de aprendizado profundo leve (*lightweight deep learning*, já que reduz significantemente o número de parâmetros do modelo.

Referências

Eduardo Bayro-Corrochano, Refugio Vallejo, and Nancy Arana-Daniel. Geometric preprocessing, geometric feedforward neural networks and Clifford support vector machines for visual learning. *Neurocomputing*, 67(1-4 SUPPL.):54–105, 8 2005. ISSN 0925-2312. doi: 10.1016/J.NEUCOM.2004.11.041.

Stephane Breuils, Kanta Tachibana, and Eckhard Hitzer. New Applications of Clifford's Geometric Algebra. *Advances in Applied Clifford Algebras 2022 32:2*, 32(2):1–39, 2 2022. ISSN 1661-4909. doi: 10.1007/S00006-021-01196-7. URL `https://link.springer.com/article/10.1007/s00006-021-01196-7`.

Sven Buchholz and Gerald Sommer. Clifford Algebra Multilayer Perceptrons. In *Geometric Computing with Clifford Algebras*, pages 315–334. Springer Berlin Heidelberg, 2001. doi: 10.1007/978-3-662-04621-0{_}13.

Sven Buchholz and Gerald Sommer. On Clifford neurons and Clifford multi-layer perceptrons. *Neural Networks*, 21(7):925–935, 9 2008. ISSN 08936080. doi: 10.1016/j.neunet.2008.03.004.

Aurelien Géron. *Hands–On Machine Learning with Scikit–Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O Reilly, Sebastopol, California, USA., 2nd edition, 10 2019. ISBN 1492032646.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

Marco Aurélio Granero, Cristhian Xavier Hernández, and Marcos Eduardo Valle. Quaternion-Valued Convolutional Neural Network Applied for Acute Lymphoblastic Leukemia Diagnosis. In *Brazilian Conference on Intelligent Systems. BRACIS 2021. Lecture Notes in Computer Science*, pages 280–293. Springer, Cham, 11 2021. ISBN 9783030916985. doi: 10.1007/978-3-030-91699-2{_}20. URL `https://link.springer.com/chapter/10.1007/978-3-030-91699-2_20`.

Eleonora Grassucci, Aston Zhang, and Danilo Comminiello. PHNNs: Lightweight Neural Networks via Parameterized Hypercomplex Convolutions. *IEEE Transactions*

on Neural Networks and Learning Systems, pages 1–13, 10 2022. ISSN 2162-237X. doi: 10.1109/TNNLS.2022.3226772. URL https://ieeexplore.ieee.org/document/9983846/.

Eleonora Grassucci, Gioia Mancini, Christian Brignone, Aurelio Uncini, and Danilo Comminiello. Dual quaternion ambisonics array for six-degree-of-freedom acoustic representation. *Pattern Recognition Letters*, 166:24–30, 2 2023. ISSN 01678655. doi: 10.1016/j.patrec.2022.12.006.

I. L. Kantor and A. S. Solodovnikov. *Hypercomplex Numbers: An Elementary Introduction to Algebras*. Springer New York, 1989. doi: 10.1007/978-1-4612-3650-4.

Ruggero Donida Labati, Vincenzo Piuri, and Fabio Scotti. All-idb: The acute lymphoblastic leukemia image database for image processing. In *2011 18th IEEE International Conference on Image Processing*, pages 2045–2048, 2011. doi: 10.1109/ICIP.2011.6115881.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 5 2015. ISSN 14764687. doi: 10.1038/NATURE14539.

Charles F.Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 11 2000. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00393-9.

Titouan Parcollet, Mohamed Morchid, and Georges Linarès. A survey of quaternion neural networks. *Artificial Intelligence Review*, 53(4):2957–2982, 4 2020. doi: 10.1007/s10462-019-09752-1.

Richard Schafer. *An Introduction to Nonassociative Algebras*. Project Gutenberg, 1961. URL https://www.gutenberg.org/ebooks/25156.

Marcos Eduardo Valle. Understanding Vector-Valued Neural Networks and Their Relationship With Real and Hypercomplex-Valued Neural Networks: Incorporating intercorrelation between features into neural networks [Hypercomplex Signal and Image Processing]. *IEEE Signal Processing Magazine*, 41(3):49–58, 5 2024. ISSN 1053-5888. doi: 10.1109/MSP.2024.3401621.

J R Vallejo and E Bayro-Corrochano. Clifford Hopfield Neural Networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3609–3612, 2008. doi: 10.1109/IJCNN.2008.4634314.

Guilherme Vieira and Marcos Eduardo Valle. A general framework for hypercomplex-valued extreme learning machines. *Journal of Computational Mathematics and Data Science*, 3:100032, 6 2022a. ISSN 2772-4158. doi: 10.1016/J.JCMDS.2022.100032. URL `https://linkinghub.elsevier.com/retrieve/pii/S2772415822000062`.

Guilherme Vieira and Marcos Eduardo Valle. Acute Lymphoblastic Leukemia Detection Using Hypercomplex-Valued Convolutional Neural Networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 7 2022b. ISBN 978-1-7281-8671-9. doi: 10.1109/IJCNN55064.2022.9892036.