Deep neural network to extract high-level features and labels in multi-label classification problems

Peer-reviewed author version

BELLO GARCIA, Marilyn; NAPOLES RUIZ, Gonzalo; Sánchez, Ricardo; Bello, Rafael & VANHOOF, Koen (2020) Deep neural network to extract high-level features and labels in multi-label classification problems. In: Neurocomputing (Amsterdam), 413 , p. 259 -270.

DOI: 10.1016/j.neucom.2020.06.117
Handle: http://hdl.handle.net/1942/32436

Deep neural network to extract high-level features and labels in multi-label classification problems

Marilyn Bello[a,c,*], Gonzalo Nápoles[a,b], Ricardo Sánchez[d], Rafael Bello[c], Koen Vanhoof[a]

[a] *Faculty of Business Economics, Universiteit Hasselt, Belgium*
[b] *Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands*
[c] *Department of Computer Science, Central University of Las Villas, Cuba*
[d] *Informatization Department, Central University of Las Villas, Cuba*

Abstract

Pooling layers help reduce redundancy and the number of parameters in deep neural networks without the need of performing additional learning processes. Although these operators are able to deal with both single-label and multi-label problems they are specifically aimed at reducing feature space. However, in the case of multi-label data, this should also be done in the label space. On the other hand, in spite of their success, existing pooling operators are not ideal when handling (multi-label) datasets that do not have an explicit topological organization. In this paper, we present a deep neural architecture using bidirectional association-based pooling layers to extract high-level features and labels in multi-label classification problems. Our approach uses an association function to detect distinct pairs of neurons that will be aggregated into pooled neurons. In the first pooling layer, our proposal computes the Pearson correlation among the variables as the basis to quantify the association values. In addition, we propose an iterative procedure that allows estimating the association degree among pooled neurons in deeper layers without the need of recomputing the correlation matrix. The main advantage of this deep neural architecture is that it allows extracting high-level features and labels on datasets with no specific topological organization. The numerical results show that our bidirectional neural network

*Corresponding author
Email address: mbgarcia@uclv.cu (Marilyn Bello)

helps reduce the number of problem features and labels while preserving network's discriminatory power.

Keywords: deep neural networks, multi-label classification, high-level features, high-level labels, association-based pooling

1. Introduction

In traditional supervised machine learning algorithms [1], instances are usually associated with a single label, so each observation belongs to a single decision class. The distinctive characteristic of multi-label data is that each instance can belong to several classes at the same time.

In a multi-label problem, every instance x is described by a number of features while being associated with a set of labels. The label set $\mathcal{L}$ of x can be represented as $\mathcal{L} = \{l_1(x), l_2(x), \ldots, l_C(x)\}$, with C being the total number of class labels. A value $l_i(x) = 1$ is interpreted as the presence of label l_i for observation x, while $l_i(x) = 0$ indicates its absence. The general format of a multi-label dataset is depicted in Table 1. The dataset contains N elements described by M features $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$.

Table 1: Multi-label dataset with N instances, M features and C labels.

f_1	f_2	$\ldots$	f_M	l_1	l_2	$\ldots$	l_C
$f_1(x_1)$	$f_2(x_1)$	$\ldots$	$f_M(x_1)$	$l_1(x_1)$	$l_2(x_1)$	$\ldots$	$l_C(x_1)$
$f_1(x_2)$	$f_2(x_2)$	$\ldots$	$f_M(x_2)$	$l_1(x_2)$	$l_2(x_2)$	$\ldots$	$l_C(x_2)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$f_1(x_N)$	$f_2(x_N)$	$\ldots$	$f_M(x_N)$	$l_1(x_N)$	$l_2(x_N)$	$\ldots$	$l_C(x_N)$

The multi-label classification (MLC) task consists in predicting the label sets of unseen instances by analyzing the training set [11, 45]. This learning concept appears in many real-world situations [31, 14, 34, 13]. For example, a gene might have more than one function in yeast gene functional analysis [6], an image might be associated with a set of labels in natural scene classification [2], whereas a document might belong to several predetermined topics in automatic webpage categorization [29].

Multi-label problems are often high-dimensional, so the extraction of features helps reduce the complexity of building and exploiting the multi-label classification algorithms. One of the tasks of deep learning [8] is dedicated

to learning high-level features from the available data. These high-level features denote information granules that enable faster learning. In the case of multi-label classification, extracting high-level features is not enough as these problems regularly involve a large number of labels.

Several authors [46, 20, 38] have proposed multi-label classification solutions inspired on deep learning techniques. Some of these solutions [4, 17] rely on the autoencoders [12, 37], which allow for the unsupervised learning of features. With this approach, the underlying features from any given data can be efficiently extracted, thus resulting in a well-coded reduced dataset. Meanwhile, other authors rely on Convolutional Neural Networks (CNNs) [8, 18, 42] to solve prediction problems, such as images processing, sound, text and videos [34, 36, 5, 47, 26]. CNNs use a mixture of convolutional, pooling and standard processing layers for capturing abstract features describing the problem. The role of convolutional layers is to detect local conjunctions of features from the previous layer, while the pooling layers are used to merge reasonably similar features into high-level ones.

Pooling layers help reduce redundancy and the number of parameters before building a multilayer or recurrent neural network that performs the remaining processing operations. In the literature, several pooling operators have been reported [24, 23]. Two common pooling methods are average pooling and max pooling, which compute the average or maximal presence of a feature in a certain neighborhood, respectively. The common ground of these operators is that they focus on data with a well-defined structure (such as image and video) where the term *feature neighborhood* makes sense. However, while it is interesting to recognize faces, or classify objects in images and videos, the truth is that there are other domains in which the data do not have a topological organization. In those cases, using standard pooling operators might have little sense, even when the application problem at hand could benefit significantly from a deep learning solution. Besides, although these operators are able to deal with both single-label and multi-label classification problems they are specifically aimed at reducing feature space. However, in the case of multi-label data, we can benefit significantly from implementing similar operations on the label space.

In this paper, we propose a *bidirectional neural network* to extract high-level features and labels from multi-label data, which have no specific structure. This architecture is composed of several stacked association-based pooling layers, which are built starting from the features and the labels at the same time. This pooling operator uses an association-based function to de-

tect pairs of features to be aggregated into high-level features. These features will compose the next pooling layer together with those neural entities that did not fulfill a user-specified minimum association threshold. Aiming at lightening the computational burden when computing the association among the neurons, we also present an iterative method termed *forward propagation of the association*. This method allows propagating the association on both the features and the labels simultaneously.

Once the high-level features and labels have been extracted, we connect them with one or several hidden layers composed of ReLU, sigmoid or hyperbolic tangent neurons. This provides our bidirectional deep neuronal network with prediction capabilities since the pooling layers are only devoted to extracting the high-level features and labels. Finally, since the output layer of the network is composed by an abstract and reduced representation of the labels, it is necessary to carry out a decoding process. To do that, we connect the high-level labels with the original ones by means of one or several hidden processing layers. The numerical simulations on several MLC datasets show a significant reduction in the number of problem features and labels, without affecting network's discriminatory capability. Having a smaller neural system would imply that the training time is smaller when compared with a model that uses the full set of features and labels.

This paper is organized as follows. Section 2 presents a brief discussion of some relevant pooling variants, and some deep learning methods used to solve the MLC problem. Section 3 describes the proposed association-based pooling architecture. Finally, Section 4 is dedicated to numerical simulations and discussion, while Section 5 provides closure to our paper.

2. Related Work

This section provides a brief discussion of some relevant pooling variants. Likewise, we review the literature on deep learning models that have been used to solve MLC problems.

2.1. Relevant pooling variants

The intuition behind pooling is that the exact location of a feature is less important than its rough location relative to other features. Roughly speaking, pooling goes about computing high-level features (that might not be perfect) such that the problem representation is confined to the relevant

features. Simplifying the problem representation implies that we need less parameters to solve the same problem, which translates into more efficient models. Besides, pooling layers do not involve learnable parameters, so they are easily computed. That is why it is common to insert a pooling layer between convolutional layers in a CNN architecture.

The simplest (yet widely used and preferred) pooling methods are the *max* pooling and the *average* pooling [39, 8]. The former selects the most representative element in each pooling region, while the latter takes the arithmetic mean of the elements in the region.

The *generalized pooling* proposed in [19] uses different strategies to combine traditional max and average pooling. The mixed max-average pooling is the simplest approach in which specific mixing proportion parameters are learned from historical data, whereas the gated max-average learns a "gating mask" with the same dimensionality as the pooled region. The scalar result of the inner product between the gating mask and the region being pooled is transformed with a sigmoid function to produce the mixing proportion. Finally, the tree max-average pooling further learns pooling filters and their responsive combination by using a binary tree.

The *global average pooling* in [21] replaces the fully connected layers with the average of each feature map, which feeds a softmax layer. One advantage of this operator is that it naturally fits the convolutional character by relating feature maps and decision classes, which can be interpreted as confidence values for outputs. Another advantage is that there is no parameter to be optimized, which might prevent overfitting to happen.

The *spatial pyramid pooling* in [10] removes the fixed-size constraint of CNNs by generating a fixed-length representation regardless of the image scale, which is then fed into the fully-connected architecture. The information aggregation at a deeper stage of the network's hierarchy suppresses the need for cropping, which not only allows for arbitrary aspect ratios but also enables arbitrary scales. Thus, when the input images have different scales, the model with the same filter sizes will extract features at different scales. Furthermore, the coarsest pyramid level has a single bin that covers the entire image, which resembles a global pooling operation.

Pooling received some criticism in [28] where the authors stated that it might dispose information about the position, orientation or scale of the features when there is no enough overlap. This information might be needed to discover relations among parts of an object. An alternative to deal with this issue could be to use capsules and routing-by-agreement, where capsules

denote different properties of the image. Capsules produce a vector using a dynamic routing mechanism to ensure that the capsule's output is sent to an appropriate parent in the layer above. Beyond whether capsules are the best option or not to deal with this issue, the reader can notice that there is an underlying assumption regarding the main action field of pooling: data need to have a specific topological organization.

2.2. Deep learning methods for MLC problems

Many researchers have employed neural network solutions [12, 9] to learn the complex multi-label class boundaries. In literature, modifications to the multilayer perceptron [44], radial basis functions [43], extreme learning machine [16] and deep neural networks [36] have been reported.

Within the deep learning field, CNNs have proved to be a powerful tool when solving image-related tasks. In [32] the authors proposed a unified framework for multilabel image classification, which uses CNNs and recurrent neural networks to model the label co-occurrence dependency in a joint image/label embedding space. Many other MLC problems have been addressed with CNNs [41, 27, 7, 33, 40, 48, 22]. However, most of these solutions are aimed at image and natural language processing.

Capturing the co-occurrence and interdependencies among multiple class labels can help improve algorithms' performance. With this goal in mind, Wicker et al. [35] proposed a multi-label classifier that uses an autoencoder to extract non-linear dependencies among features. Similarly, in [38] the authors introduced a deep neural network to exploit the correlation of labels by means of a canonical-correlated autoencoder.

On the other hand, in [25] a deep learning approach with restricted Boltzmann machines is proposed. This model attempts to reduce the interdependence among features, thus ending up in better feature-space representations. In [17] a stacked autoencoder is used to generate a discriminating and reduced input representation of the multi-label data. Stacked autoencoders are capable of extracting underlying features from any given data. In [4] the authors proposed a kernel extreme learning machine autoencoder for the input space, while also using a non-equilibrium label completion algorithm to discover the underlying correlation among the labels.

2.3. Summary and motivation

After revising the literature two conclusions emerge. Firstly, using existing pooling and convolution operators on traditional MLC datasets (those

which are not oriented to image or video) might have little sense. The reader can notice that a pooling region in an image is just a sub-image where pixels preserve their location with respect to their neighbors. This is no longer true when operating with traditional datasets since it would suffice to exchange the order of instances to obtain different pooling regions. This suggests that the invariance property of pooling regions is no longer preserved, thus more adequate operators would need to be proposed. Secondly, MLC problems often involve a large number of labels, thus leading to very dense networks. This means that we can benefit from extracting both high-level features and labels such that we can build simpler neural systems. The autoencoders found in the literature do learn effective high-level feature representations from tabular datasets, but they do not provide similar mechanisms to obtain high-label representations for the label set.

3. Bidirectional deep neural network

In this section, we propose a bidirectional network composed of stacked association-based pooling layers to extract high-level features and labels in MLC problems with no specific topological organization.

3.1. Bidirectional association-based pooling

The main building-block of our bidirectional neural network architecture refers to the association-based pooling layers, which are aimed at extracting high-level features and labels. When operating with the features, the first pooling layer is composed of neurons denoting the problem features themselves (referred to as low-level features), whereas in deeper pooling layers the neurons denote high-level features that emerge from the pooling process. The same reasoning applies when operating with the labels. In both cases, the pooled neurons in the t-th layer will be composed of neurons belonging to the previous layer such that they fulfill a certain association threshold. This also suggests that low-level features (and labels) could reach the deeper layers if they are poorly associated with each other.

Equations (1) and (2) show the operation to be performed to determine in the t-th layer the pairs of neurons that will be aggregated into the k-th pooled neuron for features and labels, respectively,

$$\left(f_p^{(t-1)}, f_q^{(t-1)}\right) = \underset{(f_i^{(t-1)}, f_j^{(t-1)}) \in \mathcal{DF}_k^{(t)}}{\operatorname{argmax}} \psi_{\mathcal{F}}\left(f_i^{(t-1)}, f_j^{(t-1)}\right) \tag{1}$$

$$\left(l_p^{(t-1)}, l_q^{(t-1)}\right) = \underset{(l_i^{(t-1)}, l_j^{(t-1)}) \in \mathcal{DL}_k^{(t)}}{\operatorname{argmax}} \psi_{\mathcal{L}}\left(l_i^{(t-1)}, l_j^{(t-1)}\right) \tag{2}$$

where $0 \leq \psi\left(\cdot,\cdot\right) \leq 1$ computes the association degree between two neurons, $\mathcal{DF}_k^{(t)}$ and $\mathcal{DL}_k^{(t)}$ represent the feasible domains of the k-th pooled neuron for problem features and labels, respectively:

$$\begin{aligned} \mathcal{DF}_k^{(t)} = \Big\{ & (f_i^{(t-1)}, f_j^{(t-1)}) \in \mathcal{F}^{(t-1)} \times \mathcal{F}^{(t-1)} : \\ & \psi_{\mathcal{F}}(f_i^{(t-1)}, f_j^{(t-1)}) \geq \xi_1, i < j \Big\} \setminus \bigcup_{s=1}^{k-1} \bar{\mathcal{DF}}_s^{(t)} \end{aligned} \tag{3}$$

$$\begin{aligned} \mathcal{DL}_k^{(t)} = \Big\{ & (l_i^{(t-1)}, l_j^{(t-1)}) \in \mathcal{L}^{(t-1)} \times \mathcal{L}^{(t-1)} : \\ & \psi_{\mathcal{L}}(l_i^{(t-1)}, l_j^{(t-1)}) \geq \xi_2, i < j \Big\} \setminus \bigcup_{s=1}^{k-1} \bar{\mathcal{DL}}_s^{(t)} \end{aligned} \tag{4}$$

where ξ_1 and ξ_2 denote two user-specified association thresholds, $\mathcal{F}^{(t-1)}$ and $\mathcal{L}^{(t-1)}$ are the sets of neurons in the $(t-1)$ pooling layers, while $\mathcal{F}^{(0)}$ and $\mathcal{L}^{(0)}$ are the sets of features and labels, respectively. Moreover, $\bar{\mathcal{DF}}_s^{(t)}$ and $\bar{\mathcal{DL}}_s^{(t)}$ are the sets of unfeasible pairs of neurons that result from the successive pooling operations, which are computed as:

$$\begin{aligned} \bar{\mathcal{DF}}_s^{(t)} = \Big\{ & (f_{s_i}^{(t-1)}, f_{s_j}^{(t-1)}) \in \mathcal{F}^{(t-1)} \times \mathcal{F}^{(t-1)} : \exists f_{s_r}^{(t-1)}, \\ & f_{s_i}^{(t-1)} \oplus f_{s_r}^{(t-1)} = f_s^{(t)} \vee f_{s_j}^{(t-1)} \oplus f_{s_r}^{(t-1)} = f_s^{(t)}, s_i < s_j \Big\} \end{aligned} \tag{5}$$

and

$$\begin{aligned} \bar{\mathcal{DL}}_s^{(t)} = \Big\{ & (l_{s_i}^{(t-1)}, l_{s_j}^{(t-1)}) \in \mathcal{L}^{(t-1)} \times \mathcal{L}^{(t-1)} : \exists l_{s_r}^{(t-1)}, \\ & l_{s_i}^{(t-1)} \odot l_{s_r}^{(t-1)} = l_s^{(t)} \vee l_{s_j}^{(t-1)} \odot l_{s_r}^{(t-1)} = l_s^{(t)}, s_i < s_j \Big\} \end{aligned} \tag{6}$$

where $f_s^{(t)}$ and $l_s^{(t)}$ denote the s-th pooled neurons in the current layer, and $\oplus$ and $\odot$ stand for the operators used to conform the pooled neurons. The current pooling layer of features and labels will be composed of pooled neurons $f_k^{(t)} = f_p^{(t-1)} \oplus f_q^{(t-1)}$ and $l_k^{(t)} = l_p^{(t-1)} \odot l_q^{(t-1)}$, respectively, and also of neurons that did not fulfill the association relation.

The intuition behind the pooling operation is as follows. Pooled neurons in the t-th layer are derived from the neurons that belong to the $(t-1)$ layer, which can be either low-level or high-level features. Aiming at determining the feasible set of features to conform the k-th pooled neuron in the t-th layer, we need to compute the feasible domain of that neuron. This set involves all pairs of neurons that belong to the $(t-1)$ layer, excluding those which have already been used to create the $k-1$ pooled neurons in the current layer. The same applies when pooling the problem labels.

The association-based pooling causes high-level features and labels to appear, therefore progressively reducing the problem dimensionality. It should be highlighted that the proposed architecture assumes that pooling layers are stacked one after another, thus there are no learnable parameters to be adjusted between two consecutive layers.

3.2. Computing the degree of association among neurons

In this subsection, we will discuss how to estimate the association degree between two neurons in our neural system. The proposed method uses the Pearson correlation among variables as the basis to compute the association degree among pooled neurons. However, this would imply that the correlation matrix needs to be re-calculated in each layer.

As an alternative, we can compute the correlation matrix among low-level neurons (i.e., features and labels of the problem) and derive the degree of association of the high-level features and labels from the degree of association between each pair of neurons in the previous layer. This method is referred to as *forward propagation-based association*, which suppresses the need for scanning the training set at each pooling layer.

The intuition behind this approach is that we can estimate the association degree between pairs of pooled neurons from the association degree among neurons that compose the pooled ones. Therefore, only the association degree values estimated in the previous pooling layer are required to determine the association degree of neurons in the current layer. Overall, there are three different scenarios that need to be considered.

Case 1. Aiming at conforming the first pooling layer, we must determine the association among the problem features,

a) $\psi_{\mathcal{F}}(f_i^{(0)}, f_j^{(0)})$, $\psi_{\mathcal{F}}(f_i^{(0)}, f_p^{(0)})$, $\psi_{\mathcal{F}}(f_i^{(0)}, f_q^{(0)})$, $\psi_{\mathcal{F}}(f_j^{(0)}, f_p^{(0)})$, $\psi_{\mathcal{F}}(f_j^{(0)}, f_q^{(0)})$ and $\psi_{\mathcal{F}}(f_p^{(0)}, f_q^{(0)})$, where $f_x^{(0)}$ with $x \in \{i, j, p, q\}$ denotes the features.

b) $\psi_{\mathcal{L}}(l_i^{(0)}, l_j^{(0)})$, $\psi_{\mathcal{L}}(l_i^{(0)}, l_p^{(0)})$, $\psi_{\mathcal{L}}(l_i^{(0)}, l_q^{(0)})$, $\psi_{\mathcal{L}}(l_j^{(0)}, l_p^{(0)})$, $\psi_{\mathcal{L}}(l_j^{(0)}, l_q^{(0)})$ and $\psi(l_p^{(0)}, l_q^{(0)})$, where $l_x^{(0)}$ with $x \in \{i, j, p, q\}$ denotes the labels.

This can be done by computing the absolute Pearson correlation among all pairs of neurons denoting the problem features.

Case 2. In deeper layers, we should estimate the degree of association between a pooled neuron and a non-pooled neuron.

a) Let us suppose that the neuron of pooled features in the t-th layer has the form $f_i^{(t)} \oplus f_j^{(t)}$ while the non-pooled neuron is $f_p^{(t)}$. Equation (7) shows how to compute the degree of association between them,

$$\psi_{\mathcal{F}}(f_i^{(t)} \oplus f_j^{(t)}, f_p^{(t)}) = \psi_{\mathcal{F}}(f_i^{(t)}, f_j^{(t)}) \times \min\left\{\psi_{\mathcal{F}}(f_i^{(t)}, f_p^{(t)}), \psi_{\mathcal{F}}(f_j^{(t)}, f_p^{(t)})\right\} \tag{7}$$

b) Let us suppose that the neuron of pooled labels in the t-th layer has the form $l_i^{(t)} \odot l_j^{(t)}$ while the non-pooled neuron is $l_p^{(t)}$. Equation (7) shows how to compute the degree of association between them,

$$\psi_{\mathcal{L}}(l_i^{(t)} \odot l_j^{(t)}, l_p^{(t)}) = \psi_{\mathcal{L}}(l_i^{(t)}, l_j^{(t)}) \times \min\left\{\psi_{\mathcal{L}}(l_i^{(t)}, l_p^{(t)}), \psi_{\mathcal{L}}(l_j^{(t)}, l_p^{(t)})\right\}. \tag{8}$$

Case 3. Similarly to the previous case, we should estimate the degree of association between two pooled neurons.

a) Let us suppose that two pooled features in the t-th layer have the form $f_i^{(t)} \oplus f_j^{(t)}$ and $f_p^{(t)} \oplus f_q^{(t)}$, respectively. Equation (9) shows how to compute the degree of association between them,

$$\begin{aligned}\psi_{\mathcal{F}}(f_i^{(t)} \oplus f_j^{(t)}, f_p^{(t)} \oplus f_q^{(t)}) = \min\big\{&\psi_{\mathcal{F}}(f_i^{(t)} \oplus f_j^{(t)}, f_p^{(t)}),\\ &\psi_{\mathcal{F}}(f_i^{(t)} \oplus f_j^{(t)}, f_q^{(t)}), \psi_{\mathcal{F}}(f_p^{(t)} \oplus f_q^{(t)}, f_i^{(t)}), \psi_{\mathcal{F}}(f_p^{(t)} \oplus f_q^{(t)}, f_j^{(t)})\big\}\end{aligned} \tag{9}$$

b) Let us suppose that two pooled labels in the t-th layer have the form $l_i^{(t)} \odot l_j^{(t)}$ and $l_p^{(t)} \odot l_q^{(t)}$, respectively. Equation (10) shows how to compute the degree of association between them,

$$\begin{aligned}\psi_{\mathcal{L}}(l_i^{(t)} \odot l_j^{(t)}, l_p^{(t)} \odot l_q^{(t)}) = \min\big\{&\psi_{\mathcal{L}}(l_i^{(t)} \odot l_j^{(t)}, l_p^{(t)}),\\ &\psi_{\mathcal{L}}(l_i^{(t)} \odot l_j^{(t)}, l_q^{(t)}), \psi_{\mathcal{L}}(l_p^{(t)} \odot l_q^{(t)}, l_i^{(t)}), \psi_{\mathcal{L}}(l_p^{(t)} \odot l_q^{(t)}, l_j^{(t)})\big\}.\end{aligned} \tag{10}$$

In addition to this forward association method, we will reduce quadratically the association thresholds when moving from one pooling layer to the next, that is, $\xi_1^{(t+1)} = (\xi_1^{(t)})^2$ and $\xi_2^{(t+1)} = (\xi_2^{(t)})^2$. This adjustment is advisable due to the fact that $0 \leq \psi(\cdot,\cdot) \leq 1$, thus the product operation will very likely degrade the estimated association degree.

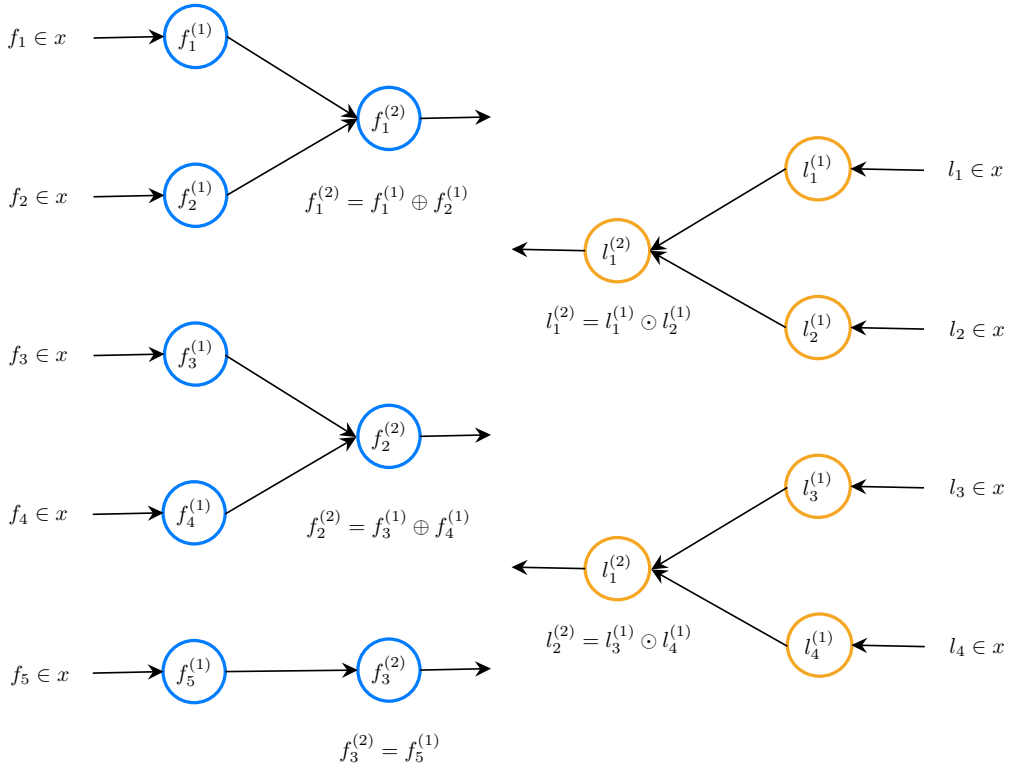
The reader can notice that, if the association among features and labels is performed using the correlation-based association approach, then the computational complexity is $O(N^2 log_2(|\mathcal{F}| * |\mathcal{L}|))$, where N denotes the number of instances, $|\mathcal{F}|$ and $|\mathcal{L}|$ are the cardinality of the set of features and labels, respectively. However, if we use the propagation-based association approach, then the computational complexity is $O(log_2(|\mathcal{F}| * |\mathcal{L}|))$.

3.3. The proposed network architecture

In this subsection, we describe a bidirectional association-based architecture that is composed of pooling and hidden layers.

Before doing that, it seems convenient to illustrate how the bidirectional association-based pooling works. Figure 1 shows an example where two pooling layers are running for both features (left figure) and labels (right figure). In this example, five high-level neurons were formed from the association of the features pairs (f_1, f_2) and (f_3, f_4), and the labels pairs (l_1, l_2) and (l_3, l_4). The f_5 feature is not associated with another feature, so it is transferred directly to the following pooling layer. Our approach reduces the number of features from 5 to 3, and the number of labels from 4 to 2, which represents a reduction rate of 40% and 50%, respectively.

After extracting the high-level features we can design a neural system to perform the multi-label classification process. The proposed architecture involves two sub-networks, one performing a high-level classification and another decoding the high-level predictions. In the first sub-network, the output of the last pooling layer is used as the input to a fully connected network with one or several hidden layers that maps high-level features to high-level labels. The second sub-network is also composed of one or several hidden layers, which decodes the high-level labels (predicted by the first sub-network) to the original problem labels. These hidden layers can be equipped with either ReLU, sigmoid or hyperbolic tangent transfer functions, thus conferring the neural system with prediction capabilities. The number of hidden layers and hidden neurons are parameters to be defined by the user based on the problem complexity and hardware availability.



(a) Feature pooling process (b) Label pooling process

Figure 1: Bidirectional association-based pooling operating with five low-level features and four low-level labels. In terms of features, the first pooling layer contains two pooled neurons and one low-level feature. This suggests that the fifth low-level feature either did not fulfill the association threshold or that the other features reported higher association values. In terms of labels, the first pooling layer contains two pooled labels as both pairs of low-level labels did fulfill the association threshold.

Figure 2 depicts the network architecture involving five high-level neurons that emerge from the association-based pooling layers. In this example, $f_1^{(2)}, f_2^{(2)}, f_3^{(2)}$ denote high-level features, $l_1^{(2)}, l_2^{(2)}$ represents high-level labels, while $l_1^{(1)}$, $l_2^{(1)}$, $l_3^{(1)}, l_4^{(1)}$ are the low-level labels associated with the problem. These high-level neurons are connected by means of two multilayered networks, each composed of two hidden layers. It is worth highlighting that the second network is needed to transform the abstract representations (high-level labels) back to its original form. Therefore, the model learns to reconstruct the data from the encoded representations.

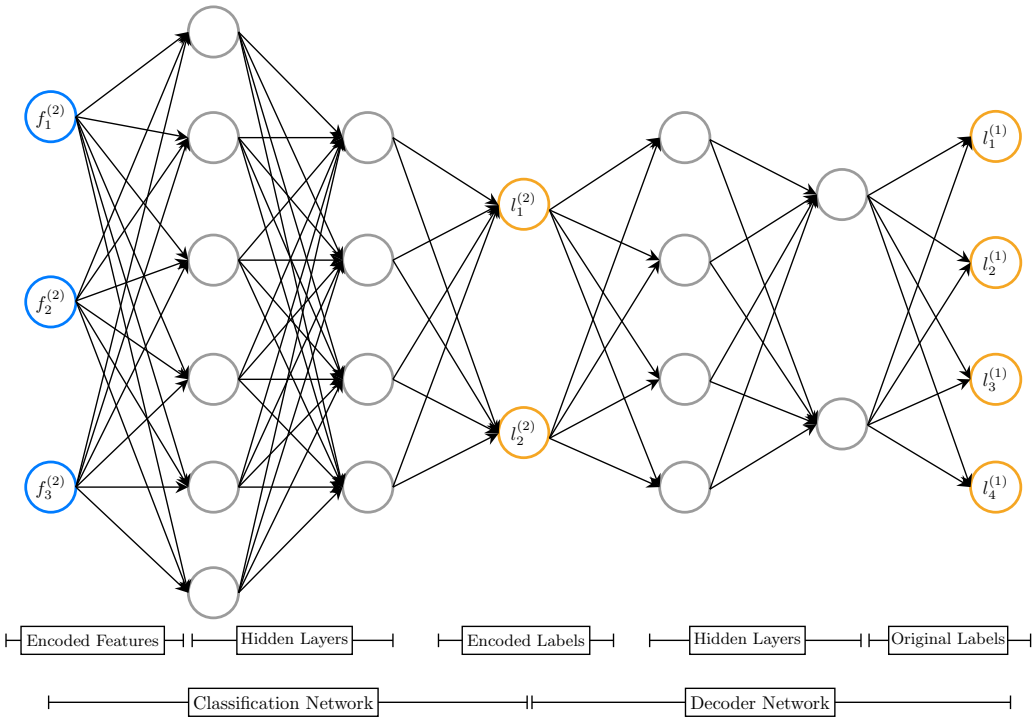


Figure 2: Neural network architecture involving three high-level features, two-high-level labels, four low-level labels and four hidden layers. As mentioned, the number of hidden layers and hidden neurons are parameters to be defined by the user based on the estimated problem complexity and hardware availability.

Remark that the goal of this neural system is to extract high-level features while performing the classification process without significant affectations. Therefore, we should not expect higher prediction rates since our proposal is not equipped with additional learning capabilities.

4. Numerical simulations

In this section, we evaluate the performance of the proposed network architecture by using several MLC problems. More specifically, we will study i) how the model performs in terms of both accuracy and number of high-level features and labels when varying the association thresholds, ii) the reduction in the problem representation, and iii) the accuracy loss induced by operating with the extracted features and labels.

4.1. Experimental setup

To perform the simulations we used 15 MLC datasets from the RUMDR repository [3]. In these problems, the number of instances ranges from 207 to 269,648, the number of features goes from 72 to 2,150, and the number of labels from 4 to 400 (see Table 2). Moreover, we report the average *maximal absolute correlation*, as computed when the association-based pooling procedure is performed on features and labels.

Table 2: Characterization of datasets used for simulations.

Dataset	Name	Instances	Features	Labels	Correlation-F	Correlation-L
D1	emotions	593	72	6	0.62	0.39
D2	scene	2,407	294	6	0.74	0.22
D3	yeast	2,417	103	14	0.49	0.57
D4	stackex-chemistry	6,961	540	175	0.18	0.13
D5	stackex-chess	1,675	585	227	0.27	0.24
D6	stackex-cooking	10,491	577	400	0.14	0.14
D7	stackex-cs	9,270	635	274	0.18	0.18
D8	GnegativePseAAC	1,392	440	8	0.29	0.22
D9	GpositivePseAAC	519	440	4	0.33	0.34
D10	VirusPseAAC	207	440	6	0.40	0.22
D11	mediamill	43,907	120	101	0.93	0.29
D12	bookmarks	87,856	2,150	208	0.37	0.23
D13	imdb	120,919	1,001	28	0.08	0.15
D14	nus-wide-BOW	269,648	501	81	0.32	0.16
D15	nus-wide-VLAD	269,648	129	81	0.1	0.16

The operators $\oplus$ and $\odot$ to create the high-level features and labels are the average (avg), and the minimum (min) and maximum (max), respectively. Moreover, we set the number of pooling layers to 5 in all cases since adding additional pooling layers for a given threshold would only have an impact on the computational complexity. Finally, the association thresholds ξ_1 and ξ_2 will range from 0.0 to 0.8 as we are interested in studying the impact of these parameters on algorithm's performance.

Regarding the two multi-layer networks that complement the pooling layers, in the first network, we used two fully-connected hidden layers such that the number of hidden neurons is equal to $2 * \bar{M}$ and $2 * \bar{C}$, respectively, where $\bar{M}$ and $\bar{C}$ are the number of neurons resulting from the pooling process on the problem features and labels, respectively. In the second network, we used two fully-connected hidden layers comprised of ReLU neurons such that the number of hidden neurons is equal to $2 * \bar{C}$ and $\lfloor C/2 \rfloor$, respectively, where C

represents the number of problem labels. It should be highlighted that other strategies to configure these networks are possible.

The binary cross-entropy is adopted as the error measure while output neurons are equipped with sigmoid transfer functions. The weights associated with the multi-layer networks are adjusted by using an extension to stochastic gradient descent, known as the Adam optimization algorithm [15], meanwhile the number of epochs is set to 100. We employ drop-out regularization [30] with probability 0.5 to alleviate the over-fitting problem.

We propose a measure to asses both the reduction achieved by our network (i.e. the number of parameters to be estimated during the learning phase) and its discriminatory power. Equation (11) attempts to establish a trade-off between network's accuracy and its density,

$$\Gamma_\alpha = \alpha * \min\left\{1, \frac{acc_{after}}{acc_{before}}\right\} + (1-\alpha) * \frac{den_{before} - den_{after}}{den_{before}} \qquad (11)$$

where acc_{before} and acc_{after} denote the accuracy of network without pooling and using the proposed architecture, respectively, and den_{before} and den_{after} the number of parameters to be estimated during the learning phase before and after applying our model, respectively. Moreover, $0 \leq \alpha \leq 1$ is a user-specified parameter determining the relevance of the accuracy of network over its density. Although this measure reports different values for different α values, it is reasonable to use $\alpha > 0.6$ since we regularly want to prioritize the network precision over the parameter reduction.

Even though the Γ_α measure depends on the network architecture, if two networks use the same topology then their sizes will depend on the number of features and labels. Therefore, we will also report the number of high-level features and labels extracted with the aid of pooling layers.

In all experiments conducted in this section, we use 80% of the dataset to build the model and 20% for testing purposes, while all results are averaged over 5 trials to draw more consistent conclusions.

4.2. Results and discussion

Tables 3 and 4 report the best performance achieved by the algorithm for each dataset for both the *max* and the *min* operators, respectively. These tables report the number of high-level features (#HLF), the reduction percentage in the number of labels (%Red-F), the number of high-level labels (#HLL), the reduction percentage in the number of labels (%Red-L), the

accuracy obtained by the network using the extracted features and labels, the accuracy using the original features and labels (baseline model), and the loss of accuracy with respect to the baseline model.

Table 3: Performance assessment of the bidirectional association-based pooling approach when using the *max* operator to create the pooled neurons.

Dataset	#HLF	%Red-F	#HLL	%Red-L	Accuracy	Baseline Accuracy	Loss
D1	33	54.17%	6	0%	0.811	0.823	-0.012
D2	24	91.84%	6	0%	0.915	0.915	0
D3	28	72.82%	13	7.14%	0.796	0.80	-0.004
D4	17	96.85%	22	87.43%	0.987	0.987	0
D5	19	96.75%	29	87.22%	0.99	0.99	0
D6	19	96.71%	50	87.5%	0.995	0.995	0
D7	20	96.85%	35	87.23%	0.99	0.991	-0.001
D8	14	96.82%	8	0%	0.916	0.918	-0.002
D9	53	87.95%	4	0%	0.84	0.866	-0.026
D10	49	88.86%	6	0%	0.809	0.794	0.015
D11	4	96.67%	4	96.04%	0.965	0.965	0
D12	78	96.37%	23	88.94%	0.99	0.99	0
D13	80	92.01%	9	67.86%	0.927	0.988	-0.06
D14	18	96.4%	3	96.3%	0.977	0.977	0
D15	9	92.97%	5	93.83%	0.977	0.977	0

Table 4: Performance assessment of the bidirectional association-based pooling approach when using the *min* operator to create the pooled neurons.

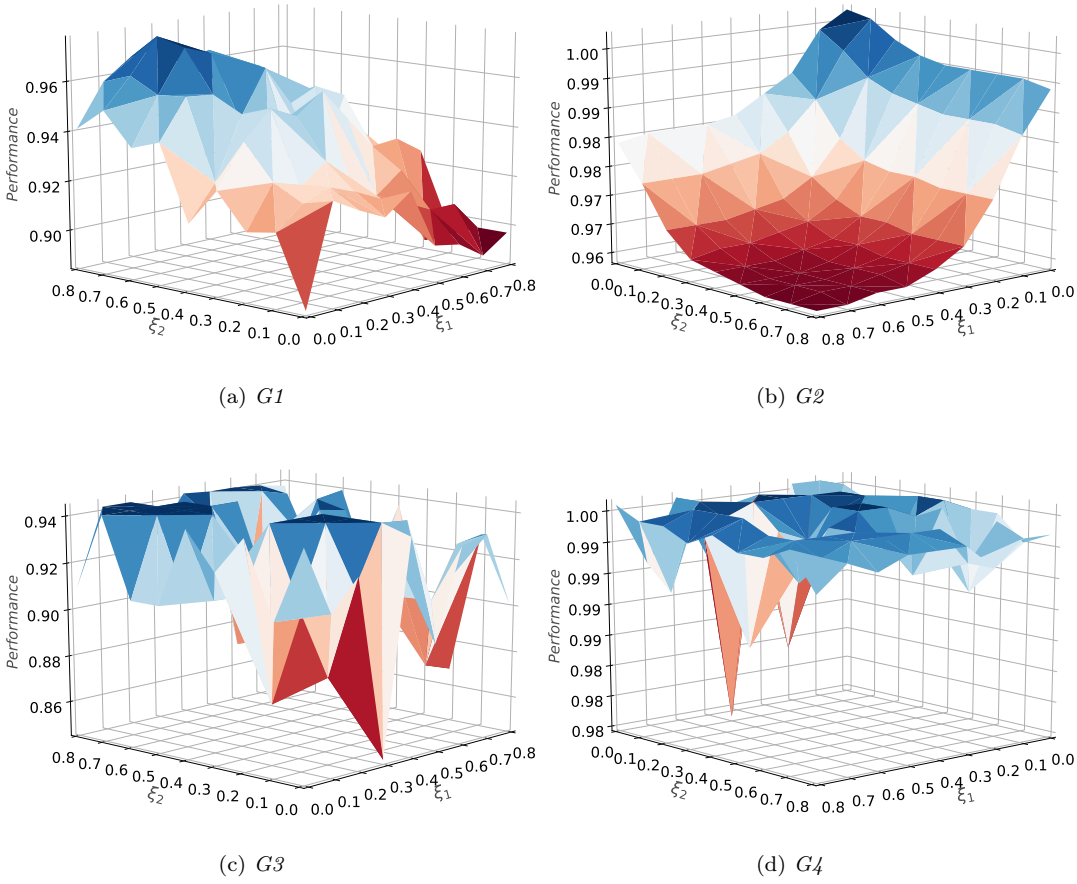
Dataset	#HLF	%Red-F	#HLL	%Red-L	Accuracy	Baseline Accuracy	Loss
D1	43	40.28%	6	0%	0.815	0.823	-0.008
D2	24	91.84%	6	0%	0.913	0.915	-0.002
D3	44	57.28%	13	7.14%	0.798	0.80	-0.002
D4	17	96.85%	22	87.43%	0.988	0.987	0.001
D5	19	96.75%	29	87.22%	0.99	0.99	0
D6	19	96.71%	50	87.5%	0.995	0.995	0
D7	20	96.85%	35	87.23%	0.991	0.991	0
D8	14	96.82%	8	0%	0.915	0.918	-0.003
D9	53	87.95%	4	0%	0.837	0.866	-0.029
D10	49	88.86%	5	16.67%	0.805	0.794	0.011
D11	4	96.67%	7	93.07%	0.965	0.965	0
D12	78	96.37%	10	95.19%	0.99	0.99	0
D13	80	92.01%	14	50%	0.928	0.988	-0.06
D14	18	96.4%	3	96.3%	0.977	0.977	0
D15	9	92.97%	3	96.3%	0.977	0.977	0

From these results we can observe that our proposal significantly reduces the number of features and labels with a percentage reduction up to 96%

and 87%, respectively. It can be noticed that the bidirectional association-based pooling reports a maximal accuracy loss of 0.06 for the $D13$ dataset. However, in some cases, we observed a small increase in the accuracy (e.g., dataset $D10$) even when our network was not conceived to increase the prediction rates but to obtain the same performance with smaller networks. For those problems having low variability in accuracy (datasets within the $G2$ and $G4$ groups), our proposal has no loss in accuracy. Moreover, the numerical results suggest that the performance of the model when using the *max* and *min* operators are similar to each other.

Figure 3 shows the average $\Gamma_{0.95}$-values for different ξ_1 and ξ_2 values. Aiming at better analyzing the results, we have gathered the datasets into four groups $G1 = \{D1, D2, D3, D8, D9, D10\}$, $G2 = \{D4, D5, D6, D7, D14, D15\}$, $G3 = \{D13\}$, and $G4 = \{D11, D12\}$. The difference among these groups relies on the correlation among the problem features and among the the labels attached to each problem, as well as the variability on the accuracy when suppressing some problem features in a random way. The first group contains the datasets having a high or middle correlation among their features and labels, together with a high variability in their accuracy. The second group consists of datasets having low correlation among their features and labels, and a small variability on their accuracy. The third group consists of problems with a high variability in its accuracy, and a low correlation among their features and labels. The last group contains datasets with a high or middle correlation among their features and labels, a small variability on their accuracy. In this simulation, we have adopted the *max* operator as we concluded that both operators lead to similar results.

From these results we can draw some interesting conclusions. Firstly, in the $G1$ group our model produces the highest $\Gamma_{0.95}$-values for large values of ξ_2 and small values of ξ_1. After further inspection, we noticed that the features in these datasets are highly correlated but the correlation with respect to the labels is less evident. Therefore, the granulation of the label space must be finer (e.g., with $\xi_2 \geq 0.5$). The results for the $G2$ group reveal that the $\Gamma_{0.95}$-values increase when both parameter values decrease. This result is somehow expected since these problems report small variability on their accuracy when using random features. If this situation comes to light, then the association matrix (resulting from the association function) can be replaced by a random matrix. Thirdly, small variations in the values of the thresholds ξ_1 and ξ_2 produce drastic changes in the accuracy of the $D13$ problem. For the fourth group, the values of the thresholds do not matter much since these problems



(a) *G1* (b) *G2*

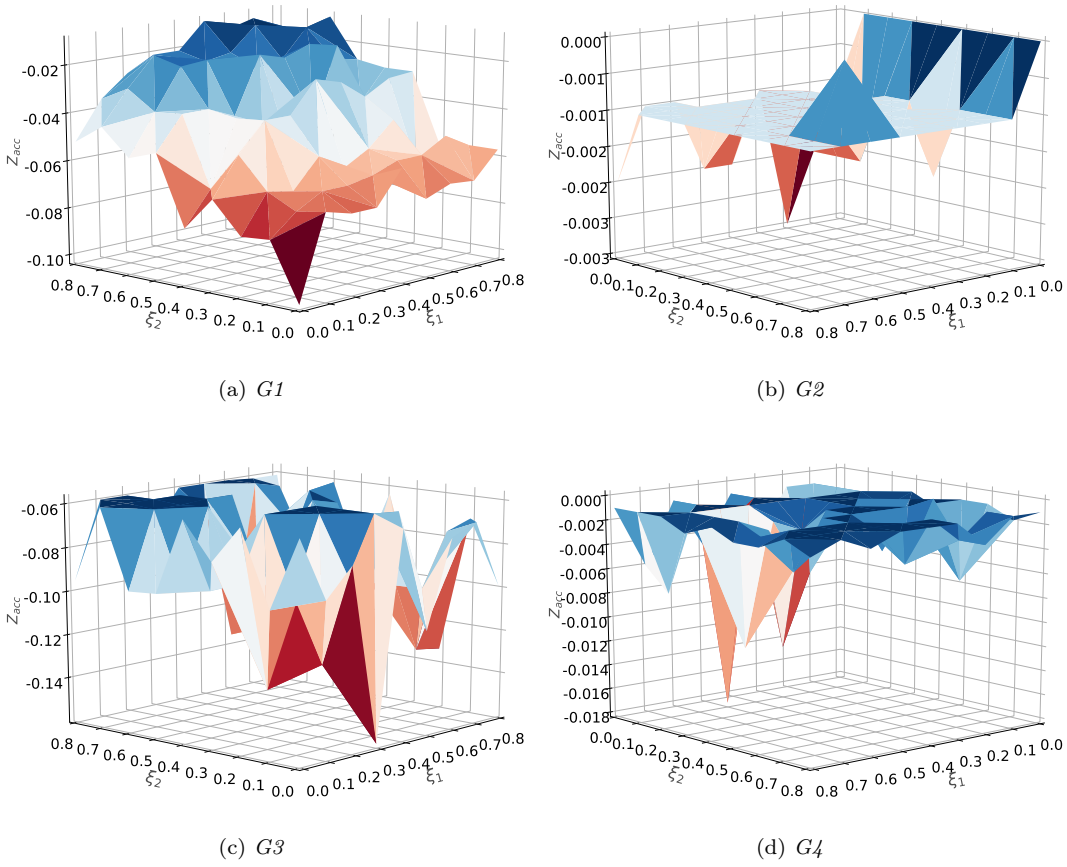
(c) *G3* (d) *G4*

Figure 3: Average $\Gamma_{0.95}$-values over $G1$, $G2$, $G3$ and $G4$ when changing the association thresholds ξ_1 and ξ_2 together with the *max* operator.

can be solved with random features.

Figure 4 illustrates the differences in accuracy (Z_{acc}) between the network without pooling and using the proposed architecture, while Figure 5 shows the reduction rate in terms of network density.

For problems in the $G1$ group, the accuracy loss is small when ξ_1 and ξ_2 are more strict (i.e. $\xi_1, \xi_2 \geq 0.6$). However, the thresholds do not seem to be very relevant for datasets in $G2$ and $G4$, as concluded from Figure 3. For the $G3$ group, the threshold ξ_2 seems to be very important in order to reduce the loss of accuracy. For the datasets in $G1$ and $G3$, the threshold ξ_2 does not play a pivotal role since these datasets have few labels, so there is no



(a) *G1* (b) *G2*

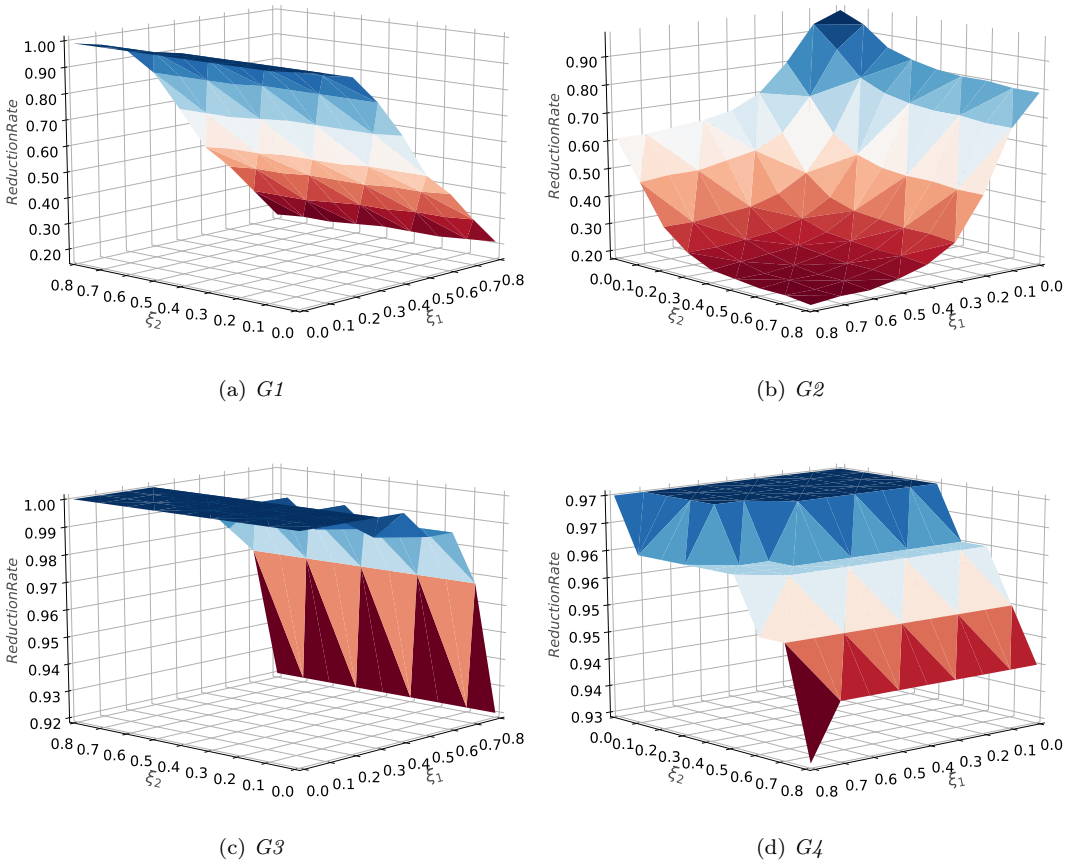
(c) *G3* (d) *G4*

Figure 4: Average of the Z_{acc} values over over $G1$, $G2$, $G3$ and $G4$ according to the association thresholds ξ_1 and ξ_2 together with the *max* operator.

significant reduction. However, for the datasets in $G2$ and $G4$ the opposite holds since the number of labels is rather large.

The following experiment is dedicated to comparing our proposal against a random pooling, which replaces the association values with random numbers. Figure 6 shows the differences in performance between the association-based pooling and the random variant for different threshold values. Positive values implies that our model performs better.

As expected, the differences in performance for the first group is more explicit and always favors our approach (the Z_{acc} values are always positive). For the third group the behavior is similar, although the results depend on

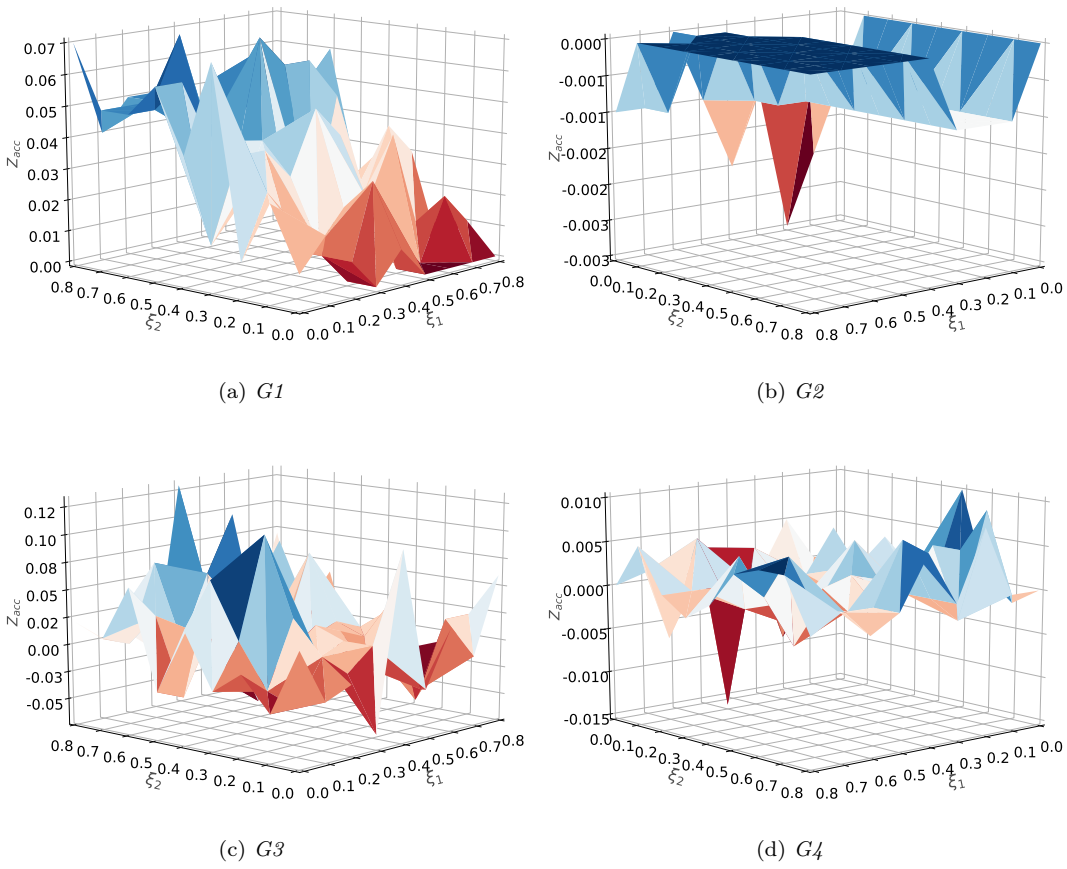


(a) *G1* (b) *G2*

(c) *G3* (d) *G4*

Figure 5: Reduction rate over $G1$, $G2$, $G3$ and $G4$ according to the association thresholds ξ_1 and ξ_2 together with the *max* operator.

the association threshold as this dataset has a very low correlation among the features and labels. The negative values suggest that computing the association by means of the correlation is not always effective, mainly when the dataset has low correlation among its features or labels. In these scenarios, the greatest differences are observed when using high association thresholds, which cause the bidirectional pooling to transfer the features from a layer to another without reducing the network significantly. For the second group these differences are very small since the datasets in that group report high accuracy values for different settings. Something similar occurs for the fourth group of datasets. However, we can obtain good results when $\xi_2 \geq 0.5$ since



(a) *G1* (b) *G2*

(c) *G3* (d) *G4*

Figure 6: Average of the Z_{acc} values over $G1$, $G2$, $G3$ and $G4$ for the association thresholds ξ_1 and ξ_2 together with the *max* operator.

these datasets, despite having small variability in their accuracy, show some correlation among their features and labels.

The numerical simulations suggest that we can compute simpler models without harming networks' prediction power. Therefore, one would expect this result to have a positive impact on the training time. Table 5 displays the time (in seconds) needed to extract the high-level features and labels, and the time needed to train the model using the extracted features and labels with respect to the baseline model. This experiment was performed in Kaggle notebook for a single training process.

The reader can notice that, once the high-level features and labels have

Table 5: Time (in seconds) needed to i) extract the high-level features and labels with the aid of pooling layers, ii) train the model with the extracted features and labels, and iii) train the model with all features and labels.

Dataset	pooling-time	model-training-time	baseline-training-time
D1	0	1	0
D2	0	1	2
D3	0	1	1
D4	4	4	17
D5	1	2	6
D6	10	14	46
D7	8	8	35
D8	0	1	2
D9	0	1	1
D10	0	1	1
D11	1	13	23
D12	829	64	1,765
D13	250	14	578
D14	142	58	534
D15	12	64	145

been extracted, the training times are significantly smaller when compared with the ones attached with the baseline neural model. The same holds if we consider the times of extracting the high-level features and labels together, mainly for larger datasets such as $D12$-$D15$. It comes with no surprise that performing the pooling operations is a rather expansive process because of the correlation component. This is why our further research efforts will be dedicated to replacing the correlation-based association function with more efficient functions. In that regard, the concept of granular entropy at a local level seems to be a good starting point.

5. Concluding remarks

In this paper, we have introduced a new network architecture that uses a bidirectional associations-based pooling to extract high-level features and labels from multi-label data. Unlike the pooling approaches reported in the literature, our proposal does not require input data to have any topological properties as typically occurs with images and videos. Our architecture was conceived so that the block of pooling layers does not involve learnable parameters changing during the training phase. Therefore, the goal behind our model is to extract high-level features rather than producing better predic-

tions. However, the features or labels must show some degree of correlation, which may not be applicable in all situations.

The numerical simulations have shown that our proposal is able to significantly reduce the number of parameters in deep feed-forward neural networks without harming their discriminatory power. It was also observed that the association threshold regulating the pooling on the problem labels has a significant effect on algorithm's performance. In contrast, when the correlation assumption is difficult to fulfill, the bidirectional-based pooling just transfers the neurons from the current layer to the following one without harming the performance or reducing the network.

Extracting high level features and labels increases the possibility of building networks with more transparent inference models. For example, by using *post-hoc* interpretability techniques we could shed light into inner reasoning of the model when operating with high level features. These techniques regularly have an exponential algorithmic complexity, thus having networks with fewer parameters certainly helps reach this goal.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable and constructive feedback. Moreover, we would like to thank Isel Grau from the Vrije Universiteit Brussel (Belgium) and Leonardo Concepción from the Universiteit Hasselt (Belgium) for their comments.

References

[1] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+ Business Media.

[2] Carneiro, G., Chan, A. B., Moreno, P. J., & Vasconcelos, N. (2007). Supervised learning of semantic classes for image annotation and retrieval. *IEEE transactions on pattern analysis and machine intelligence*, *29*, 394–410.

[3] Charte, F., Charte, D., Rivera, A., del Jesus, M. J., & Herrera, F. (2016). R ultimate multilabel dataset repository. In *International Conference on Hybrid Artificial Intelligence Systems* (pp. 487–499). Springer.

[4] Cheng, Y., Zhao, D., Wang, Y., & Pei, G. (2019). Multi-label learning with kernel extreme learning machine autoencoder. *Knowledge-Based Systems*, *178*, 1–10.

[5] Choi, K., Fazekas, G., & Sandler, M. B. (2016). Automatic tagging using deep convolutional neural networks. In *ISMIR*.

[6] Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. In *Advances in neural information processing systems* (pp. 681–687).

[7] Gargiulo, F., Silvestri, S., Ciampi, M., & De Pietro, G. (2019). Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, *79*, 125–138.

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[9] Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*, 1904–1916.

[11] Herrera, F., Charte, F., Rivera, A. J., & Del Jesus, M. J. (2016). Multilabel classification. In *Multilabel Classification* (pp. 17–31). Springer.

[12] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

[13] Jiang, M., Pan, Z., & Li, N. (2017). Multi-label text categorization using l21-norm minimization extreme learning machine. *Neurocomputing*, *261*, 4–10.

[14] Jing, X.-Y., Wu, F., Li, Z., Hu, R., & Zhang, D. (2016). Multi-label dictionary learning for image annotation. *IEEE Transactions on Image Processing*, *25*, 2712–2725.

[15] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*.

[16] Kongsorot, Y., & Horata, P. (2014). Multi-label classification with extreme learning machine. In *2014 6th International Conference on Knowledge and Smart Technology (KST)* (pp. 81–86). IEEE.

[17] Law, A., & Ghosh, A. (2019). Multi-label classification using a cascade of stacked autoencoder and extreme learning machines. *Neurocomputing*, *358*, 222–234.

[18] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*, 436–444.

[19] Lee, C.-Y., Gallagher, P. W., & Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In A. Gretton, & C. C. Robert (Eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (pp. 464–472). PMLR volume 51 of *Proceedings of Machine Learning Research*.

[20] Lian, S.-m., Liu, J.-w., Lu, R.-k., & Luo, X.-l. (2019). Captured multi-label relations via joint deep supervised autoencoder. *Applied Soft Computing*, *74*, 709–728.

[21] Lin, M., Chen, Q., & Yan, S. (2014). Network in network. *CoRR*, *abs/1312.4400*.

[22] Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 115–124). ACM.

[23] McFee, B., Salamon, J., & Bello, J. P. (2018). Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, *26*, 2180–2193.

[24] Passalis, N., & Tefas, A. (2017). Learning bag-of-features pooling for deep convolutional neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[25] Read, J., & Pérez-Cruz, F. (2014). Deep learning for multi-label classification. *ArXiv*, *abs/1502.05988*.

[26] Rios, A., & Kavuluru, R. (2015). Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics* (pp. 258–267). ACM.

[27] Roman-Rangel, E., & Marchand-Maillet, S. (2019). Inductive t-sne via deep learning to visualize multi-label images. *Engineering Applications of Artificial Intelligence*, *81*, 336–345.

[28] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 3856–3866). Curran Associates, Inc.

[29] Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine learning*, *39*, 135–168.

[30] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*, 1929–1958.

[31] Wan, S., Duan, Y., & Zou, Q. (2017). Hpslpred: an ensemble multi-label classifier for human protein subcellular location prediction with imbalanced source. *Proteomics*, *17*, 1700262.

[32] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2285–2294).

[33] Wang, S.-J., Lin, B., Wang, Y., Yi, T., Zou, B., & wen Lyu, X. (2019). Action units recognition based on deep spatial-convolutional and multi-label residual network. *Neurocomputing*, *359*, 130–138.

[34] Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., Zhao, Y., & Yan, S. (2015). Hcp: A flexible cnn framework for multi-label image classification. *IEEE transactions on pattern analysis and machine intelligence*, *38*, 1901–1907.

[35] Wicker, J., Tyukin, A., & Kramer, S. (2016). A nonlinear label compression and transformation method for multi-label classification using

autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 328–340). Springer.

[36] Wu, F., Wang, Z., Zhang, Z., Yang, Y., Luo, J., Zhu, W., & Zhuang, Y. (2015). Weakly semi-supervised deep learning for multi-label image annotation. *IEEE Transactions on Big Data*, *1*, 109–122.

[37] Xie, Y., Zhang, J., Xia, Y., & Shen, C. (2019). A mutual bootstrapping model for automated skin lesion segmentation and classification. `arXiv:1903.03313`.

[38] Yeh, C.-K., Wu, W.-C., Ko, W.-J., & Wang, Y.-C. F. (2017). Learning deep latent space for multi-label classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[39] Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology* (pp. 364–375). Springer.

[40] Yu, Q., Wang, J., Zhang, S., Gong, Y., & Zhao, J. (2017). Combining local and global hypotheses in deep neural network for multi-label image classification. *Neurocomputing*, *235*, 38–45.

[41] Yu, W.-J., Chen, Z.-D., Luo, X., Liu, W., & Xu, X.-S. (2019). Delta: A deep dual-stream network for multi-label image classification. *Pattern Recognition*, *91*, 322–331.

[42] Zhang, J., Xie, Y., Xia, Y., & Shen, C. (2019). Attention residual learning for skin lesion classification. *IEEE transactions on medical imaging*, *38*, 2092–2103.

[43] Zhang, M.-L. (2009). M l-rbf: Rbf neural networks for multi-label learning. *Neural Processing Letters*, *29*, 61–74.

[44] Zhang, M.-L., & Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, *18*, 1338–1351.

[45] Zhang, M.-L., & Zhou, Z.-H. (2013). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, *26*, 1819–1837.

[46] Zhang, N., Ding, S., & Zhang, J. (2016). Multi layer elm-rbf for multi-label learning. *Applied Soft Computing*, *43*, 535–545.

[47] Zhu, J., Liao, S., Lei, Z., & Li, S. Z. (2017). Multi-label convolutional neural network based pedestrian attribute classification. *Image and Vision Computing*, *58*, 224–229.

[48] Zhuang, N., Yan, Y., Chen, S., Wang, H., & Shen, C. (2018). Multi-label learning based deep transfer neural network for facial attribute classification. *Pattern Recognition*, *80*, 225–240.