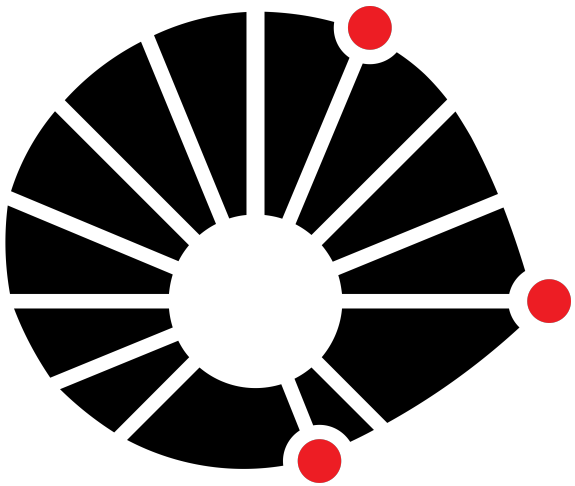
Bridging Vector-Valued Mathematical Morphology and Deep Learning

CaLISTA Workshop: Geometry–Informed Machine Learning



UNICAMP

Marcos Eduardo Valle

University of Campinas - Brazil

September 2, 2024

This work has been supported by
São Paulo Research Foundation and
Agence Nationale de la Recherche
(FAPESP-ANR, grant no 2023/03368-0).

Introduction

In this talk, we will review some basic concepts of mathematical morphology, with emphasis on vector-valued operators.

Introduction

In this talk, we will review some basic concepts of mathematical morphology, with emphasis on vector-valued operators.

Motivated by vector-valued mathematical morphology, we shall introduce vector-valued neural networks (V-nets), a broad class of neural networks that takes the intercorrelation between feature channels beforehand.

Introduction

In this talk, we will review some basic concepts of mathematical morphology, with emphasis on vector-valued operators.

Motivated by vector-valued mathematical morphology, we shall introduce vector-valued neural networks (V-nets), a broad class of neural networks that takes the intercorrelation between feature channels beforehand.

Moreover, V-nets can benefit from geometrical concepts of hypercomplex algebras.

Introduction

In this talk, we will review some basic concepts of mathematical morphology, with emphasis on vector-valued operators.

Motivated by vector-valued mathematical morphology, we shall introduce vector-valued neural networks (V-nets), a broad class of neural networks that takes the intercorrelation between feature channels beforehand.

Moreover, V-nets can benefit from geometrical concepts of hypercomplex algebras.

In addition to introducing V-nets, we show how they can be effectively implemented in deep-learning libraries.

Lattice Framework for Mathematical Morphology

Mathematical morphology (MM) provides geometrical and topological framework for the **analysis** and **processing** of images. (Heijmans, 1994; Serra, 1982; Soille, 1999).

Lattice Framework for Mathematical Morphology

Mathematical morphology (MM) provides geometrical and topological framework for the **analysis** and **processing** of images. (Heijmans, 1994; Serra, 1982; Soille, 1999).

The two elementary operators of mathematical morphology are dilation and erosion. Other operators are obtained by combining the two elementary operators.

Lattice Framework for Mathematical Morphology

Mathematical morphology (MM) provides geometrical and topological framework for the **analysis** and **processing** of images. (Heijmans, 1994; Serra, 1982; Soille, 1999).

The two elementary operators of mathematical morphology are dilation and erosion. Other operators are obtained by combining the two elementary operators.

In general, MM can be very well conducted in a mathematical framework called **complete lattice** (Birkhoff, 1993; Heijmans, 1995).

Definition (Complete Lattice)

A complete lattice $\mathcal{L}$ is a partially ordered set in which every subset (finite or infinite) has a supremum and an infimum.

The two elementary operators of mathematical morphology can be defined using the concept of adjunction (Heijmans, 1995)!

The two elementary operators of mathematical morphology can be defined using the concept of adjunction (Heijmans, 1995)!

Definition (Adjunction, Erosion and Dilation)

Let $\mathcal{L}$ and $\mathcal{M}$ be complete lattices. We say that two operators $\varepsilon : \mathcal{L} \to \mathcal{M}$ and $\delta : \mathcal{M} \to \mathcal{L}$ form an adjunction if

$$\delta(x) \leq y \quad \Longleftrightarrow \quad x \leq \varepsilon(y). \tag{1}$$

In this case, ε is an erosion and δ is a dilation.

The two elementary operators of mathematical morphology can be defined using the concept of adjunction (Heijmans, 1995)!

Definition (Adjunction, Erosion and Dilation)

Let $\mathcal{L}$ and $\mathcal{M}$ be complete lattices. We say that two operators $\varepsilon : \mathcal{L} \to \mathcal{M}$ and $\delta : \mathcal{M} \to \mathcal{L}$ form an adjunction if

$$\delta(x) \leq y \quad \Longleftrightarrow \quad x \leq \varepsilon(y). \tag{1}$$

In this case, ε is an erosion and δ is a dilation.

Furthermore, the compositions

$$\gamma = \delta \circ \varepsilon \quad \text{and} \quad \phi = \varepsilon \circ \delta,$$

represent an opening and a closing, respectively.

The two elementary operators of mathematical morphology can be defined using the concept of adjunction (Heijmans, 1995)!

Definition (Adjunction, Erosion and Dilation)

Let $\mathcal{L}$ and $\mathcal{M}$ be complete lattices. We say that two operators $\varepsilon : \mathcal{L} \to \mathcal{M}$ and $\delta : \mathcal{M} \to \mathcal{L}$ form an adjunction if

$$\delta(x) \leq y \quad \Longleftrightarrow \quad x \leq \varepsilon(y). \tag{1}$$

In this case, ε is an erosion and δ is a dilation.

Furthermore, the compositions

$$\gamma = \delta \circ \varepsilon \quad \text{and} \quad \phi = \varepsilon \circ \delta,$$

represent an opening and a closing, respectively.

Morphological operators can be found in activation functions and pooling layers in current deep-learning models (Velasco-Forero and Angulo, 2022).

A morphological neural network is an artificial neural network whose neurons perform a morphological operator, possibly followed by an activation function (Sussner and Esmi, 2011).

A morphological neural network is an artificial neural network whose neurons perform a morphological operator, possibly followed by an activation function (Sussner and Esmi, 2011).

Examples of (shallow) morphological neural networks include:

- Morphological and fuzzy morphological associative memories (Ritter et al., 1998; Santos and Valle, 2018; Sussner and Valle, 2006; Valle and Sussner, 2008),
- Multi-layer morphological perceptron networks (Ritter and Sussner, 1996; Ritter and Urcid, 2003; Sussner and Esmi, 2011),
- Deep morphological networks Franchi et al. (2020); Mondal et al. (2019); Nogueira et al. (2021)

A morphological neural network is an artificial neural network whose neurons perform a morphological operator, possibly followed by an activation function (Sussner and Esmi, 2011).

Examples of (shallow) morphological neural networks include:

- Morphological and fuzzy morphological associative memories (Ritter et al., 1998; Santos and Valle, 2018; Sussner and Valle, 2006; Valle and Sussner, 2008),
- Multi-layer morphological perceptron networks (Ritter and Sussner, 1996; Ritter and Urcid, 2003; Sussner and Esmi, 2011),
- Deep morphological networks Franchi et al. (2020); Mondal et al. (2019); Nogueira et al. (2021)

Adjunction, greedy algorithms, evolutionary programming, a difference of convex functions, or variations/adaptations of the backpropagation are used to train morphological neural networks.

Gray-Scale Morphological Operators

A gray-scale image is a function $f : \mathcal{D} \to \mathbb{T}$, where $\mathcal{D}$ is the domain, and $\mathbb{T}$ is a chain such as $\mathbb{T} = [0, 1]$ or $\mathbb{T} = \{0, 1, \ldots, 255\}$.

Gray-Scale Morphological Operators

A gray-scale image is a function $f : \mathcal{D} \to \mathbb{T}$, where $\mathcal{D}$ is the domain, and $\mathbb{T}$ is a chain such as $\mathbb{T} = [0, 1]$ or $\mathbb{T} = \{0, 1, \ldots, 255\}$.

Given a structuring element $S \subset \mathcal{D}$, we define flat elementary morphological operators as follows for $x \in \mathcal{D}$:

$$[\varepsilon_S(f)](x) = \inf_{s \in S}\{f(x + s)\} \quad \text{and} \quad [\delta_S(f)](x) = \sup_{s \in S}\{f(x - s)\}. \quad (2)$$

Gray-Scale Morphological Operators

A gray-scale image is a function $f : \mathcal{D} \to \mathbb{T}$, where $\mathcal{D}$ is the domain, and $\mathbb{T}$ is a chain such as $\mathbb{T} = [0, 1]$ or $\mathbb{T} = \{0, 1, \ldots, 255\}$.

Given a structuring element $S \subset \mathcal{D}$, we define flat elementary morphological operators as follows for $x \in \mathcal{D}$:

$$[\varepsilon_S(f)](x) = \inf_{s \in S}\{f(x + s)\} \quad \text{and} \quad [\delta_S(f)](x) = \sup_{s \in S}\{f(x - s)\}. \tag{2}$$

Non-flat gray-scale morphological operators are defined as follows using a structuring function $g : \mathcal{D} \to \mathbb{T}$:

$$[\varepsilon_g(f)](x) = \inf_{s \in S}\{f(x + s) - g(s)\}, \tag{3}$$

and

$$[\delta_g(f)](x) = \sup_{s \in S}\{f(x - s) + g(s)\}. \tag{4}$$

Vector-Valued Mathematical Morphology

A vector-valued image is a function $\boldsymbol{f} : \mathcal{D} \to \mathbb{V}$, where $\mathcal{D}$ is the domain, and $\mathbb{V}$ is the value set, usually a subset of $\mathbb{R}^d$ with $d \geq 2$.

Vector-Valued Mathematical Morphology

A vector-valued image is a function $\boldsymbol{f} : \mathcal{D} \to \mathbb{V}$, where $\mathcal{D}$ is the domain, and $\mathbb{V}$ is the value set, usually a subset of $\mathbb{R}^d$ with $d \geq 2$.

Like gray-scale morphology, flat vector-valued morphological operators are defined as follows using a structuring element $S \subset \mathcal{D}$:

$$[\varepsilon_S(\boldsymbol{f})](x) = \inf_{s \in S}\{\boldsymbol{f}(x+s)\} \quad \text{and} \quad [\delta_S(\boldsymbol{f})](x) = \sup_{s \in S}\{\boldsymbol{f}(x-s)\}. \quad (5)$$

Vector-Valued Mathematical Morphology

A vector-valued image is a function $\boldsymbol{f} : \mathcal{D} \to \mathbb{V}$, where $\mathcal{D}$ is the domain, and $\mathbb{V}$ is the value set, usually a subset of $\mathbb{R}^d$ with $d \geq 2$.

Like gray-scale morphology, flat vector-valued morphological operators are defined as follows using a structuring element $S \subset \mathcal{D}$:

$$[\varepsilon_S(\boldsymbol{f})](x) = \inf_{s \in S}\{\boldsymbol{f}(x+s)\} \quad \text{and} \quad [\delta_S(\boldsymbol{f})](x) = \sup_{s \in S}\{\boldsymbol{f}(x-s)\}. \quad (5)$$

One of the main challenges in vector-valued mathematical morphology is to determine a suitable ordering so that $\mathbb{V}$ is a complete lattice!

Non-flat vector-valued morphological operators are defined by enriching a complete lattice with additional binary operations.

Non-flat vector-valued morphological operators are defined by enriching a complete lattice with additional binary operations.

Definition (Quantale (Mulvey, 1986))

A quantale $\mathcal{Q}$ is a complete lattice $\mathcal{Q}$ together with an associative binary operation "$\cdot$" which distributes over arbitrary suprema.

Non-flat vector-valued morphological operators are defined by enriching a complete lattice with additional binary operations.

Definition (Quantale (Mulvey, 1986))

A quantale $\mathcal{Q}$ is a complete lattice $\mathcal{Q}$ together with an associative binary operation "$\cdot$" which distributes over arbitrary suprema.

The multiplication of a quantale $\mathcal{Q}$ is always residuated. Thus, there exists a binary operation "$/$" such that

$$x \cdot y \leq z \qquad \Longleftrightarrow \qquad x \leq z/y. \tag{6}$$

Non-flat vector-valued morphological operators are defined by enriching a complete lattice with additional binary operations.

Definition (Quantale (Mulvey, 1986))

A quantale $\mathcal{Q}$ is a complete lattice $\mathcal{Q}$ together with an associative binary operation "$\cdot$" which distributes over arbitrary suprema.

The multiplication of a quantale $\mathcal{Q}$ is always residuated. Thus, there exists a binary operation "$/$" such that

$$x \cdot y \leq z \qquad \Longleftrightarrow \qquad x \leq z/y. \tag{6}$$

Using a structuring function $\boldsymbol{g} : \mathcal{D} \to \mathcal{Q}$, the non-flat vector-valued morphological operators are defined as follows (Stell, 2009):

$$[\varepsilon_{\boldsymbol{g}}(\boldsymbol{f})](x) = \inf_{s \in S} \{\boldsymbol{f}(x+s)/\boldsymbol{g}(s)\}, \tag{7}$$

and

$$[\delta_{\boldsymbol{g}}(\boldsymbol{f})](x) = \sup_{s \in S} \{\boldsymbol{f}(x-s) \cdot \boldsymbol{g}(s)\}. \tag{8}$$

Marginal Ordering

In RGB space, denoted by $\mathbb{C}_{RGB} = [0, 1] \times [0, 1] \times [0, 1]$, a color $c = (c_r, c_g, c_b) \in \mathbb{C}_{RGB}$ is written in terms of the primitives red, green, and blue.

Marginal Ordering

In RGB space, denoted by $\mathbb{C}_{RGB} = [0,1] \times [0,1] \times [0,1]$, a color $c = (c_r, c_g, c_b) \in \mathbb{C}_{RGB}$ is written in terms of the primitives red, green, and blue.

We can define the **marginal ordering** as follows

$$c \leq c' \quad \Longleftrightarrow \quad c_r \leq c'_r, \quad c_g \leq c'_g \quad \text{and} \quad c_b \leq c'_b.$$

Marginal Ordering

In RGB space, denoted by $\mathbb{C}_{RGB} = [0,1] \times [0,1] \times [0,1]$, a color $c = (c_r, c_g, c_b) \in \mathbb{C}_{RGB}$ is written in terms of the primitives red, green, and blue.

We can define the **marginal ordering** as follows

$$c \leq c' \iff c_r \leq c'_r, \quad c_g \leq c'_g \quad \text{and} \quad c_b \leq c'_b.$$

In this case, the infimum and supremum of a set of colors are determined component-wise.

For example, we have

$$\inf\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,0,0)}_{\text{red}}, \quad \sup\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,1,1)}_{\text{white}}.$$

For example, we have

$$\inf\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,0,0)}_{\text{red}}, \quad \sup\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,1,1)}_{\text{white}}.$$

The flat erosion and flat dilation of a color image with marginal order are determined as follows:

$$\varepsilon_S(\boldsymbol{f})(x) = \left(\inf_{s\in S}\{f_r(x+s)\}, \inf_{s\in S}\{f_g(x+s)\}, \inf_{s\in S}\{f_b(x+s)\} \right),$$

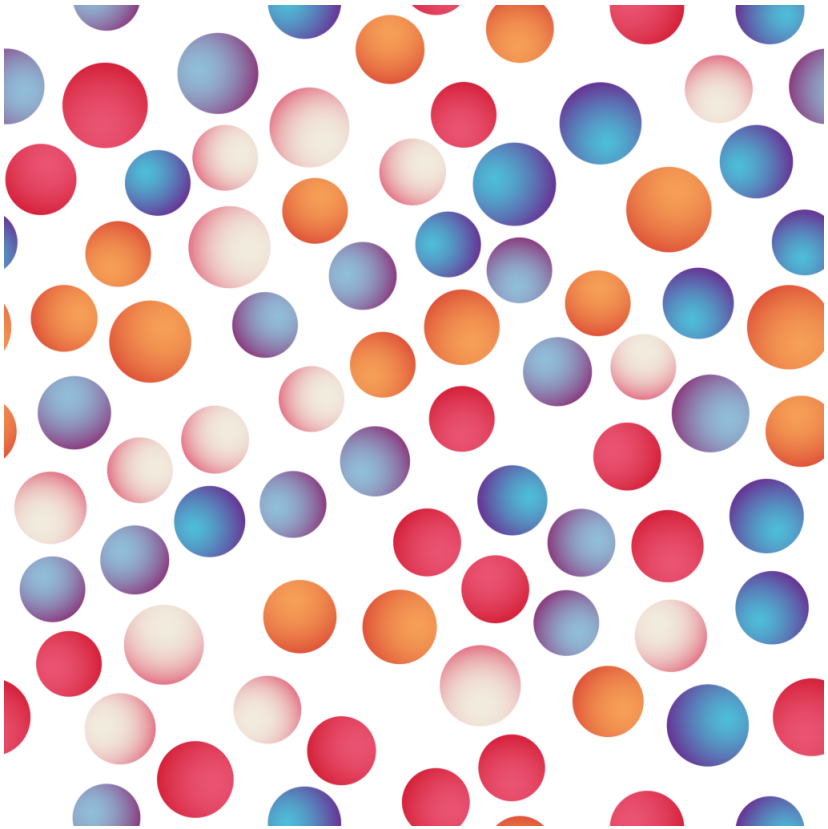
and

$$\delta_S(\boldsymbol{f})(x) = \left(\sup_{s\in S}\{f_r(x+s)\}, \sup_{s\in S}\{f_g(x+s)\}, \sup_{s\in S}\{f_b(x+s)\} \right),$$

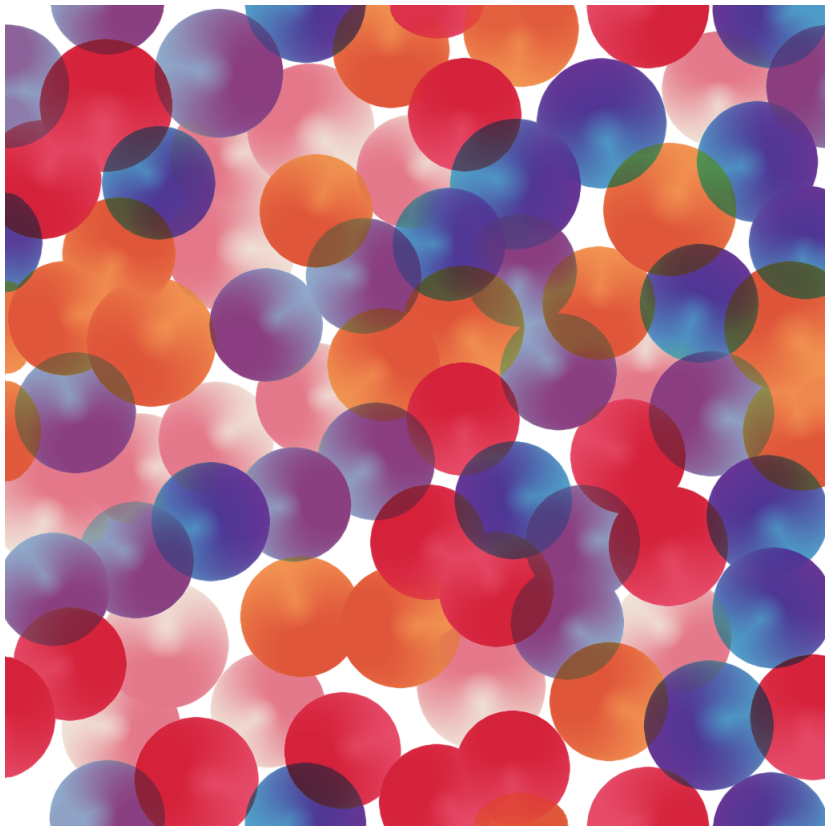
where $\boldsymbol{f}(x) = (f_r(x), f_g(x), f_b(x))$ for all $x \in \mathcal{D}$.

Example – Flat Marginal Erosion

Color image $\boldsymbol{f}$



Eroded image $\varepsilon_S(\boldsymbol{f})$



- The marginal approach may introduce “false colors”!
- It does not consider the correlation between color channels!

Lexicographical Ordering

The RGB color space can be endowed with the lexicographic ordering defined by

$$c \leq c' \quad \Longleftrightarrow \quad \begin{cases} c_r < c'_r, \\ c_r = c'_r \text{ e } c_g < c'_g, \\ c_r = c'_r,\ c_g = c'_g \text{ e } c_b \leq c'_b. \end{cases}$$

Lexicographical Ordering

The RGB color space can be endowed with the lexicographic ordering defined by

$$c \leq c' \quad \Longleftrightarrow \quad \begin{cases} c_r < c'_r, \\ c_r = c'_r \text{ e } c_g < c'_g, \\ c_r = c'_r,\ c_g = c'_g \text{ e } c_b \leq c'_b. \end{cases}$$

For example,

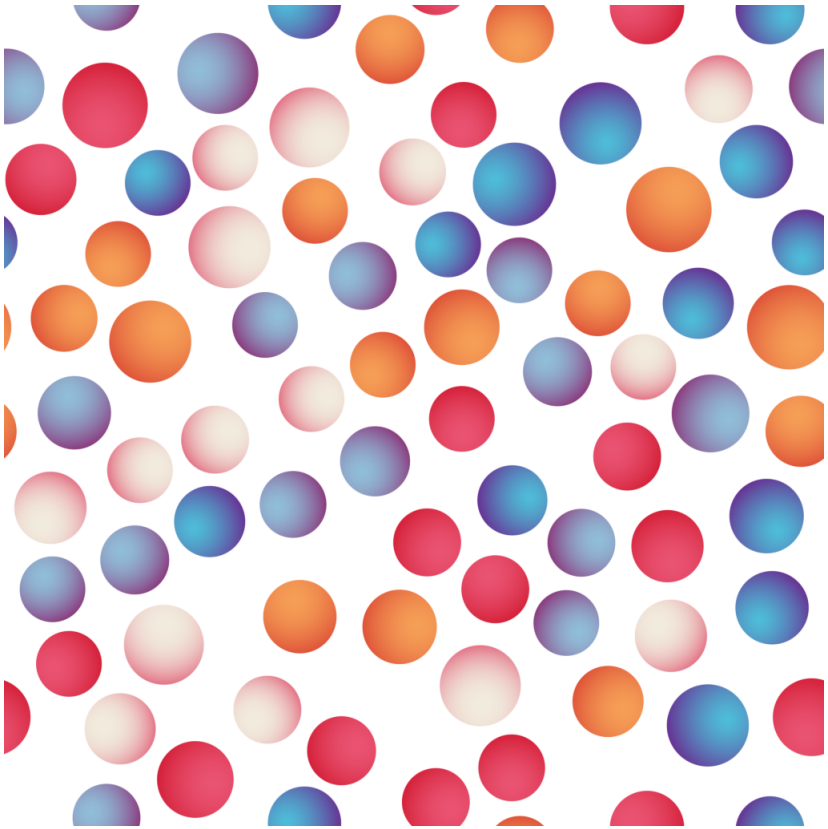
$$\inf\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,0,1)}_{\text{magenta}},$$

and

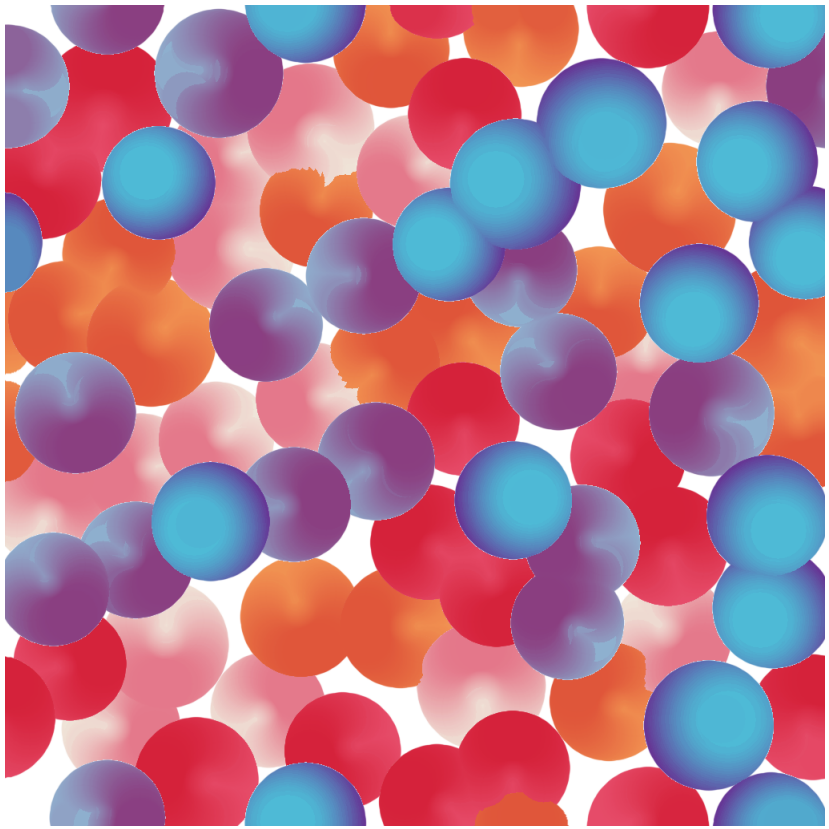
$$\sup\{\underbrace{(1,1,0)}_{\text{yellow}}, \underbrace{(1,0,1)}_{\text{magenta}}\} = \underbrace{(1,1,1)}_{\text{yellow}}.$$

Example – Flat Lexicographical Erosion

Color image $\boldsymbol{f}$



Eroded image $\varepsilon_S(\boldsymbol{f})$



The lexicographic order is a total order. Thus, we can always compare two colors.

The lexicographic order is a total order. Thus, we can always compare two colors.

In a total order, both the supremum and the infimum of a finite set are elements of the set.

The lexicographic order is a total order. Thus, we can always compare two colors.

In a total order, both the supremum and the infimum of a finite set are elements of the set.

A total order, including the lexicographic order, does not introduce false colors!

The lexicographic order is a total order. Thus, we can always compare two colors.

In a total order, both the supremum and the infimum of a finite set are elements of the set.

A total order, including the lexicographic order, does not introduce false colors!

There are many other approaches to vector-valued mathematical morphology besides the marginal and lexicographical approaches.

The lexicographic order is a total order. Thus, we can always compare two colors.

In a total order, both the supremum and the infimum of a finite set are elements of the set.

A total order, including the lexicographic order, does not introduce false colors!

There are many other approaches to vector-valued mathematical morphology besides the marginal and lexicographical approaches.

Examples include Loewner order (Burgeth and Kleefeld, 2014) and the approaches based on “color” quaternions (Angulo, 2010).

The lexicographic order is a total order. Thus, we can always compare two colors.

In a total order, both the supremum and the infimum of a finite set are elements of the set.

A total order, including the lexicographic order, does not introduce false colors!

There are many other approaches to vector-valued mathematical morphology besides the marginal and lexicographical approaches.

Examples include Loewner order (Burgeth and Kleefeld, 2014) and the approaches based on “color” quaternions (Angulo, 2010).

The following reviews an approach based on reduced ordering combined with look-up tables (Goutsias et al., 1995).

Reduced Ordering

A reduced ordering is defined as follows using a surjective mapping $\rho : \mathbb{V} \to \mathbb{L}$ that maps the value set to a complete lattice:

$$\boldsymbol{u} \leq_\rho \boldsymbol{v} \quad \Longleftrightarrow \quad \rho(\boldsymbol{u}) \leq_{\mathbb{L}} \rho(\boldsymbol{v}). \tag{9}$$

Reduced Ordering

A reduced ordering is defined as follows using a surjective mapping $\rho : \mathbb{V} \to \mathbb{L}$ that maps the value set to a complete lattice:

$$\boldsymbol{u} \leq_\rho \boldsymbol{v} \quad \Longleftrightarrow \quad \rho(\boldsymbol{u}) \leq_{\mathbb{L}} \rho(\boldsymbol{v}). \tag{9}$$

A reduced ordering is not a partial order but a preorder because $\boldsymbol{u} \leq_\rho \boldsymbol{v}$ and $\boldsymbol{v} \leq_\rho \boldsymbol{u}$ do not necessarily imply $\boldsymbol{u} = \boldsymbol{v}$.

Reduced Ordering

A reduced ordering is defined as follows using a surjective mapping $\rho : \mathbb{V} \to \mathbb{L}$ that maps the value set to a complete lattice:

$$\boldsymbol{u} \leq_\rho \boldsymbol{v} \quad \Longleftrightarrow \quad \rho(\boldsymbol{u}) \leq_{\mathbb{L}} \rho(\boldsymbol{v}). \tag{9}$$

A reduced ordering is not a partial order but a preorder because $\boldsymbol{u} \leq_\rho \boldsymbol{v}$ and $\boldsymbol{v} \leq_\rho \boldsymbol{u}$ do not necessarily imply $\boldsymbol{u} = \boldsymbol{v}$.

Nevertheless, morphological operators can be defined using reduced orderings as shown in the next slide (Goutsias et al., 1995):

Definition (ρ-Increasing Morphological Operators)

Let $\mathbb{V}$ be non-empty set, $\mathbb{L}$ a complete lattice, and $\rho : \mathbb{V} \to \mathbb{L}$ a surjective mappings. A mapping $\psi^\rho : \mathbb{V} \to \mathbb{V}$ is an ρ-increasing morphological operator if there exists an increasing morphological operator (e.g., dilation, erosion, opening, and closing) $\psi : \mathbb{L} \to \mathbb{L}$ such that

$$\rho(\psi^\rho(x)) = \psi(\rho(x)), \quad \forall x \in \mathbb{V}. \tag{10}$$

Definition (ρ-Increasing Morphological Operators)

Let $\mathbb{V}$ be non-empty set, $\mathbb{L}$ a complete lattice, and $\rho : \mathbb{V} \to \mathbb{L}$ a surjective mappings. A mapping $\psi^\rho : \mathbb{V} \to \mathbb{V}$ is an ρ-increasing morphological operator if there exists an increasing morphological operator (e.g., dilation, erosion, opening, and closing) $\psi : \mathbb{L} \to \mathbb{L}$ such that

$$\rho(\psi^\rho(x)) = \psi(\rho(x)), \quad \forall x \in \mathbb{V}. \tag{10}$$

For example, an operator $\varepsilon^\rho : \mathbb{V} \to \mathbb{V}$ is a ρ-increasing erosion, or simply a reduced erosion, if there exists an erosion $\varepsilon : \mathbb{L} \to \mathbb{L}$ such that $\rho \circ \varepsilon^\rho = \varepsilon \circ \rho$.

In practice, ρ-increasing morphological operators can be efficiently computed using look-up tables (LUT) and gray-scale morphological operators (Velasco-Forero and Angulo, 2014).

In practice, ρ-increasing morphological operators can be efficiently computed using look-up tables (LUT) and gray-scale morphological operators (Velasco-Forero and Angulo, 2014).

Briefly, each value is indexed by a scalar sorted using the mapping ρ. A morphological operator is applied to the indexed image, and the vector-valued image output is retrieved by replacing the indexes with the corresponding values.

In practice, ρ-increasing morphological operators can be efficiently computed using look-up tables (LUT) and gray-scale morphological operators (Velasco-Forero and Angulo, 2014).

Briefly, each value is indexed by a scalar sorted using the mapping ρ. A morphological operator is applied to the indexed image, and the vector-valued image output is retrieved by replacing the indexes with the corresponding values.

Interestingly, machine learning techniques can be used to determine the subjective mapping ρ (Velasco-Forero and Angulo, 2014).

In practice, ρ-increasing morphological operators can be efficiently computed using look-up tables (LUT) and gray-scale morphological operators (Velasco-Forero and Angulo, 2014).

Briefly, each value is indexed by a scalar sorted using the mapping ρ. A morphological operator is applied to the indexed image, and the vector-valued image output is retrieved by replacing the indexes with the corresponding values.

Interestingly, machine learning techniques can be used to determine the subjective mapping ρ (Velasco-Forero and Angulo, 2014).

> Unlike the marginal approach, vectors are treated as single entities in the reduced ordering approach.

Vector-Valued Neural Networks

Despite the successful applications of deep learning to multidimensional signal and image processing, most traditional neural networks process data represented by (multidimensional) arrays of real numbers.

Vector-Valued Neural Networks

Despite the successful applications of deep learning to multidimensional signal and image processing, most traditional neural networks process data represented by (multidimensional) arrays of real numbers.

The intercorrelation between feature channels is typically expected to be learned from the training data, requiring numerous parameters and careful training.

Vector-Valued Neural Networks

Despite the successful applications of deep learning to multidimensional signal and image processing, most traditional neural networks process data represented by (multidimensional) arrays of real numbers.

The intercorrelation between feature channels is typically expected to be learned from the training data, requiring numerous parameters and careful training.

Vector-valued neural networks (V-nets) are designed to process arrays of vectors and consider the intercorrelation between feature channels beforehand. (Fan et al., 2020; Valle, 2024).

Vector-Valued Neural Networks

Despite the successful applications of deep learning to multidimensional signal and image processing, most traditional neural networks process data represented by (multidimensional) arrays of real numbers.

The intercorrelation between feature channels is typically expected to be learned from the training data, requiring numerous parameters and careful training.

Vector-valued neural networks (V-nets) are designed to process arrays of vectors and consider the intercorrelation between feature channels beforehand. (Fan et al., 2020; Valle, 2024).

V-nets typically have fewer parameters and usually undergo more robust training than traditional networks.

V-nets encompass hypercomplex-valued neural networks that are obtained by considering hypercomplex algebras such as Clifford or Cayley-Dickson algebras (Vieira and Valle, 2022a).

V-nets encompass hypercomplex-valued neural networks that are obtained by considering hypercomplex algebras such as Clifford or Cayley-Dickson algebras (Vieira and Valle, 2022a).

Hypercomplex-valued neural networks can benefit from the geometric aspects of the hypercomplex algebra, resulting in rotation- or translation-invariant or -equivariant models (Ruhe et al., 2023a; Vieira et al., 2023)

V-nets encompass hypercomplex-valued neural networks that are obtained by considering hypercomplex algebras such as Clifford or Cayley-Dickson algebras (Vieira and Valle, 2022a).

Hypercomplex-valued neural networks can benefit from the geometric aspects of the hypercomplex algebra, resulting in rotation- or translation-invariant or -equivariant models (Ruhe et al., 2023a; Vieira et al., 2023)

However, research on hypercomplex-valued neural networks is predominantly based on complex numbers and quaternions.

V-nets encompass hypercomplex-valued neural networks that are obtained by considering hypercomplex algebras such as Clifford or Cayley-Dickson algebras (Vieira and Valle, 2022a).

Hypercomplex-valued neural networks can benefit from the geometric aspects of the hypercomplex algebra, resulting in rotation- or translation-invariant or -equivariant models (Ruhe et al., 2023a; Vieira et al., 2023)

However, research on hypercomplex-valued neural networks is predominantly based on complex numbers and quaternions.

In the following, we consider a broad framework encompassing hypercomplex-valued models as particular instances.

V-nets encompass hypercomplex-valued neural networks that are obtained by considering hypercomplex algebras such as Clifford or Cayley-Dickson algebras (Vieira and Valle, 2022a).

Hypercomplex-valued neural networks can benefit from the geometric aspects of the hypercomplex algebra, resulting in rotation- or translation-invariant or -equivariant models (Ruhe et al., 2023a; Vieira et al., 2023)

However, research on hypercomplex-valued neural networks is predominantly based on complex numbers and quaternions.

In the following, we consider a broad framework encompassing hypercomplex-valued models as particular instances.

Furthermore, we point out how to implement vector-valued neural networks in current deep learning libraries.

Algebras and Hypercomplex Algebras

Many image and signal processing tasks - such as those related to multivariate images and 3D audio signals (Grassucci et al., 2023; Miron et al., 2023; Parcollet et al., 2020) - deal with vector-valued data.

Algebras and Hypercomplex Algebras

Many image and signal processing tasks - such as those related to multivariate images and 3D audio signals (Grassucci et al., 2023; Miron et al., 2023; Parcollet et al., 2020) - deal with vector-valued data.

Since addition and multiplication are essential operations for the design of neural networks, let us recall the concept of algebra.

Algebras and Hypercomplex Algebras

Many image and signal processing tasks - such as those related to multivariate images and 3D audio signals (Grassucci et al., 2023; Miron et al., 2023; Parcollet et al., 2020) - deal with vector-valued data.

Since addition and multiplication are essential operations for the design of neural networks, let us recall the concept of algebra.

Definition (Algebra (Schafer, 1961))

An algebra $\mathbb{V}$ is a vector space over a field $\mathbb{F}$ equipped with a bilinear operation called multiplication or product.

In this talk, we will focus on algebras over the field of real numbers; that is, we will consider only $\mathbb{F} = \mathbb{R}$.

Finite-Dimensional Algebras

We will be concerned only with finite-dimensional vector spaces. We assume that $\mathbb{V}$ is a vector space of dimension n, that is, $dim(\mathbb{V}) = n$.

Finite-Dimensional Algebras

We will be concerned only with finite-dimensional vector spaces. We assume that $\mathbb{V}$ is a vector space of dimension n, that is, $dim(\mathbb{V}) = n$.

Let $\mathcal{E} = \{e_0, e_1, \ldots, e_{n-1}\}$ be an ordered basis for $\mathbb{V}$. Given $x \in \mathbb{V}$, there exists a unique tuple $(\xi_0, \xi_1, \ldots, \xi_{n-1}) \in \mathbb{R}^n$ such that

$$x = \sum_{i=0}^{n-1} \xi_i e_i. \tag{11}$$

Finite-Dimensional Algebras

We will be concerned only with finite-dimensional vector spaces. We assume that $\mathbb{V}$ is a vector space of dimension n, that is, $dim(\mathbb{V}) = n$.

Let $\mathcal{E} = \{e_0, e_1, \ldots, e_{n-1}\}$ be an ordered basis for $\mathbb{V}$. Given $x \in \mathbb{V}$, there exists a unique tuple $(\xi_0, \xi_1, \ldots, \xi_{n-1}) \in \mathbb{R}^n$ such that

$$x = \sum_{i=0}^{n-1} \xi_i e_i. \tag{11}$$

We will denote by $\varphi : \mathbb{V} \to \mathbb{R}^n$ the isomorphism (which depends on the ordered basis) given by the equation $\varphi(x) = [\xi_0, \ldots, \xi_{n-1}]^T$.

Finite-Dimensional Algebras

We will be concerned only with finite-dimensional vector spaces. We assume that $\mathbb{V}$ is a vector space of dimension n, that is, $dim(\mathbb{V}) = n$.

Let $\mathcal{E} = \{e_0, e_1, \ldots, e_{n-1}\}$ be an ordered basis for $\mathbb{V}$. Given $x \in \mathbb{V}$, there exists a unique tuple $(\xi_0, \xi_1, \ldots, \xi_{n-1}) \in \mathbb{R}^n$ such that

$$x = \sum_{i=0}^{n-1} \xi_i e_i. \tag{11}$$

We will denote by $\varphi : \mathbb{V} \to \mathbb{R}^n$ the isomorphism (which depends on the ordered basis) given by the equation $\varphi(x) = [\xi_0, \ldots, \xi_{n-1}]^T$.

In computational applications, $x \in \mathbb{V}$ is given by its coordinates relative to the ordered basis $\mathcal{E}$.

Multiplication Table

Given an ordered basis $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$ of $\mathbb{V}$, multiplication is completely determined by the n^3 parameters $p_{ijk} \in \mathbb{R}$ that appear in the products

$$e_i e_j = \sum_{k=0}^{n-1} p_{ijk} e_k, \quad \forall i, j = 0, 1, \ldots, n-1. \tag{12}$$

Multiplication Table

Given an ordered basis $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$ of $\mathbb{V}$, multiplication is completely determined by the n^3 parameters $p_{ijk} \in \mathbb{R}$ that appear in the products

$$e_i e_j = \sum_{k=0}^{n-1} p_{ijk} e_k, \quad \forall i, j = 0, 1, \ldots, n-1. \tag{12}$$

The products in (12) can be organized into a multiplication table:

	e_0	$\cdots$	e_j	$\cdots$	e_{n-1}
			$\vdots$		
e_i	$\cdots$		$\sum_{k=0}^{n-1} p_{ijk} e_k$		$\cdots$
			$\vdots$		

Hypercomplex Algebra

A hypercomplex algebra, denoted by $\mathbb{H}$, is a finite-dimensional algebra in which the product has an identity bilateral (Catoni et al., 2008; Kantor and Solodovnikov, 1989).

Hypercomplex Algebra

A hypercomplex algebra, denoted by $\mathbb{H}$, is a finite-dimensional algebra in which the product has an identity bilateral (Catoni et al., 2008; Kantor and Solodovnikov, 1989).

A hypercomplex algebra $\mathbb{H}$ is equipped with a (unique) element e_0 such that $xe_0 = e_0x = x$ for all $x \in \mathbb{V}$.

Hypercomplex Algebra

A hypercomplex algebra, denoted by $\mathbb{H}$, is a finite-dimensional algebra in which the product has an identity bilateral (Catoni et al., 2008; Kantor and Solodovnikov, 1989).

A hypercomplex algebra $\mathbb{H}$ is equipped with a (unique) element e_0 such that $xe_0 = e_0x = x$ for all $x \in \mathbb{V}$.

The identity is usually the first element of the ordered basis. Thus, $\mathcal{E} = \{e_0, e_1, \ldots, e_{n-1}\}$ is an ordered basis of a hypercomplex algebra.

Hypercomplex Algebra

A hypercomplex algebra, denoted by $\mathbb{H}$, is a finite-dimensional algebra in which the product has an identity bilateral (Catoni et al., 2008; Kantor and Solodovnikov, 1989).

A hypercomplex algebra $\mathbb{H}$ is equipped with a (unique) element e_0 such that $xe_0 = e_0x = x$ for all $x \in \mathbb{V}$.

The identity is usually the first element of the ordered basis. Thus, $\mathcal{E} = \{e_0, e_1, \ldots, e_{n-1}\}$ is an ordered basis of a hypercomplex algebra.

We often consider the canonical basis $\tau = \{1, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_{n-1}\}$. Thus, a hypercomplex number is given by

$$x = x_0 + x_1\boldsymbol{i}_1 + \ldots + x_{n-1}\boldsymbol{i}_{n-1}. \tag{13}$$

Example – Quaternions

Quaternions, introduced by Hamilton in the late 19th century, are hypercomplex numbers that generalize complex numbers.

Example – Quaternions

Quaternions, introduced by Hamilton in the late 19th century, are hypercomplex numbers that generalize complex numbers.

Using the canonical basis $\tau = \{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$, a quaternion is given by $x = x_0 + x_1\boldsymbol{i} + x_2\boldsymbol{j} + x_3\boldsymbol{k}$ and the quaternion multiplication table is

	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\boldsymbol{i}$	$\boldsymbol{i}$	-1	$\boldsymbol{k}$	$-\boldsymbol{j}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{k}$	-1	$\boldsymbol{i}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	-1

Example – Quaternions

Quaternions, introduced by Hamilton in the late 19th century, are hypercomplex numbers that generalize complex numbers.

Using the canonical basis $\tau = \{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$, a quaternion is given by $x = x_0 + x_1\boldsymbol{i} + x_2\boldsymbol{j} + x_3\boldsymbol{k}$ and the quaternion multiplication table is

	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\boldsymbol{i}$	$\boldsymbol{i}$	-1	$\boldsymbol{k}$	$-\boldsymbol{j}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{k}$	-1	$\boldsymbol{i}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	-1

Quaternions are adequate for describing rotations in 3D space.

Example – Quaternions

Quaternions, introduced by Hamilton in the late 19th century, are hypercomplex numbers that generalize complex numbers.

Using the canonical basis $\tau = \{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$, a quaternion is given by $x = x_0 + x_1\boldsymbol{i} + x_2\boldsymbol{j} + x_3\boldsymbol{k}$ and the quaternion multiplication table is

	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\boldsymbol{i}$	$\boldsymbol{i}$	-1	$\boldsymbol{k}$	$-\boldsymbol{j}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{k}$	-1	$\boldsymbol{i}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	-1

Quaternions are adequate for describing rotations in 3D space.

They have also been used effectively to develop neural networks (Arena et al., 1997; Parcollet et al., 2020).

Matrix Representation of the Product

The left multiplication by $a = \sum_{i=0}^{n-1} \alpha_i e_i$ yields a linear operator $\mathcal{A}_L : \mathbb{V} \to \mathbb{V}$ defined by $\mathcal{A}_L(x) = ax$, for all $x \in \mathbb{V}$.

Matrix Representation of the Product

The left multiplication by $a = \sum_{i=0}^{n-1} \alpha_i e_i$ yields a linear operator $\mathcal{A}_L : \mathbb{V} \to \mathbb{V}$ defined by $\mathcal{A}_L(x) = ax$, for all $x \in \mathbb{V}$.

The matrix representation of $\mathcal{A}_L$ with respect to an ordered basis $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$ yields a mapping $\mathcal{M}_L : \mathbb{V} \to \mathbb{R}^{n \times n}$ given by

$$\mathcal{M}_L(a) = \begin{bmatrix} | & & | \\ \varphi(ae_0) & \ldots & \varphi(ae_{n-1}) \\ | & & | \end{bmatrix}.$$

Effectively, we can write

$$\mathcal{M}_L(a) = \sum_{i=0}^{n-1} \alpha_i P_{i:}^T, \qquad P_{i:}^T = \begin{bmatrix} p_{i00} & p_{i10} & \cdots & p_{i(n-1)0} \\ p_{i01} & p_{i11} & \cdots & p_{i(n-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i0(n-1)} & p_{i1(n-1)} & \cdots & p_{i(n-1)(n-1)} \end{bmatrix}. \tag{14}$$

Effectively, we can write

$$\mathcal{M}_L(a) = \sum_{i=0}^{n-1} \alpha_i P_{i:}^T, \qquad P_{i:}^T = \begin{bmatrix} p_{i00} & p_{i10} & \cdots & p_{i(n-1)0} \\ p_{i01} & p_{i11} & \cdots & p_{i(n-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i0(n-1)} & p_{i1(n-1)} & \cdots & p_{i(n-1)(n-1)} \end{bmatrix}. \tag{14}$$

Using matrix representation, we have

$$\varphi(ax) = \mathcal{M}_L(a)\varphi(x) = \sum_{i=0}^{n-1} \alpha_i P_{i:}^T \varphi(x). \tag{15}$$

Effectively, we can write

$$\mathcal{M}_L(a) = \sum_{i=0}^{n-1} \alpha_i P_{i:}^T, \qquad P_{i:}^T = \begin{bmatrix} p_{i00} & p_{i10} & \cdots & p_{i(n-1)0} \\ p_{i01} & p_{i11} & \cdots & p_{i(n-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i0(n-1)} & p_{i1(n-1)} & \cdots & p_{i(n-1)(n-1)} \end{bmatrix}. \tag{14}$$

Using matrix representation, we have

$$\varphi(ax) = \mathcal{M}_L(a)\varphi(x) = \sum_{i=0}^{n-1} \alpha_i P_{i:}^T \varphi(x). \tag{15}$$

Concluding, we can compute the product of a and x as follows:

$$ax = \varphi^{-1}\left(\sum_{i=0}^{n-1} \alpha_i P_{i:}^T \varphi(x)\right) \tag{16}$$

Example – Quaternions

Consider the quaternions with the canonical basis $\tau = \{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$.
The product of $x = 1 + 2\boldsymbol{i} + 3\boldsymbol{j} + 4\boldsymbol{k}$ and $y = 5 + 6\boldsymbol{i} + 7\boldsymbol{j} + 8\boldsymbol{k}$ satisfies

$$\varphi(xy) = \mathcal{M}_L(x)\varphi(y) = \begin{bmatrix} 1 & -2 & -3 & -4 \\ 2 & 1 & -4 & 3 \\ 3 & 4 & 1 & -2 \\ 4 & -3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} -60 \\ 12 \\ 30 \\ 24 \end{bmatrix}.$$

Thus, $xy = -60 + 12\boldsymbol{i} + 30\boldsymbol{j} + 24\boldsymbol{k}$.

Example – Quaternions

Consider the quaternions with the canonical basis $\tau = \{1, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$.
The product of $x = 1 + 2\boldsymbol{i} + 3\boldsymbol{j} + 4\boldsymbol{k}$ and $y = 5 + 6\boldsymbol{i} + 7\boldsymbol{j} + 8\boldsymbol{k}$ satisfies

$$\varphi(xy) = \mathcal{M}_L(x)\varphi(y) = \begin{bmatrix} 1 & -2 & -3 & -4 \\ 2 & 1 & -4 & 3 \\ 3 & 4 & 1 & -2 \\ 4 & -3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} -60 \\ 12 \\ 30 \\ 24 \end{bmatrix}.$$

Thus, $xy = -60 + 12\boldsymbol{i} + 30\boldsymbol{j} + 24\boldsymbol{k}$.

Note that

$$\mathcal{M}_L(x) = 1P_{0:} + 2P_{1:} + 3P_{2:} + 4P_{n:},$$

where $P_{0:} = \mathbf{I}_{4\times 4}$ is the identity matrix and

$$P_{1:}^T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, P_{2:}^T = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, P_{3:}^T = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Computing with Vector-Valued Matrices

Matrix computation is a key concept for developing V-nets because some fundamental building blocks, such as dense and convolutional layers, compute affine transformations followed by a nonlinear function.

Computing with Vector-Valued Matrices

Matrix computation is a key concept for developing V-nets because some fundamental building blocks, such as dense and convolutional layers, compute affine transformations followed by a nonlinear function.

Let $\mathbb{V}$ be an algebra over the real numbers. The product of two vector-valued matrices $\boldsymbol{A} \in \mathbb{V}^{M \times L}$ and $\boldsymbol{B} \in \mathbb{V}^{L \times N}$ results in a new matrix $\boldsymbol{C} \in \mathbb{V}^{M \times N}$ with entries defined by

$$c_{ij} = \sum_{\ell=1}^{L} a_{i\ell} b_{\ell j}, \quad \forall i = 1, \ldots, M \quad \text{and} \quad j = 1, \ldots, N. \tag{17}$$

We compute the above operation using real-valued matrix operations to take advantage of scientific computing software.

We compute the above operation using real-valued matrix operations to take advantage of scientific computing software.

Let $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$ be a basis for $\mathbb{V}$. Using the isomorphism $\varphi : \mathbb{V} \to \mathbb{R}^n$ defined by

$$\varphi(x) = \begin{bmatrix} \xi_0 \\ \vdots \\ \xi_{n-1} \end{bmatrix}, \quad \forall x = \sum_{i=0}^{n-1} \xi_i e_i \in \mathbb{V},$$

we have

$$\varphi(c_{ij}) = \varphi\left(\sum_{\ell=1}^{L} a_{i\ell} b_{\ell j}\right) = \sum_{\ell=1}^{L} \varphi\left(a_{i\ell} b_{\ell j}\right) = \sum_{\ell=1}^{L} \mathcal{M}_L(a_{i\ell}) \varphi(b_{\ell j}),$$

where $\mathcal{M}_L : \mathbb{V} \to \mathbb{R}^{n \times n}$ is the matrix representation of the left multiplication by $a = \sum_{i=0}^{n-1} \alpha_i e_i$.

Equivalently, using real-valued matrix operations, we have

$$\varphi(\boldsymbol{C}) = \mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B}), \tag{18}$$

where $\mathcal{M}_L$ and φ are given by

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} \mathcal{M}_L(a_{11}) & \mathcal{M}_L(a_{12}) & \dots & \mathcal{M}_L(a_{1L}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_L(a_{M1}) & \mathcal{M}_L(a_{M2}) & \dots & \mathcal{M}_L(a_{ML}) \end{bmatrix} \in \mathbb{R}^{nM \times nL}, \tag{19}$$

e

$$\varphi(\boldsymbol{B}) = \begin{bmatrix} \varphi(b_{11}) & \dots & \varphi(b_{1N}) \\ \varphi(b_{21}) & \dots & \varphi(b_{2N}) \\ \vdots & \ddots & \vdots \\ \varphi(b_{L1}) & ldots & \varphi(b_{LN}) \end{bmatrix} \in \mathbb{R}^{nL \times N}. \tag{20}$$

Rearranging the elements $\varphi(\boldsymbol{C})$, we have

$$\boldsymbol{C} = \varphi^{-1}\left(\mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B})\right), \tag{21}$$

which allows the computation of vector-valued matrix products using the real-valued matrix computation often available in scientific computing software.

Rearranging the elements $\varphi(\boldsymbol{C})$, we have

$$\boldsymbol{C} = \varphi^{-1}\left(\mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B})\right), \tag{21}$$

which allows the computation of vector-valued matrix products using the real-valued matrix computation often available in scientific computing software.

To further reduce the computational effort, the real-valued matrix $\mathcal{M}_L(\boldsymbol{A}) \in \mathbb{R}^{nM \times nL}$ can be computed using the Kronecker product.

Kronecker product

The Kronecker product of two real matrices $A = (a_{ij}) \in \mathbb{R}^{N \times M}$ and $B \in \mathbb{R}^{P \times Q}$, denoted by $A \otimes B$, yields the block matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1M}B \\ a_{21}B & a_{22}B & \dots & a_{2M}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}B & a_{N2}B & \dots & a_{NM}B \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \quad (22)$$

Kronecker product

The Kronecker product of two real matrices $A = (a_{ij}) \in \mathbb{R}^{N \times M}$ and $B \in \mathbb{R}^{P \times Q}$, denoted by $A \otimes B$, yields the block matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1M}B \\ a_{21}B & a_{22}B & \dots & a_{2M}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}B & a_{N2}B & \dots & a_{NM}B \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \quad (22)$$

For example, Stenger (1968) and Loan (2000) describe basic properties and some applications of the Kronecker product.

Kronecker product

The Kronecker product of two real matrices $A = (a_{ij}) \in \mathbb{R}^{N \times M}$ and $B \in \mathbb{R}^{P \times Q}$, denoted by $A \otimes B$, yields the block matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1M}B \\ a_{21}B & a_{22}B & \dots & a_{2M}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}B & a_{N2}B & \dots & a_{NM}B \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \quad (22)$$

For example, Stenger (1968) and Loan (2000) describe basic properties and some applications of the Kronecker product.

Following Zhang et al. (2021) and Grassucci et al. (2022), we use the Kronecker product to compute $\mathcal{M}_L(\boldsymbol{A})$ as follows.

The matrix representation of left multiplication by $a_{ij} = \sum_{k=0}^{n-1} \alpha_{ijk} e_k$ satisfies

$$\mathcal{M}_L(a_{ij}) = \sum_{k=0}^{n-1} \alpha_{ijk} P_{k:}^T, \quad P_{k:}^T = \begin{bmatrix} p_{k00} & p_{k10} & \cdots & p_{k(n-1)0} \\ p_{k01} & p_{k11} & \cdots & p_{k(n-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k0(n-1)} & p_{k1(n-1)} & \cdots & p_{k(n-1)(n-1)} \end{bmatrix}.$$

The matrix representation of left multiplication by $a_{ij} = \sum_{k=0}^{n-1} \alpha_{ijk} e_k$ satisfies

$$\mathcal{M}_L(a_{ij}) = \sum_{k=0}^{n-1} \alpha_{ijk} P_{k:}^T, \quad P_{k:}^T = \begin{bmatrix} p_{k00} & p_{k10} & \cdots & p_{k(n-1)0} \\ p_{k01} & p_{k11} & \cdots & p_{k(n-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k0(n-1)} & p_{k1(n-1)} & \cdots & p_{k(n-1)(n-1)} \end{bmatrix}.$$

Therefore, we have

$$\mathcal{M}_L(\boldsymbol{A}) = \sum_{k=0}^{n-1} \begin{bmatrix} \alpha_{11k} P_{k:}^T & \alpha_{12k} P_{k:}^T & \cdots & \alpha_{iLk} P_{k:}^T \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1k} P_{k:}^T & \alpha_{M2k} P_{k:}^T & \cdots & \alpha_{MLk} P_{k:}^T \end{bmatrix}.$$

Let $A_k \in \mathbb{R}^{M \times L}$, for $k = 1, \ldots, n$, be real matrices such that

$$\boldsymbol{A} = \sum_{k=0}^{n-1} A_k e_k,$$

Let $A_k \in \mathbb{R}^{M \times L}$, for $k = 1, \ldots, n$, be real matrices such that

$$\boldsymbol{A} = \sum_{k=0}^{n-1} A_k e_k,$$

that is, A_k is the "matrix" component associated with the base element e_k of $\boldsymbol{A}$.

Let $A_k \in \mathbb{R}^{M \times L}$, for $k = 1, \ldots, n$, be real matrices such that

$$\boldsymbol{A} = \sum_{k=0}^{n-1} A_k e_k,$$

that is, A_k is the "matrix" component associated with the base element e_k of $\boldsymbol{A}$.

Using $A_k \in \mathbb{R}^{M \times L}$, we conclude that

$$\mathcal{M}_L(\boldsymbol{A}) = \sum_{k=0}^{n-1} A_k \otimes P_{k:}^T. \tag{23}$$

Let $A_k \in \mathbb{R}^{M \times L}$, for $k = 1, \ldots, n$, be real matrices such that

$$\boldsymbol{A} = \sum_{k=0}^{n-1} A_k e_k,$$

that is, A_k is the "matrix" component associated with the base element e_k of $\boldsymbol{A}$.

Using $A_k \in \mathbb{R}^{M \times L}$, we conclude that

$$\mathcal{M}_L(\boldsymbol{A}) = \sum_{k=0}^{n-1} A_k \otimes P_{k:}^T. \tag{23}$$

Therefore, $\boldsymbol{C} = \boldsymbol{AB}$ can be efficiently calculated by the equation

$$\boldsymbol{C} = \varphi^{-1}\left(\left(\sum_{k=0}^{n-1} A_k \otimes P_{k:}^T\right)\varphi(\boldsymbol{B})\right). \tag{24}$$

Example - Quaternions

Consider the matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 + 2\boldsymbol{i} & 3\boldsymbol{i} + 4\boldsymbol{j} & 5\boldsymbol{j} + 6\boldsymbol{k} \\ 7 + 8\boldsymbol{j} & 9 + 10\boldsymbol{k} & 11\, \boldsymbol{ii} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{2\times 3},$$

and the column vector

$$\boldsymbol{x} = \begin{bmatrix} 1 + 2\boldsymbol{i} + 3\boldsymbol{j} + 4\boldsymbol{k} \\ 5 + 6\boldsymbol{i} + 7\boldsymbol{j} + 8\boldsymbol{k} \\ 9 + 10\boldsymbol{i} + 11\boldsymbol{j} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{3\times 1}.$$

Example - Quaternions

Consider the matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 + 2\boldsymbol{i} & 3\boldsymbol{i} + 4\boldsymbol{j} & 5\boldsymbol{j} + 6\boldsymbol{k} \\ 7 + 8\boldsymbol{j} & 9 + 10\boldsymbol{k} & 11\, \boldsymbol{ii} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{2\times 3},$$

and the column vector

$$\boldsymbol{x} = \begin{bmatrix} 1 + 2\boldsymbol{i} + 3\boldsymbol{j} + 4\boldsymbol{k} \\ 5 + 6\boldsymbol{i} + 7\boldsymbol{j} + 8\boldsymbol{k} \\ 9 + 10\boldsymbol{i} + 11\boldsymbol{j} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{3\times 1}.$$

Using quaternion matrix algebra, we obtain

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} -176 + 45\boldsymbol{i} + 96\boldsymbol{j} + 11\boldsymbol{k} \\ -306 - 3\boldsymbol{i} + 140\boldsymbol{j} + 363\boldsymbol{k} \end{bmatrix}.$$

Using left multiplication, we compute

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} 1 & 0 & 0 \\ 7 & 9 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \cdots + A_3 \otimes P_{3:}^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \end{bmatrix} + \cdots + A_3 \otimes P_{3:}^T$$

$$= \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix}$$

Thus, we have

$$\varphi(\boldsymbol{y}) = \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

$$= \begin{bmatrix} -176 & 45 & 96 & 11 & -306 & -3 & 140 & 363 \end{bmatrix}^T.$$

Thus, we have

$$\varphi(\boldsymbol{y}) = \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

$$= \begin{bmatrix} -176 & 45 & 96 & 11 & -306 & -3 & 140 & 363 \end{bmatrix}^T.$$

Recall that

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} -176 + 45\boldsymbol{i} + 96\boldsymbol{j} + 11\boldsymbol{k} \\ -306 - 3\boldsymbol{i} + 140\boldsymbol{j} + 363\boldsymbol{k} \end{bmatrix}.$$

Vector-Valued Neural Networks (V-Nets)

Dense layers, also known as fully connected layers, are the building blocks of several neural network architectures (Géron, 2019).

Vector-Valued Neural Networks (V-Nets)

Dense layers, also known as fully connected layers, are the building blocks of several neural network architectures (Géron, 2019).

In particular, the famous multilayer perceptron (MLP) network is given by the composition of a sequence of dense layers.

Vector-Valued Neural Networks (V-Nets)

Dense layers, also known as fully connected layers, are the building blocks of several neural network architectures (Géron, 2019).

In particular, the famous multilayer perceptron (MLP) network is given by the composition of a sequence of dense layers.

The output $\boldsymbol{y} = (y_1, \ldots, y_M) \in \mathbb{V}^M$ of a dense layer with vector-valued neurons M in parallel is

$$\boldsymbol{y} = \psi(\boldsymbol{s} + \boldsymbol{b}) \quad \text{with} \quad \boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}, \tag{25}$$

where $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{V}^N$ is the input vector, $\boldsymbol{W} = (w_{ij}) \in \mathbb{V}^{M \times N}$ is the matrix containing the synaptic weights, $\boldsymbol{b} = (b_1, \ldots, b_M) \in \mathbb{V}^M$ is the bias vector, and $\psi : \mathbb{V}^M \to \mathbb{V}^M$ is the activation or transfer function.

In practice, we work with the real representation of inputs and outputs because current deep learning libraries operate almost exclusively with floating point numbers.

In practice, we work with the real representation of inputs and outputs because current deep learning libraries operate almost exclusively with floating point numbers.

A vector-valued dense layer can be emulated by a real dense layer defined by

$$\varphi(\boldsymbol{y}) = \boldsymbol{\psi}_{\mathbb{R}}\big(\varphi(\boldsymbol{s}) + \varphi(\boldsymbol{b})\big) \quad \text{with} \quad \varphi(\boldsymbol{s}) = \mathcal{M}_L(\boldsymbol{W})\varphi(\boldsymbol{x}), \tag{26}$$

where $\varphi(\boldsymbol{x}) \in \mathbb{R}^{nN}$ is a real input vector,

$$\mathcal{M}_L(\boldsymbol{W}) = \sum_{k=0}^{n-1} \boldsymbol{W}_k \otimes \boldsymbol{P}_{k:}^T \in \mathbb{R}^{nM \times nN}, \tag{27}$$

is a real synaptic weight matrix, $\varphi(\boldsymbol{b}) \in \mathbb{R}^{nM}$ is a real bias vector, and $\varphi(\boldsymbol{y}) \in \mathbb{R}^{nM}$ is the real output.

In practice, we work with the real representation of inputs and outputs because current deep learning libraries operate almost exclusively with floating point numbers.

A vector-valued dense layer can be emulated by a real dense layer defined by

$$\varphi(\boldsymbol{y}) = \boldsymbol{\psi}_{\mathbb{R}}\big(\varphi(\boldsymbol{s}) + \varphi(\boldsymbol{b})\big) \quad \text{with} \quad \varphi(\boldsymbol{s}) = \mathcal{M}_L(\boldsymbol{W})\varphi(\boldsymbol{x}), \tag{26}$$

where $\varphi(\boldsymbol{x}) \in \mathbb{R}^{nN}$ is a real input vector,

$$\mathcal{M}_L(\boldsymbol{W}) = \sum_{k=0}^{n-1} \boldsymbol{W}_k \otimes \boldsymbol{P}_{k:}^T \in \mathbb{R}^{nM \times nN}, \tag{27}$$

is a real synaptic weight matrix, $\varphi(\boldsymbol{b}) \in \mathbb{R}^{nM}$ is a real bias vector, and $\varphi(\boldsymbol{y}) \in \mathbb{R}^{nM}$ is the real output.

Therefore, a vector-valued dense layer can be computed using a real dense layer by appropriately rearranging and reusing the elements.

A vector-valued dense layer has $nM(N+1)$ parameters, while an equivalent traditional dense layer has $nM(nN+1)$ parameters.

A vector-valued dense layer has $nM(N+1)$ parameters, while an equivalent traditional dense layer has $nM(nN+1)$ parameters.

Vector-valued dense layers can be interpreted as traditional constrained dense layers, where the synaptic weights are obtained by imposing a structure that depends on the algebra.

A vector-valued dense layer has $nM(N+1)$ parameters, while an equivalent traditional dense layer has $nM(nN+1)$ parameters.

Vector-valued dense layers can be interpreted as traditional constrained dense layers, where the synaptic weights are obtained by imposing a structure that depends on the algebra.

The constraints imposed by the algebra arise from the assumption that there are intercorrelations between feature channels.

Convolutional layers are important building blocks in today's deep learning models.

Convolutional layers are important building blocks in today's deep learning models.

The vector-valued convolution of $\mathbf{W}$ and $\boldsymbol{x}$, denoted by $\mathbf{W} * \boldsymbol{x}$, is given by the sum of the cross-correlation of $\mathbf{W}(:, c, k)$ and $\boldsymbol{x}(:, c)$ on all channels $c = 1, \ldots, C$ as follows:

$$(\mathbf{W} * \boldsymbol{x})(p, k) = \sum_{c=1}^{C} \sum_{q \in D} \mathbf{W}(q, c, k) \boldsymbol{x}(p + S(q), c), \quad p \in \mathcal{D}_{\boldsymbol{y}}, \ \forall k, \quad (28)$$

where $p + S(q)$ denotes a translation that can take into account the feeds, and $\mathcal{D}_{\boldsymbol{y}}$ denotes the domain of $\boldsymbol{y}$.

Convolutional layers are important building blocks in today's deep learning models.

The vector-valued convolution of $\mathbf{W}$ and $\boldsymbol{x}$, denoted by $\mathbf{W} * \boldsymbol{x}$, is given by the sum of the cross-correlation of $\mathbf{W}(:, c, k)$ and $\boldsymbol{x}(:, c)$ on all channels $c = 1, \ldots, C$ as follows:

$$(\mathbf{W} * \boldsymbol{x})(p, k) = \sum_{c=1}^{C} \sum_{q \in D} \mathbf{W}(q, c, k) \boldsymbol{x}(p + S(q), c), \quad p \in \mathcal{D}_{\boldsymbol{y}}, \ \forall k, \quad (28)$$

where $p + S(q)$ denotes a translation that can take into account the feeds, and $\mathcal{D}_{\boldsymbol{y}}$ denotes the domain of $\boldsymbol{y}$.

The output $\boldsymbol{y}$ of a convolutional layer is given by $\boldsymbol{y} = \psi(\mathbf{W} * \boldsymbol{x} + \boldsymbol{b})$, where the activation function $\psi : \mathbb{V} \to \mathbb{V}$ is typically applied pixel-by-pixel, layer-by-layer.

Like dense layers, vector-valued convolutional layers can be emulated using traditional convolutional layers.

Like dense layers, vector-valued convolutional layers can be emulated using traditional convolutional layers.

In effect, we have

$$\varphi(\mathbf{W} * \boldsymbol{x}) = \left(\sum_{\ell=0}^{n-1} \mathbf{W}_\ell \otimes \mathbf{P}_{\ell:}^T \right) * \varphi(\boldsymbol{x}) = \sum_{\ell=0}^{n-1} (\mathbf{M}_\ell * \varphi(\boldsymbol{x})), \tag{29}$$

where $\mathbf{W} = \mathbf{W}_0 e_0 + \ldots + \mathbf{W}_{n-1} e_{n-1}$ is the representation of the filters with respect to the base $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$, $\mathbf{M}_\ell = \mathbf{W}_\ell \otimes \mathbf{P}_{\ell:}^T$ are real filters obtained using the Kronecker product, and $\varphi(\boldsymbol{x})$ is obtained by concatenating the components $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}$ of $\boldsymbol{x} = \boldsymbol{x}_0 e_0 + \ldots + \boldsymbol{x}_{n-1} e_{n-1}$ in the feature channel.

Like dense layers, vector-valued convolutional layers can be emulated using traditional convolutional layers.

In effect, we have

$$\varphi\big(\mathbf{W} * \boldsymbol{x}\big) = \left(\sum_{\ell=0}^{n-1} \mathbf{W}_\ell \otimes \mathbf{P}_{\ell:}^T\right) * \varphi(\boldsymbol{x}) = \sum_{\ell=0}^{n-1} \left(\mathbf{M}_\ell * \varphi(\boldsymbol{x})\right), \tag{29}$$

where $\mathbf{W} = \mathbf{W}_0 e_0 + \ldots + \mathbf{W}_{n-1} e_{n-1}$ is the representation of the filters with respect to the base $\mathcal{E} = \{e_0, \ldots, e_{n-1}\}$, $\mathbf{M}_\ell = \mathbf{W}_\ell \otimes \mathbf{P}_{\ell:}^T$ are real filters obtained using the Kronecker product, and $\varphi(\boldsymbol{x})$ is obtained by concatenating the components $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}$ of $\boldsymbol{x} = \boldsymbol{x}_0 e_0 + \ldots + \boldsymbol{x}_{n-1} e_{n-1}$ in the feature channel.

Examples of vector-valued convolutions and their implementations can be found in (Gaudet and Maida, 2018; Grassucci et al., 2022; Vieira and Valle, 2022b).

In addition to convolutional layers, modern deep learning models use other structures, such as layer pooling, batch normalization, and others (Géron, 2019).

In addition to convolutional layers, modern deep learning models use other structures, such as layer pooling, batch normalization, and others (Géron, 2019).

While vector-valued versions of these structures are a topic of future research, we can currently use a simple approach.

In addition to convolutional layers, modern deep learning models use other structures, such as layer pooling, batch normalization, and others (Géron, 2019).

While vector-valued versions of these structures are a topic of future research, we can currently use a simple approach.

This approach involves combining traditional structures with real-valued emulation of vector-valued convolutional and dense layers, as described in (26) and (29).

In addition to convolutional layers, modern deep learning models use other structures, such as layer pooling, batch normalization, and others (Géron, 2019).

While vector-valued versions of these structures are a topic of future research, we can currently use a simple approach.

This approach involves combining traditional structures with real-valued emulation of vector-valued convolutional and dense layers, as described in (26) and (29).

Although it may seem overly simplistic, this approach can provide valuable insights into vector-valued blocks.

Concluding Remarks

In this talk, we revised the basic concepts of mathematical morphology on complete lattices.

Concluding Remarks

In this talk, we revised the basic concepts of mathematical morphology on complete lattices.

Elementary morphological operators can be found, for example, in activation functions and pooling layers in current deep-learning models (Velasco-Forero and Angulo, 2022).

Concluding Remarks

In this talk, we revised the basic concepts of mathematical morphology on complete lattices.

Elementary morphological operators can be found, for example, in activation functions and pooling layers in current deep-learning models (Velasco-Forero and Angulo, 2022).

Furthermore, we addressed vector-valued morphology and noted the relevance of considering the intercorrelation between features for some image processing and analysis tasks.

Concluding Remarks

In this talk, we revised the basic concepts of mathematical morphology on complete lattices.

Elementary morphological operators can be found, for example, in activation functions and pooling layers in current deep-learning models (Velasco-Forero and Angulo, 2022).

Furthermore, we addressed vector-valued morphology and noted the relevance of considering the intercorrelation between features for some image processing and analysis tasks.

This remark motivated us to develop V-nets, which naturally incorporate intercorrelation between features through algebra (Valle, 2024).

The broad framework of V-nets also includes hypercomplex-valued neural networks.

The broad framework of V-nets also includes hypercomplex-valued neural networks.

The use of hypercomplex algebras can result in additional geometric and algebraic properties to the network (Hirose, 2012; Lee et al., 2022; Parcollet et al., 2020; Ruhe et al., 2023b).

The broad framework of V-nets also includes hypercomplex-valued neural networks.

The use of hypercomplex algebras can result in additional geometric and algebraic properties to the network (Hirose, 2012; Lee et al., 2022; Parcollet et al., 2020; Ruhe et al., 2023b).

Successful applications of V-nets include:

- localization and detection of sound events (Grassucci et al., 2023),
- ultrasound image enhancement (Lei et al., 2023),
- acute lymphoblastic leukemia detection (Vieira and Valle, 2022b),
- fluid dynamics simulations (Ruhe et al., 2023b).

V-nets outperformed traditional networks in these applications.

In this talk, we present the relationship between V-nets and traditional neural networks.

In this talk, we present the relationship between V-nets and traditional neural networks.

Specifically, we show how dense, vector-valued convolutional layers can be emulated using traditional layers, which allows the implementation of V-nets using current deep learning libraries such as `tensorflow` and `pytorch`.

In this talk, we present the relationship between V-nets and traditional neural networks.

Specifically, we show how dense, vector-valued convolutional layers can be emulated using traditional layers, which allows the implementation of V-nets using current deep learning libraries such as `tensorflow` and `pytorch`.

Future research includes:

- Selecting the appropriate algebra for an application.
- Efficient techniques for learning algebra
- Studying vector-valued activation functions and structures used in deep learning.

Thank you very much!

References (1)

J. Angulo. Geometric algebra colour image representations and derived total orderings for morphological operators – Part I: Colour quaternions. *Journal of Visual Communication and Image Representation*, 21(1):33–48, 1 2010. ISSN 1047-3203. doi: 10.1016/J.JVCIR.2009.10.002.

P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia. Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks*, 10(2):335–342, 3 1997. doi: 10.1016/S0893-6080(96)00048-2.

G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, 3 edition, 1993.

B. Burgeth and A. Kleefeld. An approach to color-morphology based on Einstein addition and Loewner order. *Pattern Recognition Letters*, 47(0):29–39, 2014.

References (2)

F. Catoni, D. Boccaletti, R. Cannata, V. Catoni, E. Nichelatti, and P. Zampetti. *The Mathematics of Minkowski Space-Time*. Birkhäuser Basel, 2008. doi: 10.1007/978-3-7643-8614-6.

Z.-C. Fan, T.-S. T. Chan, Y.-H. Yang, and J.-S. R. Jang. Backpropagation With N-D Vector-Valued Neurons Using Arbitrary Bilinear Products. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. ISSN 2162-237X. doi: 10.1109/TNNLS.2019.2933882. URL https://ieeexplore.ieee.org/document/8826334/.

G. Franchi, A. Fehri, and A. Yao. Deep morphological networks. *Pattern Recognition*, 102, 6 2020. ISSN 00313203. doi: 10.1016/j.patcog.2020.107246.

References (3)

C. J. Gaudet and A. S. Maida. Deep Quaternion Networks. *Proceedings of the International Joint Conference on Neural Networks*, 2018-July, 10 2018. doi: 10.1109/IJCNN.2018.8489651.

A. 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.

J. Goutsias, H. J. A. M. Heijmans, and K. Sivakumar. Morphological Operators for Image Sequences. *Computer vision and image understanding*, 62:326–346, 1995.

References (4)

E. Grassucci, A. Zhang, and D. 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/.

E. Grassucci, G. Mancini, C. Brignone, A. Uncini, and D. 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.

H. Heijmans. *Morphological Image Operators*. Academic Press, New York, NY, 1994.

References (5)

H. J. A. M. Heijmans. Mathematical Morphology: A Modern Approach in Image Processing Based on Algebra and Geometry. *SIAM Review*, 37(1):1–36, 1995.

A. Hirose. *Complex-Valued Neural Networks*. Studies in Computational Intelligence. Springer, Heidelberg, Germany, 2nd editio edition, 2012.

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.

C. Y. Lee, H. Hasegawa, and S. Gao. Complex-Valued Neural Networks: A Comprehensive Survey. *IEEE/CAA Journal of Automatica Sinica*, 9(8):1406–1426, 8 2022. ISSN 23299274. doi: 10.1109/JAS.2022.105743.

References (6)

Z. Lei, S. Gao, H. Hasegawa, Z. Zhang, M. Zhou, and K. Sedraoui. Fully Complex-Valued Gated Recurrent Neural Network for Ultrasound Imaging. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2023. ISSN 2162-237X. doi: 10.1109/TNNLS.2023.3282231. URL https://ieeexplore.ieee.org/document/10153088/.

C. F. 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.

S. Miron, J. Flamant, N. L. Bihan, P. Chainais, and D. Brie. Quaternions in Signal and Image Processing: A comprehensive and objective overview. *IEEE Signal Processing Magazine*, 40(6): 26–40, 9 2023. ISSN 1053-5888. doi: 10.1109/MSP.2023.3278071.

References (7)

R. Mondal, S. Santra, and B. Chanda. Dense Morphological Network: An Universal Function Approximator, 2019.

C. J. Mulvey. &. *Rend. Circ. Mat. Palermo*, 12:99–104, 1986.

K. Nogueira, J. Chanussot, M. D. Mura, and J. A. Santos. An Introduction to Deep Morphological Networks. *IEEE Access*, 9: 114308–114324, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3104405.

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

G. X. Ritter and P. Sussner. An Introduction to Morphological Neural Networks. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 709–717, Vienna, Austria, 1996.

References (8)

G. X. Ritter and G. Urcid. Lattice Algebra Approach to Single-Neuron Computation. *IEEE Transactions on Neural Networks*, 14(2):282–295, 2003.

G. X. Ritter, P. Sussner, and J. L. D. de Leon. Morphological Associative Memories. *IEEE Transactions on Neural Networks*, 9 (2):281–293, 1998.

D. Ruhe, J. Brandstetter, and P. Forré. Clifford Group Equivariant Neural Networks. 5 2023a.

D. Ruhe, J. K. Gupta, S. de Keninck, M. Welling, and J. Brandstetter. Geometric Clifford Algebra Networks. 2 2023b. URL `http://arxiv.org/abs/2302.06594`.

References (9)

A. S. Santos and M. E. Valle. Max-plus and min-plus projection autoassociative morphological memories and their compositions for pattern classification. *Neural Networks*, 100:84–94, 2018. doi: 10.1016/j.neunet.2018.01.013.

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

J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

P. Soille. *Morphological Image Analysis*. Springer Verlag, Berlin, 1999.

References (10)

J. G. Stell. Why mathematical morphology needs quantales. In M. Wilkinson and J. Roerdink, editors, *Abstract book of the 9th International Symposium on Mathematical Morphology (ISMM{\rq}2009)*, pages 13–16, The Netherlands, 2009. University of Groningen.

F. Stenger. Kronecker Product Extensions of Linear Operators. *SIAM Journal on Numerical Analysis*, 5(2):422–435, 6 1968. ISSN 0036-1429. doi: 10.1137/0705033.

P. Sussner and E. L. Esmi. Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm. *Information Sciences*, 181(10): 1929–1950, 5 2011. ISSN 0020-0255. doi: 10.1016/J.INS.2010.03.016.

References (11)

P. Sussner and M. E. Valle. Grayscale Morphological Associative Memories. *IEEE Transactions on Neural Networks*, 17(3): 559–570, 2006.

M. E. 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.

M. E. Valle and P. Sussner. A general framework for fuzzy morphological associative memories. *Fuzzy Sets and Systems*, 159(7):747–768, 4 2008. ISSN 01650114. doi: 10.1016/j.fss.2007.10.010.

References (12)

S. Velasco-Forero and J. Angulo. Vector Ordering and Multispectral Morphological Image Processing. In M. E. Celebi and B. Smolka, editors, *Advances in Low-Level Color Image Processing*, pages 223–239, Dordrecht, 2014. Springer Netherlands. doi: 10.1007/978-94-007-7584-8{_}7.

S. Velasco-Forero and J. Angulo. MorphoActivation: Generalizing ReLU Activation Function by Mathematical Morphology. pages 449–461. 2022. doi: 10.1007/978-3-031-19897-7{_}35.

G. Vieira and M. E. 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.

References (13)

G. Vieira and M. E. 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.

G. Vieira, E. Grassucci, M. E. Valle, and D. Comminiello. Dual Quaternion Rotational and Translational Equivariance in 3D Rigid Motion Modelling. In *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 9 2023. ISBN 979-8-3503-2411-2. doi: 10.1109/MLSP55844.2023.10285888.

References (14)

A. Zhang, Y. Tay, S. Zhang, A. Chan, A. T. Luu, S. C. Hui, and J. Fu. Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with $1/n$ Parameters. 2 2021. doi: 10.48550/arxiv.2102.08597. URL https://arxiv.org/abs/2102.08597v1.