Classifying Signals on Irregular Domains
via Convolutional Cluster Pooling

Angelo Porrello   **Davide Abati**   **Simone Calderara**   **Rita Cucchiara**

University of Modena and Reggio Emilia, Modena, Italy
`name.surname@unimore.it`

Abstract

We present a novel and hierarchical approach for supervised classification of signals spanning over a fixed graph, reflecting shared properties of the dataset. To this end, we introduce a Convolutional Cluster Pooling layer exploiting a multi-scale clustering in order to highlight, at different resolutions, locally connected regions on the input graph. Our proposal generalises well-established neural models such as Convolutional Neural Networks (CNNs) on irregular and complex domains, by means of the exploitation of the weight sharing property in a graph-oriented architecture. In this work, such property is based on the centrality of each vertex within its soft-assigned cluster. Extensive experiments on NTU RGB+D, CIFAR-10 and 20NEWS demonstrate the effectiveness of the proposed technique in capturing both local and global patterns in graph-structured data out of different domains.

1 Introduction

Convolutional Neural Networks (CNNs) have been successfully applied in different domains, such as speech recognition [7], image classification [16], and video analysis [26]. In these domains, data can be described as a signal defined on a regular grid, whose underlying dimension can be 1d, 2d or 3d. One of the key aspects of CNNs is that such a regular structure makes it possible to exploit local and stationary properties of data. Moreover, the convolution operator, its behaviour being

equivariant to translations, allows filters with a limited support on the input grid, leading to a significantly smaller number of parameters with respect to Fully Connected Networks. However, we are surrounded by data lying on an underlying structure, which typically has an irregular and non-euclidean nature. This is the case, for instance, of document databases, 3D skeletal data, information from social networks and chemical compounds. In all these domains, the relationships among entities are more complex than in the case of a simple grid-like connectivity. Instead, graphs constitute better representation forms, because they model directly the topological structures of such data domains, through edge weights. For this reason, many efforts have recently been made [2, 14, 4] in an attempt to generalise CNNs for graph-structured data. In this work we focus on signal classification in homogeneus graphs. In such context, each sample obeys a single $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ weighted graph, which reflects the physics as well as the structure of the given problem. The point in which a sample differs from the others is represented by the value of each vertex in the graph. As in the case of [8], we refer to each sample as a realisation of a signal on $\mathcal{G}$. The aim is to learn a function which maps each sample into the label space. By doing so, similarly to what CNNs do for images, at each step we shall exploit information coming from the neighbouring nodes. To this end, we propose a novel architecture, built by stacking multiple Convolutional Cluster Pooling (CCP) layers as depicted in Fig. 1. This layer, which is the main subject of this study, firstly performs a clustering operation on the input graph, resulting in a coarser output graph, whose affinity matrix reflects relationships among clusters regressed at training time. By doing so, a good basis for building local receptive fields is achieved. Secondly, according to the neighbours' vision dictated by the first step, the layer selects for each cluster a fixed number of candidate nodes for the aggregation phase, and sorts them depending on a centrality-based rank within the cluster. In this respect, it is worth noting that weight sharing across the
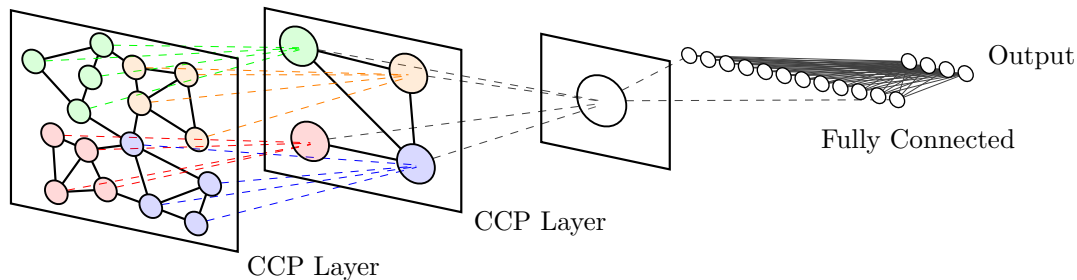
Figure 1: An overview of the proposed architecture. Multiple applications of the CCP layer lead to a multi-scale clustering of the input graph, exploiting both local and global properties during the information's flow from input to output. Finally, a Multi-Layer Perceptron classifies a global representation of the input signal, captured by a feature vector on a singleton graph.

graph's neighbourhoods can be successfully exploited. The contributions of this research are two-fold. Firstly, we provide a hierarchical framework for supervised learning in homogeneous graph contexts. Secondly, we propose a spatial formulation for graph filtering which, as for CNNs, exploits weight sharing.

2 Related Work

Because of its generality and potential applications in different domains, the possibility to extend neural networks to deal with graph-structured data has recently become an active research area. In this regard, two main approaches arise from the existing literature: spectral methods, which encode the graph structure using the graph Fourier Transform, and spatial methods, modelling the filtering operation through the construction of locally connected neighbourhoods.

In general terms, spectral approaches take advantage of the fact that eigenvectors of the graph Laplacian span a space in which the convolution operator is diagonal [8]. Bruna *et al.* [2] exploited this property and defined a frequency filtering operation for neural networks. However, with such kind of formulation, it is not possible to relate the filtering operation within the spectral domain with the one performed in the vertex domain. In order to define localized linear transformations (i.e. operations also interpretable in the vertex domain [8]), Defferrard *et al.* [4] proposed the use of polynomial spectral filters, with a theoretical guarantee of k-localisation in space. In addition, they provided a recursive approximation of such filtering through Chebyshev polynomials, which prevent expensive computations needed by the Laplacian eigenvectors.

The other branch concerns spatial methods, which directly model convolutions as a linear combination of vertices in a local neighbourhood. In this respect, the authors of Diffusion-Convolutional Neural Networks

(DCNNs) [1] presented an approach in which feature vectors are spread according to the hop distance in a depth search tree, the latter having as parent root the node for which the operation has to be done. Kipf & Welling [14] proposed a fast and simple layer-wise propagation rule, which involves the use of normalized adjacency matrix. An interesting aspect of this method is how, from a spectral perspective, it may also be seen as an approximation of a localized first-order filter. Notably, the framework described by Monti *et al.* [21] led to a unified vision for all spatial approaches, in which the differences among different types of methods lie on the notion of the local coordinate system.

Our model is to be considered a spatial approach, because we derive a convolution-like operation directly from the clustering step, the latter creating groups of spatially close vertices itself. Inspired by Deep Locally Connected Networks [2], we then assimilate the pooling operation with the filtering stage, providing a strategy to enable weight sharing across graph's clusters. Moreover, we propose a learnable multilevel strategy for graph coarsening, which may be performed directly during the learning process. On the latter point, our proposal differs from [4], where the Graclus multilevel clustering algorithm [23] has been used, the latter being performed during a pre-processing step. On this note, we were inspired by the work of Such *et al.* [25], who introduced graph embed pooling, a way to produce pooled graphs with a parametrizable number of vertices. However, our method is quite different in the computation of the pooled vertices' feature maps. Indeed, while they consider output vertices as a weighted combination of all input vertices (where weights are given by clusters' memberships), we only sample a fixed number of vertices, and combine them according to learnable kernel's weights. Our spatial formulation builds on the concept that the weight sharing property can be inducted in a graph-oriented architecture, provided that a nodes-ordering criteria

has previously been defined. A similar idea arised in PATCHY-SAN [22], in which a ranking procedure and a graph normalisation technique have been used to generate local receptive fields, resulting in an adjacency matrix for each selected node. This way, the authors managed to exploit structural and local properties of input graph very well. However, the authors did not address how intermediate sub-graphs should be merged and, consequently, how that procedure should be stacked on multiple layers. The latter point could make it difficult to capture global structures with the same effectiveness. Differently, our method generates receptive fields for entire clusters, enabling graph coarsening and a hierarchical architecture.

3 Hierarchical Graph Clustering

A graph $\mathcal{G}$ can be defined as an ordered pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $\mathcal{N}$ nodes and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ a set of edges. In this paper we are interested in classifying signals defined on an undirected and weighted graph, in which $\mathcal{E}$ can be described by a real symmetric matrix $\mathcal{A} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ which, for each couple of vertices $\mathcal{V}_i$ and $\mathcal{V}_j \in \mathcal{V}$, provides the strength (weight) of their connections. More generally, we refer to $\mathcal{A}$ as an affinity matrix, in which each entry $\mathcal{A}_{i,j}$ gives an affinity score between $\mathcal{V}_i$ and $\mathcal{V}_j$. In addition to the affinity matrix, which describes the topology of the graph and the relationships between nodes, it is common practice to define a signal $\mathcal{F} : \mathcal{V} \to \mathbb{R}^{d_{IN}}$ on the vertex set, which associates a d_{IN} dimensional feature vector to each node of the graph.

Graph Soft Clustering. Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define a soft $\mathcal{K}$-partition of the graph a function that associates at each vertex $\mathcal{V}_i \in \mathcal{V}$ a membership value, in probabilistic terms, to each of the $|\mathcal{K}|$ cluster. The $\mathcal{K}$-partition can be shortly represented by a stochastic matrix $K \in \mathbb{R}^{\mathcal{N} \times |\mathcal{K}|}$ where the element $K_{i,k}$ equals the probability of vertex $\mathcal{V}_i$ belonging to cluster $\mathcal{K}_k$, $\mathrm{P}(\mathcal{V}_i \in \mathcal{K}_k)$. Given the affinity matrix $\mathcal{A} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$, we compute the following matrix:

$$\mathcal{A}^{\mathcal{K}} = K^{\mathrm{T}}(\mathcal{A} - I_{\mathcal{N}} \odot \mathcal{A})K, \tag{1}$$

where $I_{\mathcal{N}}$ indicates the identity matrix of size $\mathcal{N}$ [1]. $\mathcal{A}^{\mathcal{K}} \in \mathbb{R}^{|\mathcal{K}| \times |\mathcal{K}|}$ is highly related to the affinity matrix of the graph that can be obtained by applying to the original graph the soft $\mathcal{K}$-partition described by K. Indeed, if all the membership distributions behaved like a multivariate Kronecker Delta distribution, given an adjacency matrix $\mathcal{A}$ describing an undirected graph,

---

[1]The subtraction of the diagonal is performed to avoid the consideration of self-connections during cluster affinity and *Cohesion* computations, in Eq. 2.

$\mathcal{A}^{\mathcal{K}}_{k,k}$ $k = 1, 2, \ldots, |\mathcal{K}|$ would be equal to the double of the number of edges existing between the nodes insides the k-th cluster, and $\mathcal{A}^{\mathcal{K}}_{k,k'}$ $k, k' = 1, 2, \ldots, |\mathcal{K}|$ $k \neq k'$ would be equal to the number of edges connecting pair of nodes respectively belonging to the k-th and k'-th cluster. Likewise, in the soft case, we have:

$$\begin{aligned} \mathcal{A}^{\mathcal{K}}_{k,k} &= \boldsymbol{Cohesion}(K_k) \\ &= 2 \sum_{(\mathcal{V}_i, \mathcal{V}_j) \in \binom{\mathcal{V}}{2}} K_{i,k} K_{j,k} \, \mathcal{A}_{i,j}, \\ \mathcal{A}^{\mathcal{K}}_{k,k'} &= \sum_{i=1}^{\mathcal{N}} K_{i,k} \sum_{\substack{j=1 \\ j \neq i}}^{\mathcal{N}} K_{j,k'} \, \mathcal{A}_{i,j}. \end{aligned} \tag{2}$$

In such form, $\mathcal{A}^{\mathcal{K}}_{k,k'}$ can be considered an affinity measure between the k-th and k'-th nodes in the graph reduced by K. We consider as a 'good' soft $\mathcal{K}$-partition a partition that produces cluster with maximal cohesion. However, equivalent to ratio and normalized cut [29], we penalise imbalanced solutions through the addition of a penalty related to the size of each cluster:

$$\begin{aligned} \max_{K \in \mathbb{R}^{\mathcal{N} \times |\mathcal{K}|}} C(K) \quad &= \frac{1}{2} \sum_{k=1}^{|\mathcal{K}|} \frac{Cohesion(K_k)}{Vol(K_k)} \\ &= \frac{1}{2} \mathbf{1}^{\mathrm{T}}_{|\mathcal{K}|} \left[\mathrm{diag}(\mathcal{A}^{\mathcal{K}}) \oslash (K^{\mathrm{T}} D) \right] \\ \text{subject to} \quad &\sum_{k=1}^{|\mathcal{K}|} K_{i,k} = 1 \quad i = 1, 2, \ldots, \mathcal{N}, \end{aligned} \tag{3}$$

where

$$\boldsymbol{Vol}(K_k) = \sum_{i=1}^{\mathcal{N}} D_i \, \mathrm{P}(\mathcal{V}_i \in \mathcal{K}_k) \quad k = 1, 2, \ldots, |\mathcal{K}| \tag{4}$$

and $\oslash$ indicates the entry-wise division between two vectors of the same length, and $D \in \mathbb{R}^{\mathcal{N}}$ stand for a column vector in which each entry is equal to the degree of the corresponding node. This way, we obtain clusters with maximal cohesion and, at the same time, minimum size. It is worth noting that the main difference between such formulation and the well-known normalized cut relies on the membership's definition, the latter being defined, in our case, by means of soft assignments.

Graph Hierarchical Soft Clustering. Let consider $\mathcal{A}^{\mathcal{K}_1}$ as the affinity matrix of the graph that can be obtained by applying a soft $\mathcal{K}$-partition, given by $K^{(1)} \in \mathbb{R}^{|\mathcal{K}_0| \times |\mathcal{K}_1|}$, to the original graph described by $\mathcal{A}$, where $|\mathcal{K}_0| = \mathcal{N}$. We can now partition $\mathcal{A}^{\mathcal{K}_1}$, based on the entries of a generic matrix $K^{(2)} \in \mathbb{R}^{|\mathcal{K}_1| \times |\mathcal{K}_2|}$, in order to obtain a new affinity matrix $\mathcal{A}^{\mathcal{K}_2}$, and so
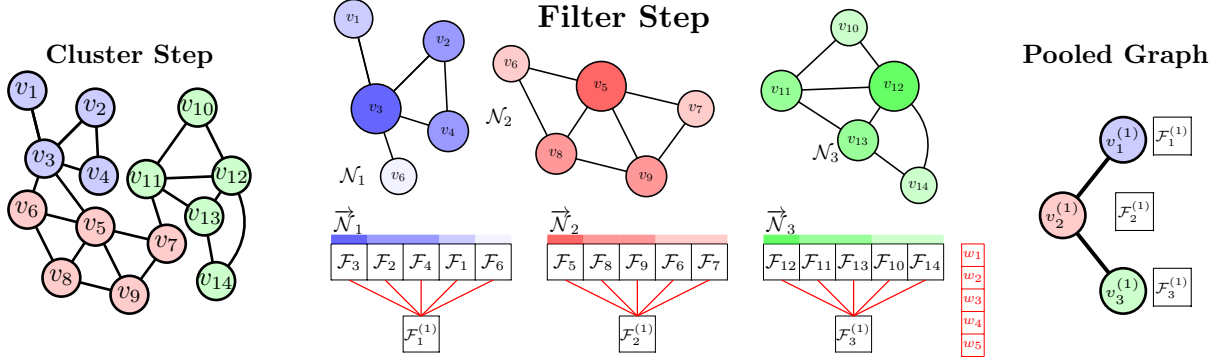
Figure 2: An illustration of the proposed CCP layer. Left, the cluster step outputs node's membership distribution among a pre-defined number of clusters. Centre, the filter step: a) selects, for each cluster, candidate nodes whose feature vectors will be aggregated; b) arranges such candidates according to a with-in cluster centrality score, building the support for the next step; c) aggregates feature vectors by means of a standard 1-d convolution, with stride equal to the kernel width (in this case, $L = 5$). Right, the result of CPP layer consists in a coarsened graph coupled with its filtered pooled signal. Best viewed in color.

on. More generally, a cascade of M soft-partitions, described by an ordered sequence of $\mathcal{A}^{\mathcal{K}_1}, \mathcal{A}^{\mathcal{K}_2}, \ldots, \mathcal{A}^{\mathcal{K}_M}$, forms a soft dendrogram for the original graph. Thus, the problem of obtaining a good dendrogram, in which clusters at each level are characterized by maximal cohesion and minimum size, is formalised as follows:

$$
\max_{\substack{K^{(i)} \in \mathbb{R}^{|\mathcal{K}_{i-1}| \times |\mathcal{K}_i|} \\ i=1,2,\ldots,M}} \quad \mathcal{L}_{\mathcal{K}} = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{|\mathcal{K}_m|} \frac{Cohesion(K_k^{(m)})}{Vol(K_k^{(m)})}
$$

$$
\text{subject to} \quad \sum_{k=1}^{|\mathcal{K}_m|} K_{i,k}^{(m)} = 1 \quad \substack{i=1,2,\ldots,|\mathcal{K}_{m-1}| \\ m=1,2,\ldots,M.}
$$
(5)

4 Convolutional Cluster Pooling

The purpose of our proposal is to exploit the clustering mechanism in order to define a convolutional-like operator, able to ensure equivariance to translation and weight sharing in graph contexts as standard convolutions do. At a high level, our CCP operator can be considered as a layer which, at step m, takes in input an affinity matrix $\mathcal{A}^{\mathcal{K}_m}$ and a multi-dimensional $\mathcal{F}^{(m)} \in \mathbb{R}^{|\mathcal{K}_m| \times d_{IN}}$ signal defined on the vertex set. The output is composed by a new reduced affinity matrix $\mathcal{A}^{\mathcal{K}_{m+1}}$ (reflecting the results of the cluster step) and a pooled signal $\mathcal{F}^{(m+1)} \in \mathbb{R}^{|\mathcal{K}_{m+1}| \times d_{OUT}}$ (reflecting the results of the filter step where d_{OUT} is the dimension of the newly computed features). All architectures used in our experiments are composed by stacking CCP layers, which combine the pooling and filtering stage and, at the same time, increase the number of feature maps, as suggested in [2]. The objective in Eq. 5 is consequently optimised by backprogating gradients.

Cluster Step. First of all, our model performs a soft-clustering step on the input graph (Fig. 2, left). To the purpose we define the stochastic matrix described in Section 3 as the output of a row-wise softmax applied on a variable matrix $U^{(m+1)} \in \mathbb{R}^{|\mathcal{K}_m| \times |\mathcal{K}_{m+1}|}$ learned during training:

$$
K_{i,k}^{(m+1)} = P(\mathcal{V}_i^{(m)} \in \mathcal{K}_k^{(m+1)}) = \frac{e^{U_{i,k}^{(m+1)}}}{\sum_{k'=1}^{|\mathcal{K}_{m+1}|} e^{U_{i,k'}^{(m+1)}}}
$$
(6)

where $i = 1, 2, \ldots, |\mathcal{K}_m|$ and $k = 1, 2, \ldots, |\mathcal{K}_{m+1}|$. In the second place, the downsampled affinity matrix $\mathcal{A}_{m+1}^{\mathcal{K}}$ describing the soft-partitioned graph induced by $K^{(m+1)}$ is computed by means of the quadratic form in Eq. 1. Eventually, we add a normalisation operation based on the degree matrix D [14] in order to prevent numerical instabilities:

$$
\overline{\mathcal{A}}^{\mathcal{K}_{m+1}} = D^{-\frac{1}{2}} \mathcal{A}^{\mathcal{K}_{m+1}} D^{-\frac{1}{2}}.
$$
(7)

Neighbourhood selection. For each cluster $\mathcal{K}_k^{(m+1)}$, we select as candidate set $\mathcal{N}_k^{(m+1)}$ for the filtering stage the set containing the most L representative nodes (where L is an hyperparameter) as:

$$
\mathcal{N}_k^{(m+1)} = \operatorname*{argmax}_{\mathcal{V}' \subset \mathcal{V}^{(m)}, |\mathcal{V}'|=L} \sum_{v \in \mathcal{V}'} Rank\,(v \to \mathcal{K}_k^{(m+1)}),
$$
(8)

where the rank of a vertex $\mathcal{V}_i^{(m)}$ for a particular cluster $\mathcal{K}_k^{(m+1)}$ is given by its centrality in that cluster:

$$
Rank\,(\mathcal{V}_i^{(m)} \to \mathcal{K}_k^{(m+1)}) = (1 + K_{i,k}^{(m+1)}) \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{K}_m|} \mathcal{A}_{i,j}^{\mathcal{K}_m} K_{j,k}^{(m+1)}
$$
(9)

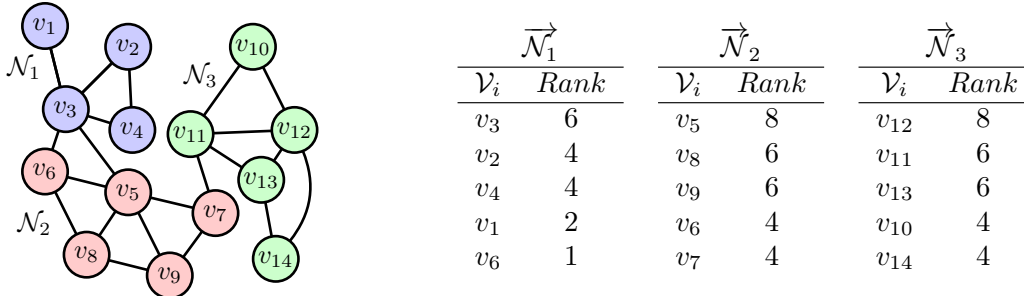| $\overrightarrow{\mathcal{N}_1}$		$\overrightarrow{\mathcal{N}_2}$		$\overrightarrow{\mathcal{N}_3}$	
$\mathcal{V}_i$	$Rank$	$\mathcal{V}_i$	$Rank$	$\mathcal{V}_i$	$Rank$
v_3	6	v_5	8	v_{12}	8
v_2	4	v_8	6	v_{11}	6
v_4	4	v_9	6	v_{13}	6
v_1	2	v_6	4	v_{10}	4
v_6	1	v_7	4	v_{14}	4

Figure 3: The ranking function (described by Equation 9) underpinning the filter step shown in Figure 2. The node colours denote cluster memberships. All edges have weight equal to one.

More intuitively, we consider a node more central if it has a high membership value for the cluster under consideration and, at the same time, a large part of its direct neighbours nodes share the same cluster in the input graph (Fig. 2, centre top).

Further, for each cluster, we compute its features as a linear combination over the feature vectors of its inner nodes. In doing so, we want to exploit the weight sharing property across all neighbours, keeping the parameters' number under control. To this end, we create a coherent support across clusters, in terms of their inner topological structure. In this respect, our proposal is to sort candidates by their centrality within the neighbourhood and, afterwards, apply the same kernel to all clusters.

$$\overrightarrow{\mathcal{N}}_k^{(m+1)} = (\mathcal{F}_{\phi(1)}^{(m)}, \ \mathcal{F}_{\phi(2)}^{(m)}, \ \dots, \ \mathcal{F}_{\phi(L)}^{(m)}),$$
$$\text{with } \overrightarrow{\mathcal{N}}_k^{(m+1)}(l,i) = \mathcal{F}_{\phi(l),i}^{(m)} \quad \substack{l=1,2,\dots,L \\ i=1,2,\dots,d_{IN}}. \tag{10}$$

In simpler terms, the ordered set $\overrightarrow{\mathcal{N}}_k^{(m+1)}$ is recovered by sorting the candidates set $\mathcal{N}_k^{(m+1)}$ according to the *Rank* function. By doing so, the l-th weight of the kernel is always multiplied by the feature vector $\mathcal{F}_{\phi(l)}^{(m)}$ being owned by the l-th node of the neighbour (in terms of centrality), namely $\mathcal{V}_{\phi(l)}^{(m)}$. Fig. 3 shows an example of the neighbourhood selection step for a simple graph.

Neighbourhood Aggregation. The problem we face when sorting nodes by cluster centrality and then applying the same kernel to all neighbours, is that, by doing so we do not take into account the irregularity of the neighbour's shapes. As a matter of fact, the risk of this solution consists in the equal treatment, for different clusters, of nodes indexed in the same position by the sorting stage, whilst exhibiting considerably different centrality values. In order to mitigate such risk, we once again use the centrality measure to implement a gating mechanism on feature vectors during the aggregation phase. The underlying idea is to make the filtering operation invariant to different neighbours

and let the gating mechanism address different cluster's structures and shapes. Roughly speaking, before applying the filtering operation described above, we are giving the centrality scores in input to a generic smoothed function $\sigma : \mathbb{R} \to (0,1)$ (e.g. the sigmoid function). Once this has been done, we perform a point-wise multiplication on the feature vectors of each candidate node. The desired effect of this operation is to attenuate information coming from distant or non-central nodes and, at the same time, preserve signals coming from nodes that reside in the inner part of the cluster. Lastly, our model computes the pooled feature vector as follows:

$$\mathcal{F}_{k,j}^{(m+1)} = \sum_{i=1}^{d_{IN}} \sum_{l=1}^{L} W_{l,i,j} \ (\sigma_{k,l} \cdot \overrightarrow{\mathcal{N}}_k^{(m+1)}(l,i)) + b_j, \tag{11}$$

where $W \in \mathbb{R}^{L \times d_{IN} \times d_{OUT}}$ and $b \in \mathbb{R}^{d_{OUT}}$ are learnable parameters of our CCP layer, whereas $\sigma_{k,l}$ refers to the gate's activation value computed at $Rank \ (\mathcal{V}_{\phi(l)}^{(m)} \to \mathcal{K}_k^{(m+1)})$. As shown in Fig. 2 (centre bottom), this operation is equivalent to a 1-d convolution, enabling weight sharing across clusters.

Optimisation. Given a particular task, we simply add to the task-specific loss $\mathcal{L}_0$ (e.g. a cross-entropy) a term based on quality of the multi-level clustering solutions (Eq. 5) provided during the training phase:

$$\mathcal{L} = \mathcal{L}_0 + \mathcal{L}_{\mathcal{K}}. \tag{12}$$

It is important to note that the presence of the supervision signal may provide information to the process of clusters formation, backpropagating its gradient towards all U variables (Eq. 6).

5   Experiments

In order to show the generality and effectiveness of our model for classification, we apply our architecture to three different domains. First, we train our model

Table 1: Summary of the architectures used in our experiments. We indicate with $(|\mathcal{K}|, d_{OUT})$ the number of nodes and feature maps of each layer. Note that a further softmax layer is employed to estimate class probabilities.

Experiment	Input	Architecture						#params
NTU RGB-D	$(2000, 6)$	$(512, 256)$ $L = 16$	$(128, 384)$ $L = 16$	$(32, 512)$ $L = 8$	$(8, 768)$ $L = 8$	$(1, 1024)$ $L = 8$	FC1024	$\sim 14 \cdot 10^6$
CIFAR-10	$(1024, 3)$	$(256, 256)$ $L = 16$	$(64, 384)$ $L = 16$	$(16, 512)$ $L = 8$	$(4, 768)$ $L = 8$	$(1, 1024)$ $L = 4$	FC1024	$\sim 10 \cdot 10^6$
20NEWS	$(10000, 1)$	$(2048, 128)$ $L = 16$	$(512, 192)$ $L = 16$	$(128, 256)$ $L = 8$	$(32, 384)$ $L = 8$	$(4, 512)$ $L = 8$ $(1, 512)$ $L = 4$	FC256	$\sim 25 \cdot 10^6$

to classify human actions, given the 3D coordinates of each skeleton's joint: to this end, we evaluate it on NTU RGB-D dataset [24]. Secondly, we conduct experiments on image classification. More specifically, we use CIFAR-10 [15] as benchmark test, which is a challenging dataset for non-CNN architectures. Finally, we apply our solution on the 20NEWS dataset, where the goal is to address a text categorisation problem.

Implementation details. In each experiment, all parameters are learned using Adam [13] as an optimisation algorithm, with an initial learning rate fixed to 0.001. We use ELU [3] as activation function and Batch Normalization [9] in all layers to speed up the convergence. Moreover, we apply dropout and l_2 weight regularisation (with value 10^{-4}) to prevent overfitting, as well as standard data augmentation for CIFAR-10 and noise injection coupled with random 3d rotations for NTU RGB-D. All the others architectures' hyperparameters are summarised in Tab. 1. In each experiment, we subsample the input graph until its cardinality becomes equal to one: afterwards, we feed its feature vector into two fully connected layers, followed by a softmax layer providing the target class predictions.

5.1 Classification performances

Action Recognition. The NTU RGB+D Human Activity Dataset [24] is one of the largest datasets for human action recognition. It contains 56,880 action samples for 60 different actions, captured by the Kinect v.2 sensors. Each sample, showing a daily action performed by one or two participants, is made available in 4 different modalities: RGB videos, depth map sequences, 3D skeletal data and infrared videos. Since we are interested in graph classification, in order to perform action recognition, we just use the 3D skeletal data, represented by a temporal sequence of 25 joints[2]. To this end, we model each sequence as a signal $\mathcal{F}^{(0)} \in \mathbb{R}^{(25 \cdot T) \times 3}$ defined on a single fixed spatio temporal graph, whose structure can be summarised as follows: a vertex set $\mathcal{V}_{ST} = \{v_{i,t} | i = 1, 2, \ldots, 25, t = 1, 2, \ldots, T\}$, which includes all joints captured in a fixed length sequence

$(T = 80)$. The edge set $\mathcal{E}_{ST}$ can be defined as the union of two distinct subsets: $\mathcal{E}_S$, which contains all edges within each frame according to the natural human-body connectivity, and $\mathcal{E}_T$, which includes all edges existing between the same joint in two adjacent frame. In order to evaluate the model's performance, as described in [24], we run two different standard benchmarks: the cross-subject setting, in which the train/test split is based on two disjoint sets of actors; and the cross-view setting, where the test samples are captured from a different camera from those collecting the training sequences. We report in Tab. 2 the top-1 classification accuracy on both settings, comparing it with other existing approaches. As illustrated, CCP outperforms previous state-of-the-art methods on this dataset, including other graph-oriented architectures [31, 17], despite being more general and not specifically designed to only address action recognition settings.

Image Classification. We conduct experiments on CIFAR-10, a popular dataset widely used for image recognition. Each image, labeled into one of ten classes, can be treated as a signal defined on a graph, which can in turn be modeled as a 32×32 grid structure. In particular, every pixel is a vertex such a graph, linked to its neighbours following a 8-connectivity. The colour information is encoded as a signal $\mathcal{F}^{(0)} \in \mathbb{R}^{1024 \times 3}$ over such vertexes. As shown in Tab. 3, CCP obtains an encouraging performance in terms of classification accuracy on test set. Indeed, our method outperforms both the best reported fully connected (FC) network [18]

Table 2: Summary of results in terms of classification accuracy for NTU RGB+D.

Method	Cross Subject	Cross View
Lie Group [28]	50.1	52.8
HBRNN-L [5]	59.1	64.0
P-LSTM [24]	62.9	70.3
ST-LSTM+TS [19]	69.2	77.7
TGCNN [31]	71.4	82.9
Temporal Conv [12]	74.3	83.1
Deep STGC$_K$ [17]	74.9	86.3
C-CNN + MTLN [11]	79.6	84.8
CCP (our)	**80.1**	**86.8**

---

[2]The pre-processing step on skeleton data has been performed according to the guidelines provided in [24].

Table 3: Image classification accuracy on CIFAR-10.

Method	Accuracy
Graph-CNNs [25]	68.3
FC [18]	78.6
CCP (our)	**84.4**
Stochastic Pooling [30]	84.9
ResNet [6]	93.6

Table 4: Text categorisation accuracy on 20NEWS.

Method	Accuracy
Linear SVM †	65.9
Softmax †	66.3
Multinomial Naive Bayes †	68.5
FC2500-FC500 †	65.8
Chebyshev - GC32 [4] †	68.3
CCP (our)	**70.1**

† Baselines' results published in [4].

and Graph-CNNs [25] - to the best of our knowledge, the only graph classification model in literature that reports results on CIFAR-10 - by a significant margin. To put our results into perspective, we report the performance obtained by [30], which is the nearest score founded in the literature given by a deep CNN, as well as the results of a state-of-art CNNs like [6]. The gap with respect to the latter is still consistent, suggesting that there is still room for improvement in euclidean domains. Fig. 4 also depicts an illustration of the hierarchical clustering computed on the input grid. As can be seen, as the input image undergoes CCP layers, its representations are computed out of compact regions, resembling dyadic clustering that has been proven a successful downsampling strategy in CNNs.

Text Categorisation. In order to further validate the quality of our proposal in diverse data domains, we apply our model on text categorisation. In this respect, we conduct experiments on the 20NEWS dataset [10], adhering to the guidelines described in [4] for the construction of the shared graph. To summarise, such protocol models each text as a graph which has a node for each common word in the document set. On the other hand, the pairwise connectivities of such graph are shared and obtained assessing the similarities inducted by word2vec embeddings [20], followed by a discretisation step computed through a K-NN pass (with $K = 16$). This way, each document $\mathcal{D}$ can be represented as a signal over a fixed graph, implemented as the word's distribution observed in $\mathcal{D}$. As indicated in Tab. 4, the discussed approach leads to good performances, defeating both baselines and the graph convolutional layer based on polynomial spectral filters. On this latter point, our architecture seems to take
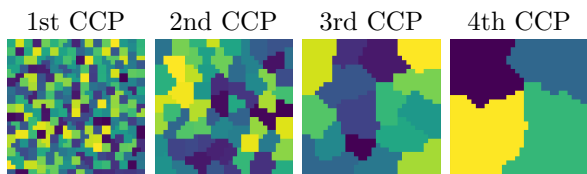
advantages of its depth and hierarchical nature, differently from [4] where a shallow graph convolutional network has been employed to categorise documents.

5.2 Model analysis

Comparisons with other coarsening approaches. We further compare our proposal w.r.t. three different works (Tab. 5): GCN [14], Chebyshev filtering [4] and Graph Attention Networks (GAT) [27]. In this respect we use the Graclus algorithm [23] for coarsening the input graph and vary the graph filtering strategy accordingly to the referenced work. For all the experiments we keep architectural settings described in Tab. 1 and use the public implementation of these works. Futhermore, we also design a non-coarsening baseline by performing a global average pooling (GAP) on nodes features (after GAT manipulation) before the fully connected classification layers. The experiment suggests that CCP outperforms, by a consistent margin, the previous GCN+coarsening approach. Moreover, it still outperforms Graclus as a coarsening strategy even though recent filters such as GAT are applied. In this regard, we empirically observed that order-invariant filters (e.g. GAT), despite being more general, may treat the same graph differently, according to the attention scores. This is in fact a great advantage when graph layout may vary across examples, though potentially unrewarding when the support remains the same through all the dataset.
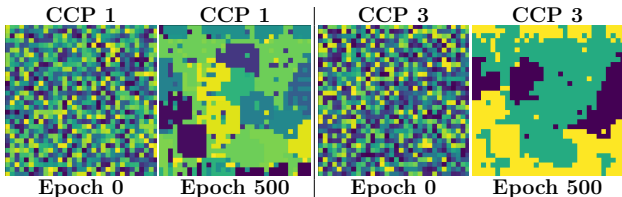
---

[3]We found it extremely hard to train, due to the huge memory footprint required, even for a shallow configuration.

1st CCP   2nd CCP   3rd CCP   4th CCP

Figure 4: Receptive fields learned during CIFAR10 training.

Table 5: Comparison of different graph corsening and filtering approches on CIFAR-10 and Cross Subject NTU RGB+D.

Filter	Coarsen	GAP	CIFAR	NTU-CS
Chebyshev [4]	Graclus	✗	78.15	74.85
GCN [14]	Graclus	✗	67.01	62.00
GAT	Graclus	✗	72.82	59.48
GAT	-	✓	66.39	26.74 [3]
CCP (ours)	CCP	✗	**84.4**	**80.1**

Table 6: Ablative results under different optimisations.

Loss $\mathcal{L}$	Gradient	CIFAR-10	NTU-CS
$\mathcal{L}_0 + \mathcal{L}_{\mathcal{K}}$ (Eq. 12)	-	**84.4**	**80.1**
$\mathcal{L}_0$	-	66.7	73.1
$\mathcal{L}_0$	$\delta\mathcal{L}_0/\delta U \leftarrow 0$	56.8	70.6
$\mathcal{L}_0 + \mathcal{L}_{\mathcal{K}}$	$\delta\mathcal{L}_0/\delta U \leftarrow 0$	83.8	78.6

CCP 1	CCP 1	CCP 3	CCP 3
Epoch 0	Epoch 500	Epoch 0	Epoch 500

Figure 5: Receptive fields arising from $\mathcal{L}_0$ minimisation on CIFAR-10.

The impact of the task-specific loss. We studied the contribution of the loss $\mathcal{L}_0$ (Eq. 12) and found three evidences supporting its beneficial effect: i) if only $\mathcal{L}_0$ is optimised, then suppressing its gradients on membership variables (i.e. cluster memberships are randomly fixed and cannot be changed during training) leads to poorer performances ($\mathcal{L}_0, \delta\mathcal{L}_0/\delta U \leftarrow 0$ against $\mathcal{L}_0$, Tab. 6); ii) when both objectives are optimised, discarding gradients of $\mathcal{L}_0$ on membership variables yields slightly degraded results ($\mathcal{L}_0 + \mathcal{L}_{\mathcal{K}}, \delta\mathcal{L}_0/\delta U \leftarrow 0$, against $\mathcal{L}_0 + \mathcal{L}_{\mathcal{K}}$, Tab. 6); iii) even when only optimising $\mathcal{L}_0$ we can observe the emergence of compact regions (i.e. clusters) in the clustering landscape (Fig. 5). Another evidence of this effect is the lowering of the $\mathcal{L}_{\mathcal{K}}$ when the network is optimised w.r.t. $\mathcal{L}_0$ (Fig. 6).

Effectiveness of the Ranking function. Finally, we conducted an ablation study for validating the effectiveness of the proposed within-cluster centrality measure in capturing shift-invariant structures on the graph. To this end, we compared it with a less principled criteria, involving a random permutation of the candidate nodes. Specifically, under the random setting, we still keep the definition of neighbourhood $\mathcal{N}_k^{(m+1)}$ given by Eq. 8. However, instead of sorting nodes in-
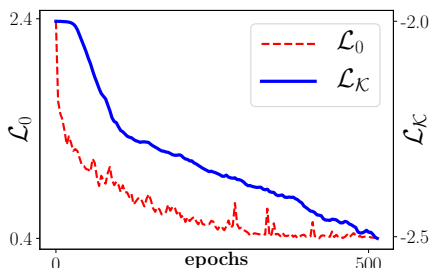
Figure 6: Loss landscapes under $\mathcal{L}_0$ minimisation.
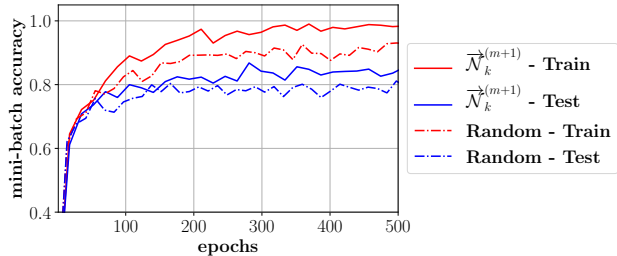
Order	Accuracy
Cluster Centrality	84.4
Random	80.3

Figure 7: Results from the ablation study conducted on CIFAR-10. The top picture shows test and training learning curve under both settings.

side it, we randomly sample a fixed permutation for each cluster, before the start of the learning. Without the sorting criteria, learnable kernels cannot rely on a coherent topological structure within their support, weakening the effect of weight sharing. We evaluated both of the policies on CIFAR-10, and reported our results in Fig. 7, in terms of learning curves and test error. The figure suggests that the sorting criterion indeed leads to a significant improvement in performance, due to a proper exploitation of weight sharing.

6 Conclusion

In this paper, we have proposed a novel approach for graph signal classification, leveraging both local and global structures, the latters arising from a multi-scale and hierarchical representation of the input signal. Our main contribution consists in a layer which performs a (soft) clustering step on the input graph and, accordingly, aggregates information within each cluster. Experiments show that our model consistently outperforms recent graph-based classification models in different data domains. The ablation study suggests that the proposed layer successfully exploit the weight sharing property in a graph convolutional architecture. For future works, we aim to generalise our architecture for vertex classification tasks and heterogeneous graphs, as well as further investigate the impact of supervision during clusters formation, which could lead to task-dependent pooling regions.

References

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Neural Information Processing Systems*, pages 1993–2001. Curran Associates, Inc., 2016.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*, 2016.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems*, 2016.

[5] Yong Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1110–1118, June 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, and Tara Sainath. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97, November 2012.

[8] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30, 10 2012.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[10] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, CS7, 1996.

[11] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Ahmed Sohel, and Farid Boussaïd. A new representation of skeleton sequences for 3d action recognition. *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 4570–4579, 2017.

[12] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, pages 1623–1631. IEEE, 2017.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[15] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Neural Information Processing Systems*, pages 1097–1105. Curran Associates, Inc., 2012.

[17] Chaolong Li, Zhen Cui, Wenming Zheng, Chunyan Xu, and Jian Yang. Spatio-temporal graph convolution for skeleton based action recognition. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[18] Zhouhan Lin, Roland Memisevic, and Kishore Reddy Konda. How far can we go without convolution: Improving fully-connected networks. *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2016.

[19] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, 2016.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *International Conference on Learning Representations*, 2013.

[21] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE International Conference on Computer Vision and Pattern Recognition*, page 3, 2017.

[22] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, ICML'16, pages 2014–2023. JMLR.org, 2016.

[23] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1944–57, 12 2007.

[24] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2016.

[25] Felipe Petroski Such, Shagan Sah, Miguel Domínguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D. Cahill, and Raymond W. Ptucha. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11:884–896, 2017.

[26] Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, ECCV'10, pages 140–153, Berlin, Heidelberg, 2010. Springer-Verlag.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[28] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *IEEE International Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 588–595, Washington, DC, USA, 2014. IEEE Computer Society.

[29] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007.

[30] Matthew Zeiler and Robert Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representations*, 2013.

[31] Tong Zhang, Wenming Zheng, Zhen Cui, and Yang Li. Tensor graph convolutional neural network. *CoRR*, abs/1803.10071, 2018.