Information Fusion 99 (2023) 101893

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full Length Article

Generalizing max pooling via (a, b)-grouping functions for Convolutional Neural Networks

Iosu Rodriguez-Martinez [a], Tiago da Cruz Asmus [b], Graçaliz Pereira Dimuro [a,c], Francisco Herrera [d], Zdenko Takáč [e], Humberto Bustince [a,*]

[a] Universidad Pública de Navarra, Departamento de Estadística, Informática y Matemáticas, Campus Arrosadia s/n, Pamplona, 31006, Navarra, Spain
[b] Universidade Federal do Rio Grande, Instituto de Matemática, Estadística e Física, Av. Itália km 08, Campus Carreiros, Rio Grande, 96201-900, Rio Grande, Brazil
[c] Universidade Federal do Rio Grande, Centro de ciencia Computacionais, Av. Itália km 08, Campus Carreiros, Rio Grande, 96201-900, Rio Grande, Brazil
[d] Universidad de Granada, Instituto Andaluz Interuniversitario en Data Science and Computational Intelligence, Av. del Conocimiento 41, Granada, 18071, Granada, Spain
[e] Slovak University of Technology in Bratislava, Faculty of Materials Science and Technology in Trnava, Jana Bottu 2781/25, Trnava, 917 24, Trnava, Slovakia

ARTICLE INFO

Keywords:
Convolutional neural networks
Grouping functions
Pooling functions
Image classification

ABSTRACT

Due to their high adaptability to varied settings and effective optimization algorithm, Convolutional Neural Networks (CNNs) have set the state-of-the-art on image processing jobs for the previous decade. CNNs work in a sequential fashion, alternating between extracting significant features from an input image and aggregating these features locally through "pooling" functions, in order to produce a more compact representation.

Functions like the arithmetic mean or, more typically, the maximum are commonly used to perform this downsampling operation. Despite the fact that many studies have been devoted to the development of alternative pooling algorithms, in practice, "max-pooling" still equals or exceeds most of these possibilities, and has become the standard for CNN construction.

In this paper we focus on the properties that make the maximum such an efficient solution in the context of CNN feature downsampling and propose its replacement by grouping functions, a family of functions that share those desirable properties. In order to adapt these functions to the context of CNNs, we present (a, b)-grouping functions, an extension of grouping functions to work with real valued data. We present different construction methods for (a, b)-grouping functions, and demonstrate their empirical applicability for replacing max-pooling by using them to replace the pooling function of many well-known CNN architectures, finding promising results.

1. Introduction

The irruption of Deep Learning [1] during the last decade has revolutionized the field of machine learning research, with impressive results in fields as diverse as medicine [2,3], natural language processing [4] or synthetic image generation [5]. In the field of computer vision, Convolutional Neural Networks (CNNs) have been established as the state-of-the-art technique for classification [6–8] and segmentation tasks [2,9], among others [10–12].

Unlike traditional Computer Vision methods such as Bag of Features (BoF) [13], the parameters of these models are automatically optimized through gradient descent optimization in a supervised way, easing their application and motivating their wide adoption. CNNs extract complex visual features in a sequential process, generating feature vectors which can be later fed to different algorithms.

If no feature reduction technique were applied, the dimensionality of these feature vectors would be too high. Pooling layers take care of this, performing image downsampling through the fusion of local areas of the feature images the model works with, while trying to preserve the most discriminative values. This data fusion process is usually performed by simple operations such as the arithmetic mean or, more commonly in practice, the maximum. Both theoretical studies [14,15] as well as empirical claims seem to set maximum pooling as the default pooling operator.

Even so, there are some problems with maximum pooling. While providing some amount of shift invariance to the model, maximum

* Corresponding author.
E-mail address: bustince@unavarra.es (H. Bustince).

https://doi.org/10.1016/j.inffus.2023.101893
Received 28 February 2023; Received in revised form 13 June 2023; Accepted 15 June 2023
Available online 21 June 2023

pooling discards many potentially discriminative information in favour of prioritizing high activations. This, in turn, implies that the gradient flow of the model gets truncated for all non maximum values of a neighbourhood, which may slow convergence. Although some alternatives have been proposed in order to try to address these problems [16–19], max pooling remains the most commonly adapted strategy.

Previous work has demonstrated that other functions with similar properties to the maximum can act as powerful alternatives as pooling functions. In [20], t-conorm functions, a family of functions to which the maximum belongs, outperformed models which implemented max pooling, despite requiring a suboptimal implementation. Grouping functions [21,22] are a generalization of t-conorms which ignore the associative property, avoiding those implementation mishaps. They have been used with success in different domains such as decision making [21] and image thresholding [23], and we consider them an interesting family of functions to explore in search of a replacement for the maximum as pooling operator.

In this work, we present (a, b)-grouping functions, an adaptation of the concept of grouping function to work with real valued data, which makes them suitable to act as pooling the operator of a CNN. Several construction methods are presented, and many examples of grouping functions are explored, in search of the most interesting properties for CNNs. In particular, we demonstrate that (a, b)-grouping functions can be constructed through the composition of (a, b)-grouping functions as well as through their convex combination, which offers a wide spectrum of functions to apply.

We empirically test the suitability of these functions in a range of CNN models of different depth and complexity, obtaining favourable results with most options. We also analyse the activations and gradients produced by these functions, comparing them with the effect of traditional pooling operators.

The paper is structured in the following way: Section 2 recalls some important notions about CNNs and grouping functions; Section 3 introduces (a, b)-grouping functions, providing several construction methods and examples of such functions, as well as exploring their suitability as pooling candidates; Section 4 is dedicated to our new pooling operators, as well as to reviewing some other recently introduced in the literature pooling layers. Section 5 presents our experiments and offers some insights into the challenges of replacing maximum pooling by other functions; finally, Section 6 concludes the paper with some final remarks and future lines of research.

2. Preliminaries

2.1. Convolutional Neural Networks

CNNs are a family of Neural Network specialized in the "automatic" extraction of local features from a given data source, such as an image or video. We say that this process is "automatic" because the backpropagation algorithm is used for finetuning the model parameters through gradient descent optimization [1], instead of requiring careful manual adjustments.

Similarly to other neural network models, CNNs are formed by a series of "layers" which act sequentially over a given input. In particular, the feature extraction process is taken care of in "convolutional layers", where a series of "feature filters" are convolved over different regions of an input, before applying a non-linear activation function to the result. This generates an output which represents the presence or absence of a given feature along the different positions of the input.

In the case of image data, a convolutional layer counts with a series of small two dimensional filters which represent visual characteristics. After filtering the input image by n such filters, n different "feature images" are generated. Each of them is appended as a different channel of an output matrix which the layer finally outputs. Given the sequential nature of CNNs, future convolution layers act over the output of the initial layers. This implies that the features extracted at deeper levels of the model are progressively more complex, since they combine the simpler features obtained by previous layers.

The extracted features can be later fed as the input of several algorithms, depending on the task at hand. In the case of image classification, a Multilayer Perceptron (MLP) is typically used, since it is an efficient classifier which shares the same optimization procedure as the rest of the model. Whatever the classifier, however, its behaviour can be expected to improve the smaller the number of features fed to it is. Unfortunately, in order to capture as rich descriptors as possible from the input image, convolutional layers typically have many filters, which increases the number of channels of the generated feature matrices.

Several strategies can be used in order to reduce the dimensionality of the feature extractor output. One of the most common ones is the addition of "pooling layers" along the feature extractor, in order to perform an image downsampling process to the obtained feature images. In this way, each of these feature images is divided in small disjoint windows whose values are aggregated into single representatives, through functions such as the arithmetic mean or the maximum. This strategy is based on the idea that much of the information represented by feature images is redundant, and helps to provide the model with some amount of invariance to slight displacement of the detected features.

Some authors have proposed replacing simple average and maximum pooling by other data fusion operations: in [16] Stochastic pooling is proposed, which chooses one of the input values as output according to a multinomial distribution based on their magnitudes; Ordered Weighted Average aggregation operators (OWA) [24] were proposed in [18] as another trial at replacing pooling functions by learnable aggregation functions; in [25], another family of functions based on the minimization of the distance among the input values and the produced reduced output are proposed; in [17] authors propose to combine both average and maximum pooling through a strategy referred to as Mixed Pooling, via a convex combination with learnable coefficients; in [19] we proposed a generalization of this approach in which several increasing functions were combined obtaining increasing pooling functions as a result.

A new pooling strategy is proposed in the remainder of this paper, based on the concept of grouping functions, which we recall in the following section.

2.2. Grouping functions

In this section, we recall some theoretical concepts which have served as a basis for the new proposed pooling functions.

Let us denote $\vec{x} = (x_1, \dots, x_n) \in [0, 1]^n$.

Definition 2.1 (*[26]*)**.** A function $N : [0, 1] \to [0, 1]$ is a fuzzy negation if the following conditions hold:

(N1) $N(0) = 1$ and $N(1) = 0$;

(N2) If $x \leq y$ then $N(y) \leq N(x)$, for all $x, y \in [0, 1]$.

If N also satisfies the involutive property,

(N3) $N(N(x)) = x$, for all $x \in [0, 1]$,

then it is said to be a strong fuzzy negation.

Example 2.1. The Zadeh negation given, for all $x \in [0, 1]$, by

$$N_Z(x) = 1 - x,$$

is a strong fuzzy negation.

Example 2.2. The functions $N_1 = 1 - x^2$ and $N_2 = \sqrt{1 - x}$ are examples of non-strong fuzzy negations.

Definition 2.2 ([26]). Given a strong fuzzy negation $N : [0,1] \to [0,1]$ and function $F : [0,1]^n \to [0,1]$, then the function $F^N : [0,1]^n \to [0,1]$ defined, for all $\vec{x} \in [0,1]^n$, by

$$F^N(\vec{x}) = N(F(N(x_1), \dots, N(x_n))), \tag{1}$$

is the N-dual of F.

Functions of the type $F : [0,1]^n \to [0,1]$ are usually referred to as fusion functions [27]. An interesting subclass of fusion functions is that of aggregation functions.

Note that from now on, by the monotonicity (increasingness /decreasingness) of n-ary functions we understand the monotonicity in each variable.

Definition 2.3 ([28]). An aggregation function is any function $A : [0,1]^n \to [0,1]$ that respects the following conditions:

(A1) A is increasing;

(A2) $A(0, \dots, 0) = 0$ and $A(1, \dots, 1) = 1$.

Example 2.3. Consider $\vec{w} \in [0,1]^n$ such that $\sum_{i=1}^n w_i = 1$. Then, the function $AW : [0,1]^n \to [0,1]$ (weighted arithmetic mean), given, for all $\vec{x} \in [0,1]^n$, by

$$AW(\vec{x}) = \sum_{i=1}^{n} w_i \cdot x_i, \tag{2}$$

is an aggregation function. When $w_i = 1/n$ for every $i \in \{1, \dots, n\}$, we can rewrite Eq. (2) to express the usual arithmetic mean $AM : [0,1]^n \to [0,1]$, given by

$$AM(\vec{x}) = \frac{\sum_{i=1}^n x_i}{n}, \tag{3}$$

which is also an aggregation function.

There are many subclasses of aggregation functions defined in the literature. Here we highlight some of them that are going to be of importance to this work.

Definition 2.4 ([29,30]). A function $O : [0,1]^n \to [0,1]$ is an n-dimensional overlap function if, for all $\vec{x} \in [0,1]^n$, the following conditions hold:

(O1) O is symmetric;

(O2) $O(\vec{x}) = 0$ if and only if $\prod_{i=1}^n x_i = 0$;

(O3) $O(\vec{x}) = 1$ if and only if $\prod_{i=1}^n x_i = 1$;

(O4) O is increasing;

(O5) O is continuous.

A 2-dimensional overlap function is just called overlap function [22, 31].

Example 2.4.

(a) The function $O_{min} : [0,1]^n \to [0,1]$ (minimum) given, for all $\vec{x} \in [0,1]^n$, by

$$O_{min}(\vec{x}) = \min\{x_1, \dots, x_n\} \tag{4}$$

is an n-dimensional overlap function.

(b) The function $O_{geom} : [0,1]^n \to [0,1]$ (geometric mean), given, for all $\vec{x} \in [0,1]^n$, by

$$O_{geom}(\vec{x}) = \sqrt[n]{\prod_{i=1}^{n} x_i}, \tag{5}$$

is an n-dimensional overlap function.

Table 1
Examples of n-dimensional Grouping Functions.

$G_{max}(x_1, \dots, x_n) = \left(\max\{x_1, \dots, x_n\}\right)^p$, with $p > 0$
$G_{prod}(x_1, \dots, x_n) = 1 - \prod_{i=1}^n (1 - x_i)^p$, with $p > 0$
$G_{geom}(x_1, \dots, x_n) = 1 - \sqrt[n]{\prod_{i=1}^n (1 - x_i)^p}$, with $p > 0$
$G_{ob}(x_1, \dots, x_n) = 1 - \left(\sqrt{\left(\prod_{i=1}^n 1 - x_i\right) \cdot \min\{1 - x_1, \dots, 1 - x_n\}}\right)$
$G_u(x_1, \dots, x_n) = \frac{\max\{x_1, \dots, x_n\}}{\max\{x_1, \dots, x_n\} + \sqrt[n]{\prod_{i=1}^n (1 - x_i)}}$

Definition 2.5 ([29]). A function $G : [0,1]^n \to [0,1]$ is said to be an n-dimensional grouping function if, for all $\vec{x} \in [0,1]^n$, the following conditions hold:

(G1) G is symmetric;

(G2) $G(\vec{x}) = 0$ if and only if $x_i = 0$ for all $i \in \{1, \dots, n\}$;

(G3) $G(\vec{x}) = 1$ if and only if there exists $i \in \{1, \dots, n\}$ such that $x_i = 1$;

(G4) G is increasing;

(G5) G is continuous.

By N-duality, one can obtain n-dimensional grouping functions from n-dimensional overlap functions, and vice-versa.

Example 2.5.

(a) The function $G_{max} : [0,1]^n \to [0,1]$ (maximum), given, for all $\vec{x} \in [0,1]^n$, by

$$G_{max}(\vec{x}) = \max\{x_1, \dots, x_n\}, \tag{6}$$

is an n-dimensional grouping function.

(b) The function $G_{geom} : [0,1]^n \to [0,1]$ (dual of the geometric mean), given, for all $\vec{x} \in [0,1]^n$, by

$$G_{geom}(\vec{x}) = 1 - \sqrt[n]{\prod_{i=1}^{n} 1 - x_i}, \tag{7}$$

is an n-dimensional grouping function.

In Table 1, we show some other examples of n-dimensional grouping functions.

2.3. (a, b)-aggregation functions

Grouping functions are restricted to the domain $[0,1]$, but CNNs operate with real values. A strategy for adapting the definition of several aggregation functions to work with real valued data is recalled here.

Let $a, b \in \mathbb{R}$, such that $a < b$. In [28], fusion functions, as well as aggregation functions were already defined in the context of a domain $[a,b]$. Here, to avoid confusion, we will call them fusion and aggregation functions only when $a = 0$ and $b = 1$ (Definition 2.3). Otherwise, we will call them (a,b)-fusion functions and (a,b)-aggregation functions, just to standardize the notation.

(a,b)-fusion functions are arbitrary functions of the type $F^{a,b} : [a,b]^n \to [a,b]$. The definition of (a,b)-aggregation function is given as follows:

Definition 2.6 ([28]). An (a,b)-aggregation function is any function $A^{a,b} : [a,b]^n \to [a,b]$ that respects the following conditions:

(AB1) $A^{a,b}$ is increasing;

(AB2) $A^{a,b}(a, \dots, a) = a$ and $A^{a,b}(b, \dots, b) = b$.

Example 2.6.

(a) The arithmetic mean $AM : [a,b]^n \to [a,b]$, given by Eq. (3) is an (a,b)-aggregation function;

(b) If $a \neq 0$ or $b \neq 1$, then neither the geometric mean $O_{geom} : [a,b]^n \to [a,b]$, given by Eq. (5), nor its dual $G_{geom} : [a,b]^n \to [a,b]$, given by Eq. (7), are (a,b)-aggregation functions.

In [20], different (a,b)-aggregation functions were defined in a similar manner as Definition 2.6. Let us recall the definition of (a,b)-overlap functions:

Definition 2.7 ([20]). A function $O^{a,b} : [a,b]^n \to [a,b]$ is said to be an (a,b)-overlap function if, for all $\vec{x} \in [a,b]^n$, the following conditions hold:

(OAB1) $O^{a,b}$ is symmetric;

(OAB2) $O^{a,b}(x_1,\dots,x_n) = a$ if and only if $\prod_{i=1}^n (x_i - a) = 0$;

(OAB3) $O^{a,b}(x_1,\dots,x_n) = b$ if and only if $\prod_{i=1}^n (\frac{x_i - a}{b-a}) = 1$;

(OAB4) $O^{a,b}$ is increasing;

(OAB5) $O^{a,b}$ is continuous.

3. (a,b)-grouping functions

Here, we introduce the concept of (a,b)-grouping function as follows:

Definition 3.1. A function $G^{a,b} : [a,b]^n \to [a,b]$ is said to be an (a,b)-grouping function if, for all $\vec{x} \in [a,b]^n$, the following conditions hold:

(GAB1) $G^{a,b}$ is symmetric;

(GAB2) $G^{a,b}(\vec{x}) = a$ if and only if $x_i = a$ for all $i \in \{1,\dots,n\}$;

(GAB3) $G^{a,b}(\vec{x}) = b$ if and only if there exists $i \in \{1,\dots,n\}$ such that $x_i = b$;

(GAB4) $G^{a,b}$ is increasing;

(GAB5) $G^{a,b}$ is continuous.

It is immediate that every (a,b)-grouping function is also an (a,b)-aggregation function, but the converse does not hold.

Example 3.1. The function $G^{a,b}_{max} : [a,b]^n \to [a,b]$ (maximum), given, for all $\vec{x} \in [0,1]^n$, by

$$G^{a,b}_{max}(\vec{x}) = \max\{x_1,\dots,x_n\}, \tag{8}$$

is an n-dimensional (a,b)-grouping function.

3.1. Construction methods

Although Definition 3.1 seems intuitive and keeps the same properties of Definition 2.5 in the context of the domain $[a,b]$, it is not trivial to obtain expressions for (a,b)-grouping functions, since the expressions of some known n-dimensional grouping functions (as the ones shown in Table 1) do not respect the conditions from Definition 3.1. So, in the following, we introduce some construction methods for (a,b)-grouping functions.

Theorem 3.1. *Consider a fusion function $G : [0,1]^n \to [0,1]$, an increasing and bijective function $\phi : [a,b] \to [0,1]$ and an (a,b)-fusion function $G^{a,b} : [a,b]^n \to [a,b]$ given, for all $x_1,\dots,x_n \in [a,b]$, by*

$$G^{a,b}(x_1,\dots,x_n) = \phi^{-1}\left(G\left(\phi(x_1),\dots,\phi(x_n)\right)\right), \tag{9}$$

Then, $G^{a,b}$ is an n-dimensional (a,b)-grouping function if and only if G is an n-dimensional grouping function.

Proof. $\to$ Suppose that $G^{a,b}$ is an n-dimensional (a,b)-grouping function. Then, it is immediate that G is increasing, symmetric and continuous. Let us prove that G respects the remaining conditions of Definition 2.5:

(G2)

$$\begin{aligned}
&G(x_1,\dots,x_n) = 0\\
&\Leftrightarrow G(\phi(\phi^{-1}(x_1)),\dots,\phi(\phi^{-1}(x_n))) = 0,\\
&\quad \text{since } \phi \text{ is bijective}\\
&\Leftrightarrow \phi^{-1}(G(\phi(\phi^{-1}(x_1)),\dots,\phi(\phi^{-1}(x_n)))) = \phi^{-1}(0)\\
&\Leftrightarrow G^{a,b}(\phi^{-1}(x_1),\dots,\phi^{-1}(x_n)) = a,\\
&\quad \text{by Eq. (9)}\\
&\Leftrightarrow \phi^{-1}(x_i) = a, \text{ for all } i \in \{1,\dots,n\},\\
&\quad \text{by (GAB2)}\\
&\Leftrightarrow x_i = 0, \text{ for all } i \in \{1,\dots,n\}.
\end{aligned}$$

(G3)

$$\begin{aligned}
&G(x_1,\dots,x_n) = 1\\
&\Leftrightarrow G(\phi(\phi^{-1}(x_1)),\dots,\phi(\phi^{-1}(x_n))) = 1,\\
&\quad \text{since } \phi \text{ is bijective}\\
&\Leftrightarrow \phi^{-1}(G(\phi(\phi^{-1}(x_1)),\dots,\phi(\phi^{-1}(x_n)))) = \phi^{-1}(1)\\
&\Leftrightarrow G^{a,b}(\phi^{-1}(x_1),\dots,\phi^{-1}(x_n)) = b,\\
&\quad \text{by Eq. (9)}\\
&\Leftrightarrow \phi^{-1}(x_i) = b, \text{ for some } i \in \{1,\dots,n\},\\
&\quad \text{by (GAB3)}\\
&\Leftrightarrow x_i = 1, \text{ for some } i \in \{1,\dots,n\}.
\end{aligned}$$

($\Leftarrow$) Suppose that G is an n-dimensional grouping function. From **(G1)**, **(G4)** and **(G5)**, we also have that $G^{a,b}$ is symmetric, increasing and continuous. Now, let us prove that it respects the remaining conditions of Definition 3.1:

(GAB2) Suppose that $G^{a,b}(\vec{x}) = a$, for some $\vec{x} \in [a,b]^n$. Then, from Eq. (9), we have that:

$$a = \phi^{-1}\left(G\left(\phi(x_1),\dots,\phi(x_n)\right)\right) \text{ if and only if}$$
$$0 = G\left(\phi(x_1),\dots,\phi(x_n)\right),$$

since ϕ is increasing and bijective. From **(G2)**, it follows that:

$$\phi(x_i) = 0 \text{ for all } i \in \{1,\dots,n\} \text{ if and only if}$$
$$x_i = a \text{ for all } i \in \{1,\dots,n\}.$$

(GAB3) Suppose that $G^{a,b}(\vec{x}) = b$, for some $\vec{x} \in [a,b]^n$. Then, from Eq. (9), we have that:

$$b = \phi^{-1}\left(G\left(\phi(x_1),\dots,\phi(x_n)\right)\right) \text{ if and only if}$$
$$1 = G\left(\phi(x_1),\dots,\phi(x_n)\right),$$

since ϕ is increasing and bijective. From **(G3)**, it follows that:

$$\phi(x_i) = 1 \text{ for some } i \in \{1,\dots,n\} \text{ if and only if}$$
$$x_i = b \text{ for some } i \in \{1,\dots,n\}. \quad \square$$

Example 3.2. Consider the n-dimensional grouping functions G_{prod}, $G_{geom} : [0,1]^n \to [0,1]$ defined in Table 1 and Example 2.5, respectively, and the increasing and bijective function $\phi : [a,b] \to [0,1]$, defined, for all $x \in [a,b]$, by

$$\phi(x) = \left(\frac{x-a}{b-a}\right)^p, \ p > 0. \tag{10}$$

Then, the functions $G^{a,b}_{prod}, G^{a,b}_{geom} : [a,b]^n \to [a,b]$, given, for all $\vec{x} \in [a,b]^n$, respectively, by

$$G^{a,b}_{prod}(\vec{x}) = \phi^{-1}\left(G_{prod}\left(\phi(x_1),\dots,\phi(x_n)\right)\right) \tag{11}$$

and

$$G^{a,b}_{geom}(\vec{x}) = \phi^{-1}\left(G_{geom}\left(\phi(x_1),\dots,\phi(x_n)\right)\right) \tag{12}$$

are n-dimensional (a,b)-grouping functions. By taking $p=1$, we can rewrite Eqs. (11) and (12), respectively as follows:

$$G_{prod}{}^{a,b}(x_1,\dots,x_n) = G_{prod}\left(\frac{x_1-a}{b-a},\dots,\frac{x_n-a}{b-a}\right)\cdot(b-a)+a, \tag{13}$$

and

$$G_{geom}{}^{a,b}(x_1,\dots,x_n) = G_{geom}\left(\frac{x_1-a}{b-a},\dots,\frac{x_n-a}{b-a}\right)\cdot(b-a)+a, \tag{14}$$

For $p=1$, the (a,b)-grouping function $G^{a,b}_{geom}$, can be simplified as follows:

$$\begin{aligned} &G_{geom}{}^{a,b}(x_1,\dots,x_n) = \\ &G_{geom}\left(\frac{x_1-a}{b-a},\dots,\frac{x_n-a}{b-a}\right)\cdot(b-a)+a = \\ &\left(1-\sqrt[n]{\prod_{i=1}^{n}\left(1-\frac{x_i-a}{b-a}\right)}\right)\cdot(b-a)+a = \\ &b-\sqrt[n]{\prod_{i=1}^{n}(b-x_i)} \end{aligned} \tag{15}$$

Remark 3.1. Any n-dimensional grouping function (such as the ones in Table 1) can be the core of the construction method presented in Theorem 3.1. The constructed n-dimensional (a,b)-grouping function is a counterpart in $[a,b]$ for the core n-dimensional grouping function defined in $[0,1]$.

The next construction method derives from the composition of (a,b)-grouping functions by an (a,b)-aggregation function with some conditions.

Theorem 3.2. *Consider a continuous (a,b)-aggregation function $A^{a,b}:[a,b]^m\to[a,b]$, such that*

(PA*) *$A^{a,b}(\vec{x})=a$ if and only if $x_i=a$ for some $i\in\{1,\dots,m\}$;*

(PB*) *$A^{a,b}(\vec{x})=b$ if and only if $x_i=b$ for some $i\in\{1,\dots,m\}$ and $x_i\neq a$ for all $i\in\{1,\dots,m\}$,*

and a tuple $\overrightarrow{G^{a,b}}=(G_1^{a,b},\dots,G_m^{a,b})$ of n-dimensional (a,b)-grouping functions. Then, the mapping $A^{a,b}_{\overrightarrow{G^{a,b}}}:[a,b]^n\to[a,b]$, defined for all $\vec{x}\in[a,b]^n$, by

$$A^{a,b}_{\overrightarrow{G^{a,b}}}(\vec{x}) = A^{a,b}(G_1^{a,b}(\vec{x}),\dots,G_m^{a,b}(\vec{x})), \tag{16}$$

is an n-dimensional (a,b)-grouping function.

Proof. It is immediate that $A^{a,b}_{\overrightarrow{G^{a,b}}}$ is well defined. Then, by **(G1)**, **(G4)** and **(G5)**, we have that $A^{a,b}_{\overrightarrow{G^{a,b}}}$ respects conditions **(GAB1)**, **(GAB4)** and **(GAB5)**. Now, let us prove that $A^{a,b}_{\overrightarrow{G^{a,b}}}$ respects the remaining conditions of Definition 3.1:

(GAB2) Suppose that $A^{a,b}_{\overrightarrow{G^{a,b}}}(\vec{x})=a$, for some $\vec{x}\in[a,b]^n$. Then, by Eq. (16) and **(PA*)**, it follows that:

$G_j^{a,b}(\vec{x})=a$ for some $j\in\{1,\dots,m\}\Leftrightarrow\vec{x}=(a,\dots,a)$

by (**GAB2**).

Conversely, if $\vec{x}=(a,\dots,a)$, then, by **(GAB2)**, **(AB2)** and Eq. (16), we have that $A^{a,b}_{\overrightarrow{G^{a,b}}}(\vec{x})=a$;

(OAB3) Suppose that $A^{a,b}_{\overrightarrow{G^{a,b}}}(\vec{x})=b$, for some $\vec{x}\in[a,b]^n$. Then, by Eq. (16) and **(PB*)**, we have that:

$G_j^{a,b}(\vec{x})=b$ for some $j\in\{1,\dots,m\}\Leftrightarrow x_i=b$

for some $i\in\{1,\dots,n\}$ by (**GAB3**).

On the other hand, if we take $\vec{x}\in[a,b]^n$, such that $\vec{x}=(x_1,\dots,x_i,\dots,x_n)$ with $x_i=b$ for some $i\in\{1,\dots,n\}$, then, by **(GAB3)**, **(PB*)** and Eq. (16), we have that $A^{a,b}_{\overrightarrow{G^{a,b}}}(\vec{x})=b$. □

Corollary 3.1. *Consider an m-ary (a,b)-grouping function $GC^{a,b}:[a,b]^m\to[a,b]$ and the tuple $\overrightarrow{G^{a,b}}=(G_1^{a,b},\dots,G_m^{a,b})$ of n-ary (a,b)-grouping functions. Then, the mapping $GC_{\overrightarrow{G^{a,b}}}:[a,b]^n\to[a,b]$, defined for all $\vec{x}\in[a,b]^n$, by*

$$GC_{\overrightarrow{G^{a,b}}}(\vec{x}) = GC^{a,b}(G_1^{a,b}(\vec{x}),\dots,G_m^{a,b}(\vec{x})),$$

is an n-ary (a,b)-grouping function.

Corollary 3.2. *Consider the weighted arithmetic mean $AW^{a,b}:[a,b]^m\to[a,b]$ given by Eq. (2), with $\vec{w}\in[0,1]^m$ such that $\sum_{i=1}^m w_i=1$ and the tuple $\overrightarrow{G^{a,b}}=(G_1^{a,b},\dots,G_m^{a,b})$ of n-ary (a,b)-grouping functions. Then, the mapping $AW_{\overrightarrow{G^{a,b}}}:[a,b]^n\to[a,b]$, defined for all $\vec{x}\in[a,b]^n$, by*

$$\begin{aligned} AW_{\overrightarrow{G^{a,b}}}(\vec{x}) &= AW^{a,b}(G_1^{a,b}(\vec{x}),\dots,G_m^{a,b}(\vec{x})) \\ &= G_1^{a,b}(\vec{x})\cdot w_1+\cdots+G_m^{a,b}(\vec{x})\cdot w_m, \end{aligned}$$

is an n-ary (a,b)-grouping function.

Remark 3.2. Notice that, by Corollary 3.1, one can state that the class of n-dimensional (a,b)-grouping functions is self closed with respect to the generalized composition, and, by Corollary 3.2, one can observe that the convex sum of n-dimensional (a,b)-grouping functions is also an n-dimensional (a,b)-grouping function. These properties are especially useful in practical applications, since one can combine different n-dimensional (a,b)-grouping functions to obtain new functions with the same behaviour. This is to be expected, since n-dimensional grouping functions (and their dual, n-dimensional overlap functions), for having analogous properties, have been the preferred choice as n-dimensional aggregation operators over the traditional t-conorms (and their dual, t-norms) on such applications [21,23,30,32].

Example 3.3.

(a) Consider the bivariate (a,b)-grouping function $G^{a,b}_{max}:[a,b]^2\to[a,b]$ given by $G^{a,b}_{max}(x,y)=\max\{x,y\}$. Moreover, take the (a,b)-grouping functions $G^{a,b}_{geom},G^{a,b}_{prod}:[a,b]^n\to[a,b]$ defined in Example 3.2. Then, the function $G^{a,b}_{max_{(G^{a,b}_{geom},G^{a,b}_{prod})}}:[a,b]^n\to[a,b]$, given by

$$G^{a,b}_{max_{(G^{a,b}_{geom},G^{a,b}_{prod})}}(\vec{x}) = G^{a,b}_{max}(G^{a,b}_{geom}(\vec{x}),G^{a,b}_{prod}(\vec{x})),$$

is an (a,b)-grouping function.

(b) Considering the same (a,b)-grouping functions $G^{a,b}_{geom},G^{a,b}_{prod}:[a,b]^n\to[a,b]$, then, the function $AM^{a,b}_{(G^{a,b}_{geom},G^{a,b}_{prod})}:[a,b]^n\to[a,b]$, given by

$$AM^{a,b}_{(G^{a,b}_{geom},G^{a,b}_{prod})}(\vec{x}) = \frac{G^{a,b}_{geom}(\vec{x})+G^{a,b}_{prod}(\vec{x})}{2},$$

is also an (a,b)-grouping function.

Theorem 3.3. *The function $G^{a,b}:[a,b]^n\to[a,b]$ is an (a,b)-grouping function if and only if*

$$G^{a,b}(x_1,\dots,x_n) = \frac{a\,f(x_1,\dots,x_n)+b\,g(x_1,\dots,x_n)}{f(x_1,\dots,x_n)+g(x_1,\dots,x_n)} \tag{17}$$

for some continuous symmetric functions $f,g:[a,b]^n\to[0,b-a]$ such that

- f is non-increasing and satisfies: $f(x_1,\dots,x_n)=0$ if and only if there exists $i\in\{1,\dots,n\}$ such that $x_i=b$;

- g is non-decreasing and satisfies: $g(x_1, \dots, x_n) = 0$ if and only if $x_i = a$ for all $i \in \{1, \dots, n\}$.

Proof. Observe that the function $G^{a,b}$ is well defined since, for all $x_1, \dots, x_n \in [a, b]$, $G^{a,b}(x_1, \dots, x_n) \in [a, b]$ and $f(x_1, \dots, x_n) + g(x_1, \dots, x_n) > 0$.

Necessity: Consider (a, b)-grouping function $G^{a,b}$, then the functions $f(x_1, \dots, x_n) = b - G^{a,b}(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n) = G^{a,b}(x_1, \dots, x_n) - a$ satisfy all the conditions imposed in theorem.

Sufficiency: Conditions $(GAB1)$ and $(GAB5)$ are straightforward. The condition $(GAB2)$ follows from the observation: $G^{a,b}(x_1, \dots, x_n) = a$ if and only if $g(x_1, \dots, x_n) = 0$; and the condition $(GAB3)$ follows from the observation: $G^{a,b}(x_1, \dots, x_n) = b$ if and only if $f(x_1, \dots, x_n) = 0$. Finally, the condition $(GAB4)$ can be proved by reformulation of Eq. (17) as follows:

$$G^{a,b}(x_1, \dots, x_n) = \frac{a\, f(x_1, \dots, x_n) + b\, g(x_1, \dots, x_n)}{f(x_1, \dots, x_n) + g(x_1, \dots, x_n)} = b - \frac{f(x_1, \dots, x_n)}{f(x_1, \dots, x_n) + g(x_1, \dots, x_n)} \cdot (b - a) \tag{18}$$

and the fact that f is non-increasing and g is non-decreasing. □

Example 3.4. The assumptions of Theorem 3.3 satisfy, for example, the functions:

$$f(x_1, \dots, x_n) = \frac{1}{(b-a)^{n-1}} \cdot \prod_{i=1}^{n} (b - x_i)$$

and

$$g(x_1, \dots, x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i - a,$$

hence we obtain the following (a, b)-grouping function:

$$G^{a,b}(x_1, \dots, x_n) = \frac{\frac{a}{(b-a)^{n-1}} \cdot \prod_{i=1}^{n} (b - x_i) + \frac{b}{n} \sum_{i=1}^{n} x_i - ab}{\frac{1}{(b-a)^{n-1}} \cdot \prod_{i=1}^{n} (b - x_i) + \frac{1}{n} \sum_{i=1}^{n} x_i - a}$$

4. Replacing the pooling operator of CNNs

4.1. (a, b)-grouping functions as pooling operator

When a pooling layer receives a feature matrix $X \in \mathbb{R}^{m \times n \times c}$ as input, it divides each of its c feature channels in disjoint windows $\vec{x} \in \mathbb{R}^{k_1 \times k_2}$ (or vectors of values $\vec{x} \in \mathbb{R}^{k_1 \cdot k_2}$), usually with $k_1 = k_2$ a small value. Afterwards, the values of each of those windows is replaced by a single representative computed through a function $F : \mathbb{R}^{k_1 \cdot k_2} \to \mathbb{R}$. The output of the layer, therefore, is a feature map $Y \in \mathbb{R}^{(m/k_1) \times (n/k_2) \times c}$, which effectively reduces the dimensionality of the features extracted by the convolutional layers of the CNN.

Although the first CNN models were presented using the arithmetic mean as pooling function [33], more complex models tend to default to maximum pooling [2,6], which usually yields better results in practice. In previous works, maximum pooling has been replaced by (a, b)-t-conorms, a family of functions to which the maximum belongs, obtaining favourable results [20]. This has motivated us to consider other families of functions, such as grouping functions, as potential replacement for the maximum.

One of the main drawbacks of the maximum is the fact that it ignores the information from all values to be aggregated except for one of them. On the one hand, this means that much of the information provided by the neighbour values is discarded, which could be of interest. On the other hand, the derivative of $\max(\vec{x})$ with respect to x_i is 0 whenever there exists a value $x_j \in \vec{x}$ such that $x_j > x_i$. If we take into account the fact that during training time, parameters are updated according to the derivative of the loss function with respect to them, then the gradient will not flow through those values which may slow the training of the model.

However, preserving high activations offers good empirical results, which seems to justify the popularity of max pooling with respect to average pooling. When combined with the ReLU activation function, e. i. $f(x) = \max\{0, x\}$, it makes sense to prioritize high values, since they reflect the highest presence of a given feature on a feature image.

(a, b)-grouping functions preserve the positive behaviour of maximum pooling while addressing its main drawback. Notice that they are defined in a closed interval $[a, b] \subset \mathbb{R}$ instead of the whole real line, so we will set $a = \min(X)$ and $b = \max(X)$. In this way, peak activation values would be preserved after applying the pooling function, as ensured by property **(GAB3)** of Definition 3.1, while taking into consideration the value of other elements of the pooling window.

We will test the suitability of (a, b)-grouping function based pooling in Section 5.[1]

4.2. Related work

A number of alternative pooling methods have been proposed over the last few years. We review several of the most notorious ones here, and will compare our results against some of them:

- **"Mixed" pooling**
  Introduced in [17], "mixed" pooling presents the idea of combining both maximum and average pooling by means of a convex combination of both reductions. The authors propose the addition of a learnable α coefficient which can vary on number, depending on if a different coefficient is learnt for each feature channel or patch of the input. The authors find that "mixed" pooling outperforms both maximum and average pooling individually.
- **"Gated" pooling**
  Introduced in tandem with "mixed" pooling, gated pooling differs with respect to the strategy to compute the α coefficients of the combination. Instead of directly learning α, a set of weights of size $k \times k$, with k being the size of the pooling window, are learnt. Then, prior to aggregating the values of each window, a linear transformation is applied to those same values, using the previous weights. The output is treated as the α coefficient of the combination. Although it implies an increase in the number of necessary parameters for the model, results obtained with this strategy improve with respect to the simple "mixed" strategy.
- **"Attention" pooling**
  In [34], the authors introduce "attention-based pooling" in the context of an aerial image scene classification problem, as a mean to capture the semantic information of the data being pooled. This pooling layer uses a 1×1 convolution layer in order to compute an "attention" matrix whose values represent the relevance of a given pixel according to all feature maps. The values of each pooled region are then weighted by the values of that "attention" matrix, so that the final pooling layer acts as a learnable weighted mean pooling.
- **Deep Generalized Max pooling**
  This method was presented in [35] as an adaptation of the ideas of [36] to Deep Neural Networks. Unlike previous pooling operators, this layer tries to generalize the Global Average pooling layer common in many modern CNNs [37], which downsamples all values of each feature image after the last convolution layer of a CNN into a single representative value. It tries to utilize the information of small patches of the image in the global pooling process, balancing the frequency of frequent and rare features. To this end, a system of linear equations is constructed with as many unknowns as feature channels, and its solution acts as the output of the layer.

[1] Code is available at https://github.com/iosurodri/overlapsAndGroupings.

- **"Strip" pooling**
  Designed for working with images on real scene parsing in [38], strip pooling replaces the classic square pooling window of traditional methods for thin windows which span all the columns/rows of the image. The main objective of this strategy is to capture the anisotropy context common in real world images, rather than reducing the size of images. In particular, for each position of a feature channel, a horizontal stripe spanning all the columns of the image and a vertical one spanning the rows are used to extract two vectors of values. A 1D convolutional layer is applied to each of those filters, whose outputs are later expanded and combined in order to create a more informed new representative for the value.

5. Experimental evaluation

5.1. Experimental framework

5.1.1. CNN models

We have tested grouping pooling in three different classic CNN models of varying "depth" and parameter count. Our intention is to empirically study if the replacement of classic pooling operators by (a, b)-grouping functions is similar in all cases, or these factors influence the behaviour of the new operators. The employed models have been the following ones.

1. **LeNet-5**: Presented in [33], this model presents the classic structure of vanilla CNNs. It employs two sets of convolutional and pooling layers as feature extractor followed by a 3 layer MLP which takes charge of the classification task. Batch Normalization layers [39] have been inserted after pooling layers and each hidden layer of the MLP, and we use the ReLU function as activation function.
2. **VGG16**: This architecture proposed in [6] can be understood as a deeper, more sophisticated version of the first model. While it is still composed of a feature extractor based on convolutions and pooling operations, and a MLP classifier with 3 hidden layers, the feature extractor has 13 convolution layers. Between each reduction step performed by a pooling layer with kernel size 2×2, several convolution layers are applied, all of them composed by filters of size 3×3. In order to capture more discriminative features, after performing a pooling operation, the number of filters of successive convolution layers is doubled. Furthermore, in order to ease the training of such a deep model, both Dropout [40] and Batch Normalization layers are applied after each convolution layer.
3. **ResNet-56**: Introduced in [7], ResNet models have become an staple CNN architecture. They receive their name from the concept of "residual" connections, a mechanism which adds the values of the input of a layer to its produced activations. This strategy improves the gradient flow during backpropagation, since the partial derivative of each convolution layer with respect to input activations is added a value of 1. Thus, ResNets can incorporate tens of layers without incurring in vanishing gradient problems. In our experiments, we work with a model of 56 layers with reduced kernel filters of size 3×3, to account for the fact that we will be working with small size images. We have also replaced the convolution layers which took care of the image downsampling process at the end of each step by pooling layers, so that our strategy could be tested.

We follow a similar training regime for all models: They have been trained along 100 epochs using Stochastic Gradient descent with momentum, with initial learning rates of 0.001 for LeNet-5, 0.01 for VGG16, and 0.1 for ResNet. We control the evolution of the learning rate through Cosine Annealing [41], progressively reducing it from its initial value to 0 on the last epoch.

5.1.2. Datasets

- **MNIST**: Introduced in [33], it consists of gray-scale images representing handwritten numerical digits, which must be classified in 10 classes. Images are of size 28×28 and are divided in a 60,000 samples partition and a 10,000 test one.
- **CIFAR-10/CIFAR-100**: Presented in [42], both datasets share the same 32×32 RGB images representing animals and objects of the real world, as well as the same 50,000 train partition and 10,000 test partition. Classes are completely balanced. Both versions of the dataset differ with respect of the number of classes in which they need be classified, which are 10 or 100 respectively. The same simple data augmentation procedure employed papers such as [37,43] have been used. It consists of padding 4 rows and columns on all sides of the images, taking a random crop of size 32×32 and randomly flipping them horizontally with a probability of 0.5.

Additionally, data sharing is not applicable to this article as no own datasets were generated or analysed during the current study.

5.2. Experiments and results

We have organized the different experiments according to the pooling function used. For each experiment, we have substituted the pooling function of each of the presented CNN models by a different (a, b)-grouping function. We have tested examples of functions obtained through each of the construction methods presented in Section 3.1.

In particular, we refer to functions constructed using Theorem 3.1 as "individual" functions. Corollary 3.2 presents a method for constructing (a, b)-grouping functions through a convex combination of any other (a, b)-grouping function. Similarly, Corollary 3.1 shows a method for the construction of (a, b)-grouping functions as the composition of a set of (a, b)-grouping functions by another (a, b)-grouping function. We have set a series of individual functions which are useful to construct new ones. All the considered functions are presented in Table 2.

Table 3 shows the obtained results for these initial tests. We report the mean and standard deviation of 5 independent runs for each model. We include tests using the average since it is another standard pooling operator.

In the case of the MNIST dataset, most models achieve an accuracy above 0.99, with very slight changes between them. Therefore, results on that dataset are not useful for analysis and we have ignored MNIST for the remaining tests.

In the case of CIFAR-10 and CIFAR-100, groupings $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$ offer well all-around results in most cases, with the latter one performing best. Although their results do not differ too much from results obtained using average or maximum pooling (first two rows on Table 3), they are resilient to changes on the base model, which makes them more reliable to apply in any situation.

(a, b)-groupings constructed through the combination of (a, b)-grouping functions also offer comparable results, specially when combining the maximum and groupings $G^{a,b}_{prod}$ or $G^{a,b}_{ob}$, with some instances offering the best all-around results. The same occurs with (a, b)-grouping compositions. However, we consider that results do not justify their use above individual pooling operators, which are simpler and easier to implement.

Finally, in Table 4 we compare our proposed pooling functions with three notable pooling operators: mixed, gated and attention pooling. Notice how the best (a, b)-grouping functions perform on par with all those alternative pooling layers, while introducing no additional parameters to the model and having a more straight-forward implementation. Interestingly, attention pooling offers poor results for the VGG16 model, which may suggest a difficulty for training the parameters needed for the attention mechanism.

Table 2
(a, b)-grouping functions used as pooling operator candidates. Presented divided according to the construction method used for their definition.

	Name	Function
Individual groupings	$G^{a,b}_{max}$	$G_{max}(\mathbf{x}) = \max_{i=1}^{n} x_i$
	$G^{a,b}_{prod}$	$G_{prod}(\mathbf{x}) = 1 - \prod_{i=1}^{n}(1 - x_i)$
	$G^{a,b}_{geom}$	$G_{geom}(\mathbf{x}) = 1 - \sqrt[n]{\prod_{i=1}^{n}(1 - x_i)}$
	$G^{a,b}_{ob}$	$G_{ob}(\mathbf{x}) = 1 - \sqrt{\min_{i=1}^{n}(1 - x_i) \cdot \prod_{i=1}^{n}(1 - x_i)}$
	$G^{a,b}_{u}$	$G_{u}(\mathbf{x}) = \frac{\max_{i=1}^{n} x_i}{\max_{i=1}^{n} x_i + \sqrt[n]{\prod_{i=1}^{n}(1-x_i)}}$
Convex combinations	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob})}$	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob})}(\mathbf{x}) = w_1 G^{a,b}_{max}(\mathbf{x}) + w_2 G^{a,b}_{ob}(\mathbf{x})$
	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{prod})}$	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{prod})}(\mathbf{x}) = w_1 G^{a,b}_{max}(\mathbf{x}) + w_2 G^{a,b}_{prod}(\mathbf{x})$
	$AW^{a,b}_{(G^{a,b}_{prod}, G^{a,b}_{ob})}$	$AW^{a,b}_{(G^{a,b}_{prod}, G^{a,b}_{ob})}(\mathbf{x}) = w_1 G^{a,b}_{prod}(\mathbf{x}) + w_2 G^{a,b}_{ob}(\mathbf{x})$
	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob}, G^{a,b}_{prod})}$	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob}, G^{a,b}_{prod})}(\mathbf{x}) = w_1 G^{a,b}_{max}(\mathbf{x}) + w_2 G^{a,b}_{ob}(\mathbf{x}) + w_3 G^{a,b}_{prod}(\mathbf{x})$
Grouping compositions	$G^{a,b}_{max_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}$	$G^{a,b}_{max_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}(\mathbf{x}) = G^{a,b}_{max}(G^{a,b}_{prod}(\mathbf{x}), G^{a,b}_{ob}(\mathbf{x}))$
	$G^{a,b}_{ob_{(G^{a,b}_{max}, G^{a,b}_{prod})}}$	$G^{a,b}_{ob_{(G^{a,b}_{max}, G^{a,b}_{prod})}}(\mathbf{x}) = G^{a,b}_{ob}(G^{a,b}_{max}(\mathbf{x}), G^{a,b}_{prod}(\mathbf{x}))$
	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{ob})}}$	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{ob})}}(\mathbf{x}) = G^{a,b}_{prod}(G^{a,b}_{max}(\mathbf{x}), G^{a,b}_{ob}(\mathbf{x}))$
	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{prod})}}$	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{prod})}}(\mathbf{x}) = G^{a,b}_{prod}(G^{a,b}_{max}(\mathbf{x}), G^{a,b}_{prod}(\mathbf{x}))$
	$G^{a,b}_{prod_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}$	$G^{a,b}_{prod_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}(\mathbf{x}) = G^{a,b}_{prod}(G^{a,b}_{prod}(\mathbf{x}), G^{a,b}_{ob}(\mathbf{x}))$

Table 3
Accuracy rate obtained for pooling operators based on different (a, b)-grouping construction methods for the datasets MNIST, CIFAR10 and CIFAR100. In the case of CIFAR-100, top-5 accuracy has also been calculated, and is presented to the right of its top-1 accuracy value.

		MNIST			CIFAR-10			CIFAR-100		
		LeNet-5	VGG16	ResNet	LeNet-5	VGG16	ResNet	LeNet-5	VGG16	ResNet
Individual	Avg	**0.992 ± 0.000**	**0.996 ± 0.000**	**0.996 ± 0.001**	0.825 ± 0.003	**0.915 ± 0.001**	**0.919 ± 0.004**	0.558 ± 0.002/0.826 ± 0.001	**0.682 ± 0.002/0.891 ± 0.004**	0.681 ± 0.007/0.902 ± 0.005
	Max	**0.992 ± 0.001**	**0.996 ± 0.000**	**0.996 ± 0.001**	0.837 ± 0.003	0.911 ± 0.003	**0.919 ± 0.003**	0.561 ± 0.000/0.832 ± 0.003	0.676 ± 0.003/0.888 ± 0.004	0.681 ± 0.005/0.898 ± 0.004
	$G^{a,b}_{prod}$	**0.992 ± 0.000**	**0.996 ± 0.001**	**0.996 ± 0.000**	**0.839 ± 0.003**	0.912 ± 0.003	0.918 ± 0.004	0.557 ± 0.000/0.827 ± 0.006	0.678 ± 0.004/0.889 ± 0.004	0.664 ± 0.014/0.891 ± 0.010
	$G^{a,b}_{ob}$	**0.992 ± 0.000**	**0.996 ± 0.000**	**0.996 ± 0.001**	0.830 ± 0.003	**0.915 ± 0.002**	0.918 ± 0.002	**0.562 ± 0.003/0.834 ± 0.002**	0.680 ± 0.001/0.891 ± 0.003	**0.684 ± 0.018/0.902 ± 0.004**
	$G^{a,b}_{geom}$	**0.992 ± 0.000**	**0.996 ± 0.001**	**0.996 ± 0.000**	0.824 ± 0.004	0.906 ± 0.010	0.916 ± 0.009	0.466 ± 0.019/0.752 ± 0.022	0.639 ± 0.044/0.864 ± 0.026	0.623 ± 0.046/0.859 ± 0.047
	$G^{a,b}_{u}$	0.991 ± 0.001	**0.996 ± 0.001**	0.995 ± 0.000	0.825 ± 0.002	0.911 ± 0.004	0.917 ± 0.005	0.559 ± 0.013/0.829 ± 0.007	0.650 ± 0.037/0.871 ± 0.027	0.653 ± 0.031/0.887 ± 0.032
Combinations	$AW^{a,b}_{(G^{a,b}_{prod}, G^{a,b}_{ob})}$	**0.992 ± 0.000**	**0.996 ± 0.001**	0.995 ± 0.001	0.829 ± 0.002	0.914 ± 0.002	0.914 ± 0.008	0.561 ± 0.001/0.833 ± 0.004	0.679 ± 0.002/0.890 ± 0.001	**0.674 ± 0.016/0.898 ± 0.009**
	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{prod})}$	**0.992 ± 0.001**	**0.996 ± 0.000**	**0.996 ± 0.000**	0.828 ± 0.003	0.913 ± 0.002	0.920 ± 0.007	0.565 ± 0.004/0.833 ± 0.002	0.679 ± 0.002/0.890 ± 0.003	0.622 ± 0.051/0.863 ± 0.052
	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob})}$	0.991 ± 0.001	**0.996 ± 0.000**	**0.996 ± 0.001**	**0.831 ± 0.002**	0.914 ± 0.001	**0.923 ± 0.001**	**0.567 ± 0.002/0.833 ± 0.001**	0.679 ± 0.004/0.891 ± 0.002	0.671 ± 0.007/0.898 ± 0.005
	$AW^{a,b}_{(G^{a,b}_{max}, G^{a,b}_{ob}, G^{a,b}_{prod})}$	**0.992 ± 0.001**	**0.996 ± 0.000**	0.995 ± 0.001	0.828 ± 0.002	**0.916 ± 0.002**	0.922 ± 0.002	0.563 ± 0.005/0.832 ± 0.001	**0.681 ± 0.005/0.888 ± 0.002**	**0.674 ± 0.006/0.898 ± 0.002**
Compositions	$G^{a,b}_{max_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}$	**0.992 ± 0.000**	**0.996 ± 0.000**	**0.996 ± 0.000**	0.823 ± 0.006	0.913 ± 0.001	**0.919 ± 0.004**	0.561 ± 0.004/0.831 ± 0.002	0.678 ± 0.003/0.888 ± 0.002	0.665 ± 0.019/0.890 ± 0.020
	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{prod})}}$	**0.992 ± 0.001**	**0.996 ± 0.000**	**0.996 ± 0.000**	**0.824 ± 0.002**	0.911 ± 0.003	0.891 ± 0.038	0.561 ± 0.003/0.831 ± 0.001	0.679 ± 0.005/0.890 ± 0.002	0.609 ± 0.093/0.894 ± 0.006
	$G^{a,b}_{prod_{(G^{a,b}_{max}, G^{a,b}_{ob})}}$	**0.992 ± 0.001**	**0.996 ± 0.001**	0.995 ± 0.001	0.824 ± 0.005	**0.914 ± 0.001**	0.900 ± 0.016	0.565 ± 0.002/0.834 ± 0.001	**0.681 ± 0.002/0.889 ± 0.001**	**0.669 ± 0.027/0.894 ± 0.006**
	$G^{a,b}_{prod_{(G^{a,b}_{prod}, G^{a,b}_{ob})}}$	**0.992 ± 0.001**	**0.996 ± 0.000**	0.995 ± 0.001	0.820 ± 0.004	0.910 ± 0.001	0.860 ± 0.066	0.562 ± 0.004/0.830 ± 0.002	0.671 ± 0.006/0.885 ± 0.003	0.456 ± 0.119/0.728 ± 0.113
	$G^{a,b}_{ob_{(G^{a,b}_{max}, G^{a,b}_{prod})}}$	**0.992 ± 0.000**	**0.996 ± 0.000**	0.995 ± 0.001	0.823 ± 0.002	0.912 ± 0.001	0.908 ± 0.013	**0.567 ± 0.001/0.834 ± 0.003**	0.677 ± 0.003/0.889 ± 0.002	0.683 ± 0.040/0.898 ± 0.004

Table 4
Comparison with modern pooling operators "mixed", "gated" and "attention". Notice how (a, b)-grouping functions perform in par with most of them, while avoiding the inclusion of additional parameters to the base CNN architecture.

	CIFAR-10			CIFAR-100		
	LeNet-5	VGG16	ResNet	LeNet-5	VGG16	ResNet
Avg	0.825 ± 0.003	0.915 ± 0.001	0.919 ± 0.004	0.558 ± 0.002/0.826 ± 0.001	0.682 ± 0.002/0.891 ± 0.004	0.681 ± 0.007/0.902 ± 0.005
Max	0.837 ± 0.003	0.911 ± 0.003	0.919 ± 0.003	0.561 ± 0.000/0.832 ± 0.003	0.676 ± 0.003/0.888 ± 0.004	0.681 ± 0.005/0.898 ± 0.004
Best grouping	0.839 ± 0.003	**0.916 ± 0.002**	**0.923 ± 0.001**	0.567 ± 0.001/0.834 ± 0.003	0.681 ± 0.002/0.889 ± 0.001	0.684 ± 0.018/0.902 ± 0.004
Mixed pooling	**0.842 ± 0.001**	**0.916 ± 0.002**	0.922 ± 0.002	0.561 ± 0.002/0.830 ± 0.001	**0.683 ± 0.002/0.892 ± 0.002**	0.680 ± 0.002/0.901 ± 0.001
Gated pooling	**0.842 ± 0.003**	0.913 ± 0.003	0.922 ± 0.002	**0.572 ± 0.004/0.836 ± 0.001**	0.682 ± 0.003/0.892 ± 0.001	0.686 ± 0.003/0.901 ± 0.003
Attention pooling	0.836 ± 0.002	0.884 ± 0.008	**0.923 ± 0.003**	0.563 ± 0.003/0.830 ± 0.003	0.614 ± 0.006/0.850 ± 0.008	**0.681 ± 0.005/0.903 ± 0.004**

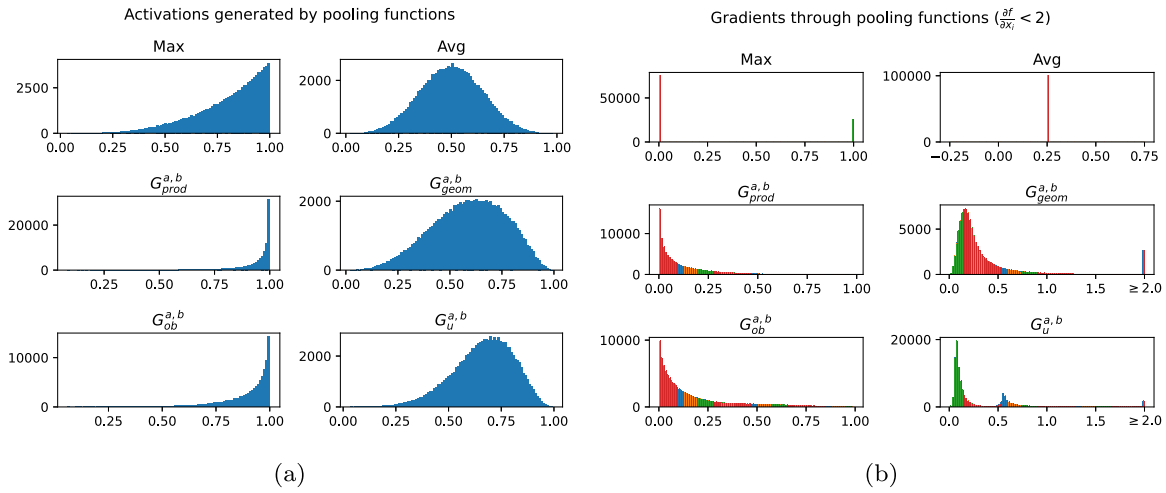

Fig. 1. (a) Distribution of activations generated for 10,000 random tuples of 4 values sampled from a uniform distribution on $[0, 1]$. Notice how $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$ act as a more extreme version of max pooling, while $G^{a,b}_{geom}$ and $G^{a,b}_{u}$ output slightly higher values than average pooling. (b) Gradients generated during backward pass for those same activations. Unlike max pooling, $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$ return non-zero gradients for values other than the maximum, which may benefit the gradient flow of the model. By contrast, notice how for grouping functions $G^{a,b}_{geom}$ and $G^{a,b}_{u}$ gradients have been clipped to an upmost value of 2. Very high outliers can be generated on those functions, leading to exploding gradient problems and difficulties on training, which may explain their subpar behaviour.

Table 5
Results offered by combinations of (a, b)-grouping functions with the arithmetic mean. Mixed pooling corresponds to the combination of average pooling and maximum pooling.

	CIFAR-10			CIFAR-100		
	LeNet-5	VGG16	ResNet	LeNet-5	VGG16	ResNet
Avg	0.825 ± 0.003	0.915 ± 0.001	0.919 ± 0.004	$0.558 \pm 0.002 / 0.826 \pm 0.001$	$0.682 \pm 0.002 / 0.891 \pm 0.004$	$0.681 \pm 0.007 / 0.902 \pm 0.005$
Max	0.837 ± 0.003	0.911 ± 0.003	0.919 ± 0.003	$0.561 \pm 0.000 / 0.832 \pm 0.003$	$0.676 \pm 0.003 / 0.888 \pm 0.004$	$0.681 \pm 0.005 / 0.898 \pm 0.004$
Best grouping	0.839 ± 0.003	$\mathbf{0.916 \pm 0.002}$	$\mathbf{0.923 \pm 0.001}$	$\mathbf{0.567 \pm 0.001 / 0.834 \pm 0.003}$	$0.681 \pm 0.002 / 0.889 \pm 0.001$	$0.684 \pm 0.018 / 0.902 \pm 0.004$
Mixed pooling	$\mathbf{0.842 \pm 0.001}$	$\mathbf{0.916 \pm 0.002}$	0.922 ± 0.002	$0.561 \pm 0.002 / 0.830 \pm 0.001$	$0.683 \pm 0.002 / 0.892 \pm 0.002$	$0.680 \pm 0.002 / 0.901 \pm 0.001$
$AW^{a,b}_{(Avg, G^{a,b}_{ob})}$	0.841 ± 0.001	0.914 ± 0.001	0.921 ± 0.002	$0.561 \pm 0.001 / 0.831 \pm 0.001$	$\mathbf{0.681 \pm 0.001 / 0.893 \pm 0.001}$	$\mathbf{0.684 \pm 0.002 / 0.904 \pm 0.005}$
$AW^{a,b}_{(Avg, G^{a,b}_{prod})}$	0.837 ± 0.002	0.915 ± 0.001	$\mathbf{0.923 \pm 0.002}$	$0.560 \pm 0.004 / 0.830 \pm 0.001$	$0.681 \pm 0.003 / 0.892 \pm 0.001$	$0.677 \pm 0.012 / 0.900 \pm 0.006$

5.3. Feature visualization

In order to further understand the impact of (a, b)-grouping functions as pooling operator, we have studied the activation and gradient values they produce, both during forward pass and backward pass. In Fig. 1, we show the results obtained through a simulation in which 10,000 random tuples of 4 values sampled from a random uniform distribution on $[0, 1]$ have been reduced through several different pooling functions. We have used max pooling, average pooling, and the individual (a, b)-grouping functions. Interestingly, while $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$ act as a more extreme case of max pooling, with a higher proportion of high activitations being generated, $G^{a,b}_{geom}$ and $G^{a,b}_{u}$ behave similarly to average pooling, with a skew with respect to higher activations.

After generating the resulting activations, the gradients of the output of each pooling function with respect to its input have also been graphed. An important detail of max pooling is the fact that it outputs a gradient of zero for non-maximal elements of a pooling window, blocking the flow of the gradient through those values. By contrast, $G^{a,b}_{prod}$ and specially $G^{a,b}_{ob}$ return non-zero gradients for plenty of elements, which may explain its good performance. In the case of $G^{a,b}_{geom}$ and $G^{a,b}_{u}$, gradients have been clipped to a value of 2, but very high outlier values could be generated, producing exploding gradient problems. Therefore, we suggest avoiding their use.

In order to further visualize the effect of $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$ functions, in Fig. 2, three feature maps extracted from the first block of our ResNet-56 model have been reduced with both functions, together with maximum and average pooling. While both maximum and average pooling output fairly similar feature maps, (a, b)-grouping functions accentuate high intensity activations, amplifying their result for following layers, with $G^{a,b}_{prod}$ acting in a more extreme way.

5.4. Combinations of (a, b)-grouping functions with the arithmetic mean

As a final study, motivated by the good results offered by both average and mixed pooling layers, we also study the possibility of combining the arithmetic mean with (a, b)-grouping functions. Table 5 summarize the accuracy rates offered by the different methods.

Once again, although good results are obtained, they do not differ much from the ones offered by $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$. The same happens with mixed pooling, which in this case does outperform maximum pooling in most situations.

A similar study to that of Fig. 1 has been performed in this case, which we present in Fig. 3. If we compare distributions generated by combinations with the (a, b)-grouping functions with the ones outputted by their individual versions, we find that their activation distributions behaves as a skewed version of average pooling. However, the gradients of both functions retain the distribution of their individual counterparts. On the contrary, mixed pooling solves the problem on gradient flow of max pooling, with previous gradient values of zero having been replaced by a factor of the derivative of average pooling. We believe that this is the case for mixed pooling outperforming max pooling, similarly to (a, b)-grouping functions.

6. Conclusions and future work

In this paper we have proven that better alternatives to the maximum as a pooling function exist. In particular, (a, b)-grouping functions

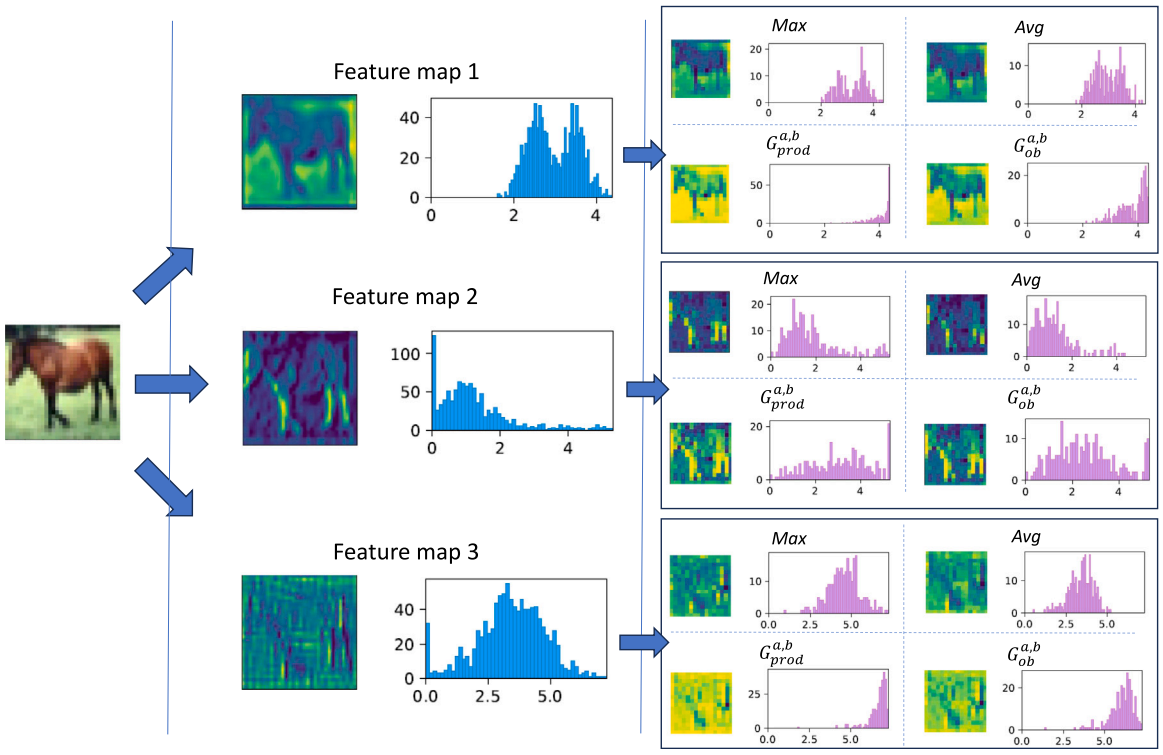

Fig. 2. Comparison on the effect performed by four different pooling functions among three different feature maps. Namely, we compare max pooling (Max), average pooling (Avg) and the (a, b)-grouping functions $G^{a,b}_{prod}$ and $G^{a,b}_{ob}$. The grouping functions present a distinctive behaviour which produces high activation values, with $G^{a,b}_{prod}$ being the most extreme of the pair.

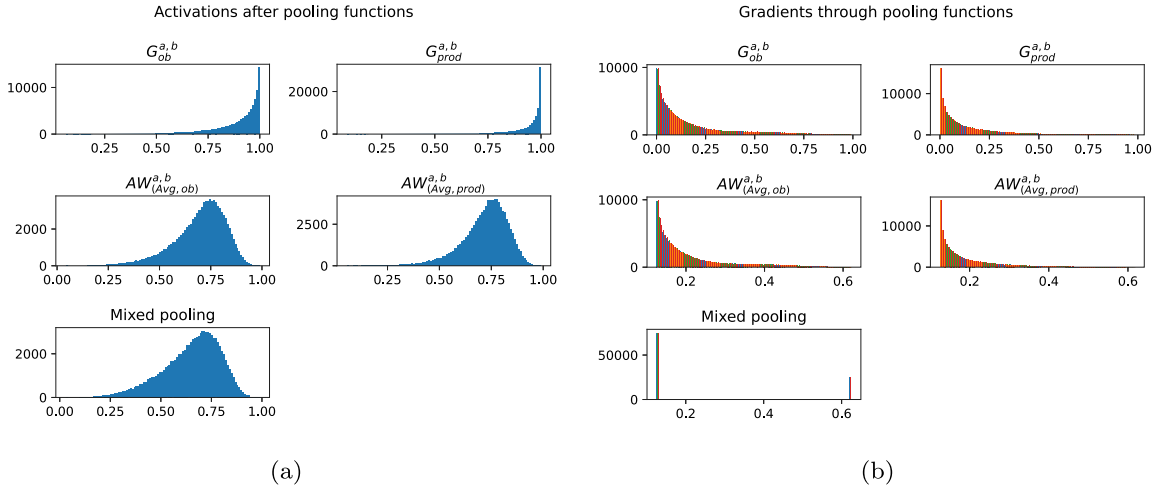

Fig. 3. (a) Distribution of activations generated for $10,000$ random tuples of 4 values sampled from a uniform distribution on $[0, 1]$ when using combinations of functions with the arithmetic mean. "Mixed" pooling equals to the combination of average and max pooling. As expected, activations are fairly similar among all combinations. (b) Gradients generated during backward pass for those same activations. Unlike the gradient of max pooling in 1(b) which generated loads of zeros, "mixed" pooling returns non-zero values for those same reductions, which may justify its improved results. Combinations with other grouping functions are very similar to their individual versions.

share some of the same interesting properties, while taking into consideration more of the available information in the feature images. However, we have also seen that some expressions must be avoided since they may incur in exploding gradient problems, and that analysing their differentiability is key.

Additionally, we have shown that several of these functions result competitive with modern pooling operators such as "mixed", "gated" and "attention" pooling, without the expense of including additional parameters to the model. Further, they can be applied to different architectures of different complexities, without additional considerations, unlike "attention" pooling.

In the future, we would like to explore other feature aggregation processes such as Global Average Pooling layers and bottleneck convolutions, and present alternatives which ease the discrimination among features.

CRediT authorship contribution statement

Iosu Rodriguez-Martinez: Investigation, Methodology, Software, Writing – original draft. **Tiago da Cruz Asmus:** Conceptualization, Formal analysis. **Graçaliz Pereira Dimuro:** Supervision, Formal analysis, Writing – review & editing. **Francisco Herrera:** Validation, Writing –

review & editing. **Zdenko Takáč:** Validation, Writing – review & editing. **Humberto Bustince:** Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code has been made available through github. A link is provided inside the paper's body

Acknowledgements

The authors gratefully acknowledge the financial support of Tracasa Instrumental (iTRACASA) and of the Gobierno de Navarra - Departamento de Universidad, Innovación y Transformación Digital, as well as that of the Spanish Ministry of Science (project PID2019-108392GB-I00 (AEI/10.13039/501100011033)) and the project PC095-096 FUSIPROD. T. Asmus and G.P. Dimuro are supported by the projects CNPq (301618/2019-4) and FAPERGS (19/2551-0001279-9). F. Herrera is supported by the Andalusian Excellence project P18-FR-4961. Z. Takáč is supported by grant VEGA 1/0267/21. Open access funding provided by Universidad Pública de Navarra

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[2] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al., Highly accurate protein structure prediction with alphafold, Nature 596 (7873) (2021) 583–589.
[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[5] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, Hierarchical text-conditional image generation with clip latents, 2022, arXiv preprint arXiv:2204.06125.
[6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.
[7] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[8] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[9] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder–decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495.
[10] J. Pan, D. Sun, J. Zhang, J. Tang, J. Yang, Y.-W. Tai, M.-H. Yang, Dual convolutional neural networks for low-level vision, Int. J. Comput. Vis. 130 (6) (2022) 1440–1458.
[11] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, M.-H. Yang, Depth-aware video frame interpolation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3703–3712.
[12] Y. Zhang, Y. Liu, P. Sun, H. Yan, X. Zhao, L. Zhang, Ifcnn: A general image fusion framework based on convolutional neural network, Inf. Fusion 54 (2020) 99–118.
[13] Y. Cao, C. Wang, Z. Li, L. Zhang, L. Zhang, Spatial-bag-of-features, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3352–3359.
[14] Y.-L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning mid-level features for recognition, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2559–2566.
[15] Y.-L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the 27th International Conference on Machine Learning, ICML-10, 2010, pp. 111–118.
[16] M. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May (2013) 2-4, Conference Track Proceedings, 2013.
[17] C.-Y. Lee, P. Gallagher, Z. Tu, Generalizing pooling functions in CNNS: Mixed, gated, and tree, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 863–875, http://dx.doi.org/10.1109/TPAMI.2017.2703082.
[18] J.I. Forcen, M. Pagola, E. Barrenechea, H. Bustince, Learning ordered pooling weights in image classification, Neurocomputing 411 (2020) 45–53.
[19] I. Rodriguez-Martinez, J. Lafuente, R.H. Santiago, G.P. Dimuro, F. Herrera, H. Bustince, Replacing pooling functions in convolutional neural networks by linear combinations of increasing functions, Neural Netw. 152 (2022) 380–393.
[20] T.d.C. Asmus, G.P. Dimuro, B. Bedregal, J.A. Sanz, J. Fernandez, I. Rodriguez-Martinez, R. Mesiar, H. Bustince, A construtive framework to define fusion functions with floating domains in arbitrary closed real intervals, Inform. Sci. 610 (2022) 800–829.
[21] H. Bustince, M. Pagola, R. Mesiar, E. Hullermeier, F. Herrera, Grouping, overlap, and generalized bientropic functions for fuzzy modeling of pairwise comparisons, IEEE Trans. Fuzzy Syst. 20 (3) (2011) 405–415.
[22] B. Bedregal, G.P. Dimuro, H. Bustince, E. Barrenechea, New results on overlap and grouping functions, Inform. Sci. 249 (2013) 148–170.
[23] A. Jurio, H. Bustince, M. Pagola, A. Pradera, R.R. Yager, Some properties of overlap and grouping functions and their application to image thresholding, Fuzzy Sets and Systems 229 (2013) 69–90.
[24] R.R. Yager, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Trans. Syst. Man Cybern. 18 (1) (1988) 183–190.
[25] M. Papčo, I. Rodríguez-Martínez, J. Fumanal-Idocin, A.H. Altalhi, H. Bustince, A fusion method for multi-valued data, Inf. Fusion 71 (2021) 1–10.
[26] E.P. Klement, R. Mesiar, E. Pap, Triangular norms, Vol. 8, Springer Science & Business Media, 2013.
[27] R. Mesiar, A. Kolesárová, H. Bustince, G.P. Dimuro, B. Bedregal, Fusion functions based discrete choquet-like integrals, European J. Oper. Res. 252 (2) (2016) 601–609.
[28] G. Beliakov, H.B. Sola, T.C. Sánchez, A Practical Guide to Averaging Functions, Springer, 2016.
[29] D. Gómez, J.T. Rodriguez, J. Montero, H. Bustince, E. Barrenechea, N-dimensional overlap functions, Fuzzy Sets and Systems 287 (2016) 57–75.
[30] M. Elkano, M. Galar, J.A. Sanz, A. Fernández, E. Barrenechea, F. Herrera, H. Bustince, Enhancing multiclass classification in farc-hd fuzzy classifier: On the synergy between n-dimensional overlap functions and decomposition strategies, IEEE Trans. Fuzzy Syst. 23 (5) (2014) 1562–1580.
[31] H. Bustince, J. Fernandez, R. Mesiar, J. Montero, R. Orduna, Overlap functions, Nonlinear Anal. TMA 72 (3–4) (2010) 1488–1499.
[32] T. Batista, B. Bedregal, R. Moraes, Constructing multi-layer classifier ensembles using the choquet integral based on overlap and quasi-overlap functions, Neurocomputing (2022).
[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[34] Q. Bi, K. Qin, H. Zhang, J. Xie, Z. Li, K. Xu, Apdc-net: Attention pooling-based convolutional network for aerial scene classification, IEEE Geosci. Remote Sens. Lett. 17 (9) (2019) 1603–1607.
[35] V. Christlein, L. Spranger, M. Seuret, A. Nicolaou, P. Král, A. Maier, Deep generalized max pooling, in: 2019 International Conference on Document Analysis and Recognition, ICDAR, IEEE, 2019, pp. 1090–1096.
[36] N. Murray, F. Perronnin, Generalized max pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2473–2480.
[37] M. Lin, Q. Chen, S. Yan, Network in network, in: International Conference on Learning Representations, ICLR, 2014.
[38] Q. Hou, L. Zhang, M.-M. Cheng, J. Feng, Strip pooling: Rethinking spatial pooling for scene parsing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4003–4012.
[39] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[41] I. Loshchilov, F. Hutter, Sgdr: Stochastic gradient descent with warm restarts, in: International Conference on Learning Representations, ICLR 2017, 2017.
[42] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Tech. Rep. 0, University of Toronto, Toronto, Ontario, 2009.
[43] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Artificial Intelligence and Statistics, PMLR, 2015, pp. 562–570.