Applied Intelligence (2022) 52:9033–9044
https://doi.org/10.1007/s10489-021-02973-4

Graph convolutional networks: analysis, improvements and results

Ihsan Ullah[1] · **Mario Manzo**[2] · **Mitul Shah**[3] · **Michael G. Madden**[3]

Accepted: 29 October 2021 / Published online: 16 November 2021

Abstract
A graph can represent a complex organization of data in which dependencies exist between multiple entities or activities. Such complex structures create challenges for machine learning algorithms, particularly when combined with the high dimensionality of data in current applications. Graph convolutional networks were introduced to adopt concepts from deep convolutional networks (i.e. the convolutional operations/layers) that have shown good results. In this context, we propose two major enhancements to two of the existing graph convolutional network frameworks: (1) topological information enrichment through clustering coefficients; and (2) structural redesign of the network through the addition of dense layers. Furthermore, we propose minor enhancements using convex combinations of activation functions and hyper-parameter optimization. We present extensive results on four state-of-art benchmark datasets. We show that our approach achieves competitive results for three of the datasets and state-of-the-art results for the fourth dataset while having lower computational costs compared to competing methods.

Keywords Graph convolutional networks · Clustering coefficients · Dimensionality reduction

1 Introduction

Traditionally in Machine Learning, data are represented as points in a vector space. In reality, however, structured data is omnipresent, and the ability to include structural information between points allows the model hypothesis language to be expanded relative to table-structured data, so that more expressive and accurate models can be learned from the data. Graphs are widely used to represent structured information using vertices/nodes and edges, including local and spatial information derived from the data, but most Machine Learning methods cannot handle graph-structured data.

Very often, learning objectives concern predictions about the properties of nodes in such graphs. For example, given a network that represents a human phenomenon, such as a mutual exchange of messages in a social network, the goal may be to predict which users belong to a community of common interests. Performing forecasting, especially in semi-supervised environments, has been a central focus of graph-based semi-supervised learning (*SSL*) [19]. The Graph-based SSL approach is similar to traditional SSL, where the training data consists of a small set of labelled data that is used as a reference in training for classifying the majority of the data, which is unlabelled. In mathematical notation, the structure described bthe graph is normally incorporated as an explicit regularizer which applies a sliding constraint on the node labels to be estimated.

Recently, Graph Convolutional Networks (GCNs) [6, 9] have been proposed; these are designed to work on graph-structured data with the deep neural network paradigm. In this paper, we consider the task of graph-based semi-supervised learning using GCNs. A GCN progressively estimates a transformation (also called an embedding) from graph space to vector space, and an aggregation

✉ Mario Manzo
mmanzo@unior.it

Ihsan Ullah
ihsan.ullah@ucd.ie

Mitul Shah
m.shah4@nuigalway.ie

Michael G. Madden
michael.madden@nuigalway.ie

[1] CeADAR Ireland's Center for Applied AI, University College Dublin (UCD), Dublin, Ireland

[2] Information Technology Services, University of Naples "L'Orientale", 80121, Naples, Italy

[3] Machine Learning and Data Mining Group, School of Computer Science, National University of Ireland, Galway, Galway, Ireland

of neighborhood nodes, in which a target loss function for backpropagation errors is adopted. The resulting node embedding represents an estimation for label scores on the nodes. Confidence based Graph Convolutional Networks (ConfGCN) were proposed [24] to obtain confidence estimates for label scores. These confidence scores can be used to understand the reliability of the estimated labels on a generic node.

In this context of enhancing GCN and ConfGCN, the aims of our paper are threefold:

1. Standard GCN and ConfGCN algorithms only make use of information relating to the degree of individual nodes (matrix $\tilde{D}$ in (1); see below in Section 3) to process the graphs. We introduce a measure that provides additional topological information called clustering coefficients.
2. In deep learning, common approaches to improving performance include adding additional layers or changing the regularization methods. In addition, the structure and layers of a network can be redesigned to obtain better results compared to existing models. To this end, we have combined GCN and Dense layers, and show that this provides better results compared to GCN alone and avoids the oversmoothing issue which can arise in the GCN when the depth is increased.
3. In the past few years, many researchers have worked on designing novel activation functions to help deep neural networks in converging and obtaining better performance.

   GCN and its variations employ a single logistic sigmoid or ReLU activation function. Sigmoid is affected by the saturation problem, whereas ReLU is affected by the dying ReLU problem [13] that results in reducing the ability to learn. In this paper, we used an efficient approach to learn during training from a combination of newer activation functions (such as *ReLU6*); our goal is to search through a space of activation functions defined by a convex combination of base functions.

To achieve the above objectives, we analyze GCN and ConfCGN to show the impact of proposed changes during training and testing. The paper is organized as follows: Section 2 gives an overview of related work. Section 3 provides an overview of GCN and ConfCGN. Section 4 explains the proposed enhancements. Then, Section 5 discusses the results achieved with proposed networks. Finally, Section 6 gives some future directions and concludes our paper.

2 Related work

Recent literature provides some interesting insights about the application of neural networks and data organized as graphs. In [9], a variant of convolutional neural networks, called Graph Convolutional Networks (GCNs), which operate directly on graphs, is presented. The main motivation for using a convolutional architecture is related to the localized first-order approximation of spectral graph convolutions. GCN works by linearly scaling node connections and adopting hidden layer representations that encode both the structure and features of graphs.

In [6], the authors generalize convolutional neural networks (CNNs) from low-dimensional regular grids to high-dimensional irregular domains represented in the form of graphs. The authors presents a CNN formulation in the spectral graph theory domain, which is useful to work as fast localized convolutional filters on graphs. The proposed formulation does not alter the computational complexity of standard CNNs, despite being able to process graph structures.

In [15], an enhanced version of work presented in [9] is introduced. It can work with syntactic dependency graphs in the form of sentence encoders that can extract latent feature representations of words arranged in a sentence. Moreover, the authors showed that the layers are complementary to LSTM layers.

In [25], a neural network architecture for inductive and transductive problems on graph-structured data is proposed. It is based on masked self-attentional layers, called graph attention networks (GATs). In a GAT, nodes can contribute to neighboring nodes' feature extraction and different weights are assigned to different nodes in a neighborhood, eliminating expensive matrix operations. In this way, several key challenges of spectral-based graph neural networks are addressed at the same time.

In [24], a modified version of [9] called the Confidence-based Graph Convolutional Network (ConfGCN) is introduced. It provides a confidence estimation about label scores, not available in GCN. ConfGCN adopts label score estimation to identify the influence of a node on its neighborhood during aggregation, thus acquiring anisotropic abilities. In [28], another modified version of [9] named Lovász Convolutional Networks (LCNs) is introduced. The network

can capture global graph properties through Lovász orthonormal embedding of the nodes.

In [1], a Diffusion-Convolutional Neural Network (DCNN) is described. Diffusion-convolution operation is useful to learn representations as an effective basis for node classification. The network includes different qualities such as latent representation for graphical data, invariance under isomorphism, polynomial-time prediction and learning.

In [4], possible generalizations of Convolutional Neural Networks (CNNs) to signals are defined for more general domains. In particular, two networks are described, one based upon a hierarchical clustering of the domain and another based on the spectrum of the graph Laplacian.

The networks can utilize convolutional operations with some parameters independent of the input size, resulting in efficient deep architectures. In addition, a deep architecture with low learning complexity on general non-Euclidean domains is introduced in [8] as an extension of Spectral Networks, by including a graph estimation procedure.

In [12], a graph partition neural network (GPNN) is described, which is an extension of graph neural networks (GNNs) that is applicable to large graphs. GPNNs combine local information between nodes in small subgraphs and global information between the subgraphs. Graphs are partitioned efficiently through several algorithms and, additionally, a novel variant for fast processing of large scale graphs is introduced. Similarly, in [10] the Gated Graph Sequence Neural Network (GGNN) is proposed, which is an extended version of the Graph Neural Network (GNN) [20]. It uses modified gated recurrent units and modern optimization techniques, and extends output sequences.

In the following section, we explain baseline GCN and ConfGCN networks.

3 Baseline networks

In this section, we first set out the basic notation and definition of graph structures, which are useful for understanding the node classification problem. Subsequently, we briefly introduce the Graph Convolutional Network (GCN) [9], and its enhancement the Confidence-based Graph Convolutional Network (ConfGCN) [24]. These two frameworks are compared and analysed in terms of limitations and differences. Finally, we propose a set of improvements and evaluate them experimentally.

3.1 Notation and problem statement

Graphs are data structures that can be useful for representing dynamic and interactive phenomena such as social networks, citation networks, chemical molecules, and recommendation systems. A graph is composed of two basic elements: nodes and edges. An edge represents the relationship between nodes. For example, considering a social network, nodes represent entities such as members, while edges describe relationships between those entities, such as friendships between members. Optionally, there may be multiple different types of nodes and edges, depending on the domain. A graph with only one type of node and one type of edge is termed homogeneous. A social network could be an example of a homogeneous graph, with nodes representing members and edges representing friendships, as there is just one type of node and one type of edge. Conversely, when two or more types of nodes and/or edges are present, the graph is termed heterogeneous. In a heterogeneous social network graph, edges could represent multiple types of connection (friendship, co-worker, collaboration, or degree of kinship). The nodes and edges could also include properties, attributes or features. In addition, graphs can be either directed (representing a specific relationship in one direction) or undirected (where relationship are in both directions). In this paper, the datasets we utilize contain data about citation networks where nodes are scientific publications and citation links are the edges between nodes.

In the following subsection, we will define the key terms and notations adopted for graphs and other variables used in this paper.

3.2 Graph convolutional networks

Graph Convolutional Networks (GCNs) [9] work on undirected graphs. Given a graph $G = (V, E, X)$,

$V = V_l \cup V_u$ is the set containing labeled (V_l) and unlabeled (V_u) nodes in the graph of dimension n_l and n_u, E is the set of edges, and $X \in \mathbb{R}^{(n_l+n_u)\times d}$ represents the input node features, the label of a node v is represented by a vector $Y_v \in \mathbb{R}^m$, belonging to m classes. In this context, the goal is to predict the labels, $Y \in \mathbb{R}^{n_l \times m}$, of the unlabeled nodes of G. To denote confidence, a label distribution $\mu_v \in \mathbb{R}^m$ and a diagonal covariance matrix $\Sigma_v \in R_{m\times m}$ of estimations are added. $\forall v \in V$, $\mu_{v,i}$ represents the score of label i on node v, while $(\Sigma_v)_{ii}$ represents the variance in the estimation of $\mu_{v,i}$. In other words, $(\Sigma_v^{-1})_{ii}$ is the confidence in $\mu_{v,i}$.

The node representation after a single layer of GCN can be defined as:

$$H = f((\tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}})XW) \tag{1}$$

Here, $W \in \mathbb{R}^{d\times d}$ includes the network parameters, A represents nodes adjacency, $\tilde{D}_{ii} = \sum_j (A + I)_{ij}$, and f is any activation function such as $ReLU$, $f(x) = max(0, x)$. (1) can be reformulated as:

$$h_v = f\left(\sum_{u\in N(v)} Wh_u + b\right), \forall v \in V \tag{2}$$

where $b \in \mathbb{R}^d$ represents bias, $N(v)$ includes nodes neighborhood of v in graph G including v and h_v is representation of node v.

The goal is to acquire multi-hop dependencies between nodes, different GCN layers can be superimposed over one another. The representation of the node v after k layers can be written as

$$h_v = f\left(\sum_{u\in N(v)} (W^k h_u^k + b^k)\right), \forall v \in V \tag{3}$$

where W^k and b^k represent the weight and bias parameters of GCN layer, respectively. However, increasing the depth

of GCN can give rise to an oversmoothing issue [5, 30], see Section 5.4.

3.3 Confidence based Graph Convolutional Networks

In [24], the Confidence-based Graph Convolutional Network (ConfGCN) framework is described. The authors define the influence score of node u relative to its near node v during the GCN process as follows:

$$r_{uv} = \frac{1}{d_M(u, v)} \tag{4}$$

where $d_M(u, v)$ represents the Mahalanobis distance between two nodes [17]:

$$d_M(u, v) = (\mu_u - \mu_v)^T(\Sigma_u^{-1} + \Sigma_v^{-1})(\mu_u - \mu_v) \tag{5}$$

Specifically, considering nodes u and v with label distributions μ_u and μ_v and covariance matrices Σ_u and Σ_v, r_{uv} gives greater importance to spatially close nodes that belong to same class, and reduces the importance of nodes with low confidence scores. This leads to inclusion of the anisotropic capability during neighborhood exploration. For a node v, (3) can be rewritten as:

$$h_v = f\left(\sum_{u \in N(v)} r_{uv} \times (W^k h_u^k + b^k)\right), \forall v \in V. \tag{6}$$

The final label prediction is obtained by (7) with K number of layers.

$$\tilde{Y}_v = softmax(W^K h_v^K + b^K), \forall v \in V \tag{7}$$

3.4 GCN versus ConfGCN

We analysed [9] and [24] and found the following differences between the two network types:

1. The major difference between both is that GCN implements a node-embedding projection from graph space to vector space to describe the neighborhood, while ConfGCN implements a confidence-based prediction scheme where the higher the confidence of neighboring nodes, the more influence those neighbouring nodes have on the label of the unknown nodes.
2. GCN implements the Chebychev polynomial method for computational cost reduction while ConfGCN uses loss smoothening, regularization and optimization for better efficiency. Compared to GCN, ConfGCN has better accuracy on the same datasets but has higher execution time.
3. GCN does not have constraints on the number of nodes that influence the representation of a given target node and each node is influenced by all the nodes in its k-hop neighborhood. On the other hand, in ConfGCN, the label confidences are used to ignore less confident nodes and nodes having higher confidence would be considered important.
4. ConfGCN adopts neighborhood label entropy to quantify label mismatch while GCN does not do this analysis. This helps ConfGCN in achieving better performance.
5. ConfGCN has higher computational cost than GCN. While calculating confidence value (4), the cost increases because it includes an additional exploration of the neighborhood equal to its width (number of nodes to consider).

Some of the limitations of GCN and ConfGCN include:

1. GCN [9] and ConfGCN [24] are not applicable to directed graphs. Neither of them support edge features and they are limited to undirected graphs (weighted or unweighted).
2. In GCN, locality is assumed for the nodes. As the size of the neighborhood grows, the algorithmic time and space complexity grow. For that reason, GCN cannot handle very dense graphs, compared to ConfGCN.
3. In ConfGCN, increasing the number of layers beyond a certain level reduces accuracy. This behavior is connected to the increase of influencing nodes with increasing layers beyond a certain number, which results in average/ambiguous information during aggregation. In addition, it results in creation of embeddings with almost similar values. This is also known as oversmoothing. This will be explained in detail in Section 5.4.

In the following section, we will explain our proposed enhancements and resulting network structures.

4 Proposed enhancements

Figure 1 shows an overview of our proposed framework. We propose four enhancements for both types of networks. The first enhancement is to change the hyper-parameters and training algorithm. The second and third are major enhancements: adding more structural information to adjacency matrix by utilizing clustering coefficients (CC) and introducing a canonical optimization technique (also referred to as convex optimization). The fourth concerns a combination of two base networks with the introduction of additional dense layers. All of these enhancements are applied to both the baseline networks. Below we will explain the design and implementation of our enhancements.

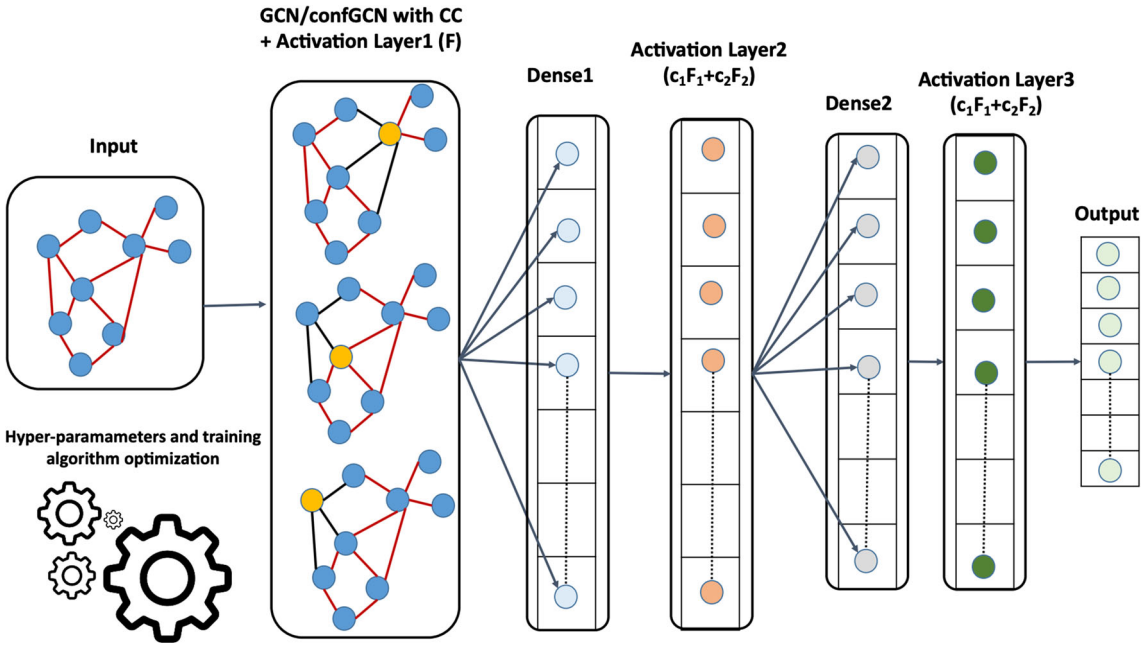

Fig. 1 Proposed approach for enhancing GCN/ConfGCN. Here, CC represents the clustering coefficient added after GCN/ConfGCN, F is the activation function in Layer 1, F_1 and F_2 represent the activation functions in Layers 2 and 3 respectively, and c_1 and c_2 are the two parameters for canonical convex optimization. In Layer 1, the black coloured edges indicate the three nodes (assuming kernel size=3) that are considered for a graph-convolution operation at a specific time. Finally, the optimization of the training algorithm and hyper-parameters are shown symbolically in the bottom-left

4.1 Optimizing hyper-parameters

First, we optimize the baseline networks by fine-tuning the hyper-parameters, including the activation function (AF), loss function (LF), and the number of nodes in each hidden layer. For possible AFs, we have explored the set $\{ReLU,\ ReLU6,\ ELU,$ and $SELU\}$. For loss functions, we evaluated both simple *cross entropy* and *cross entropy softmax V2*. To optimise the number of nodes, we considered the following possible numbers: {16, 32, 48, 64, 80, 96, 100, 112 and 200}. The objective is to optimize the parameters globally for an optimal combination that will lead us to the best performance in fewer time. In the remainder of this paper, the best network that results from exploring these combinations of hyper-parameters will be called the Optimized Graph Convolutional Network (OpGCN) and Optimized Confidence based Graph Convolutional Network (OpConfGCN), respectively.

4.2 Convex combination of activation functions

A standard neural network N_d is composed of a set of hidden layers d and a set functions L_i that lead to a final mapping $\overline{L}$ related to a problem to address: $N_d = \overline{L} \circ L_d \circ \cdots \circ L_1$, where $\circ$ indicates the composition of functions. Specifically, each hidden layer function L_i is composed of two functions, g_i and σ_i, which include parameters within the spaces H_{gi} and $H_{\sigma i}$. A remapping of the layer input neurons in form of activation function can be seen as: $L_i = \sigma_i \circ g_i$.

The learning process of L_i consists of an optimization procedure in the space $H_i = H_{\sigma i} \times H_{gi}$. In general, σ_i does not perform any role in the learning phase and $H_{\sigma i}$ is a singleton, therefore, $H_i = \{\sigma_i\} \times H_{gi}$. If we consider a fully-connected layer from $\mathbb{R}^{n_i}$ to $\mathbb{R}^{m_i}$ which adopts a ReLU AF, H_{gi} represents the set of all affine transformations from $\mathbb{R}^{n_i}$ to $\mathbb{R}^{m_i}$, then $H_i = ReLu \times Lin(\mathbb{R}^{n_i}, \mathbb{R}^{m_i}) \times K(\mathbb{R}^{m_i})$, where $Lin(A, B)$ and $K(B)$ are the sets of linear maps between A and B respectively, and the set of translations of B.

In this paper, we adopt a technique to define learnable activation functions [14] that can be used in all hidden layers of a GCN architecture.

The approach consists of a hypothesis space $H_{\sigma i}$ and is based on the following idea:

- Select a set of activation functions $F = \{f_1, \ldots, f_N\}$, in which elements can be adopted as base elements;
- Fix the activation function σ_i to combine in linear way the elements belonging to F set;
- Search for an optimal hypothesis space;
- Perform GCN optimization, where the hypothesis space of each hidden layer is $H_i = H_{\sigma i} \times H_{gi}$.

Table 1 Baseline network structure for enhancing with convex approach

Input Size	L1-Nodes	L1-ActivationFun	OutputNodes	loss function
1433	16	ReLU	3	Cross Entropy

Given a vector space V and a finite subset $A \subseteq V$ we can define the following subset of V, termed the convex hull as:

$$conv(A) := \{\Sigma_i c_i a_i | \Sigma_i c_i = 1, c_i \geq 0, a_i \in A\}; \quad (8)$$

$conv(A)$ is not a vector subspace of V and is a generic convex subset in V reducing to a simplex of dimension $(|A| - 1)$ when the elements of A are linearly independent. If we consider $F := \{f_0, f_1, \ldots, f_N\}$ the set of activation functions f_i, the vector space $\boldsymbol{F}$ is defined from F considering all linear combinations $\sum_i c_i f_i$ with $c_i \geq 0$, $\Sigma_i c_i = 1$. Note that, even though F is a spanning set of $\boldsymbol{F}$, it is not generally a basis; indeed $|F| \geq dim\, \boldsymbol{F}$. Based on previous definitions, we can now define the technique to build learnable activation functions as follows:

- Fix a finite set $F = \{f_1, \ldots, f_N\}$, where each f_i is a learnable activation function;
- Create an additional activation function $\overline{f}$ as a linear combination of all the $f_i \in F$;
- Select as the hypothesis space $H_{\overline{f}}$ the $conv(F)$ set;

From this approach, several combinations of activation functions in tuples were used e.g. as shown in F:

$$F := \{ReLU, ReLU6\} \quad (9)$$

where

$$ReLU6 = min(max(0, x), 6) \quad (10)$$

To summarise this subsection, for convex combination we have implemented two methods:

1. Taking two input layers of a network, use a different activation function for each of them and then apply any mathematical operation on the inputs, i.e. summation, subtraction, maximum, minimum and average values of output from the two input layers.
2. Examining those results, we observed that summation provides better results compared to other operations. Therefore, we applied the canonical form on the outputs, due to which the convex combination became: $conv(A) := c_1 ReLU6 + c_2 ReLU6$. The structure of the baseline network with optimized results is shown in Table 1, and its enhanced network structure is given in Table 2.

From here on, we will call the two enhanced versions, Convex Graph Convolutional Networks (ConvGCN) and Convex Confidence based Graph Convolutional Networks (ConvConfGCN).

4.3 Clustering coefficients

In (1), the adjacency matrix A that describes the topology of the network is a very significant part of both networks. The identity matrix I is added to A to remove zero values on the main diagonal. Our idea is to further add more information about nodes by introducing a particular property called Clustering Coefficients. In graph theory, the clustering coefficient describes the degree of aggregation of nodes in a graph. The measure is based on triplets of nodes. A triplet is defined as three connected nodes. A triangle can include three closed triplets, each one centered on one of the nodes. The two possible versions can be defined as the Global Clustering Coefficients (GCCs) and the local Clustering Coefficients (CCs) [16]. We adopted the second one which is defined as:

$$CC_i = \frac{\delta_i}{k_i(k_i - 1)} \quad (11)$$

where k_i is the degree of node i and δ_i is the number of edges between the k_i neighbors of node i. The measure is in the range $[0, 1]$, 0 if none of the neighbors of a node are connected and 1 if all of the neighbors are connected. Topological information is provided through CCs, which is connected to other structural properties [22], such as transitivity, density, characteristic path length, and efficiency, useful for representation in the vector space. In this work, we propose to replace the main diagonal of the matrix I with CC values. This new matrix is represented by ${}'C'_n$.

For a graph of $n \times n$ nodes the ${}'C'_n$ becomes:

$$C_n = \begin{bmatrix} CC_1 & 0 & 0 & \cdots & 0 \\ 0 & CC_2 & 0 & \cdots & 0 \\ 0 & 0 & CC_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & CC_n \end{bmatrix} \quad (12)$$

Table 2 Enhanced network structure for convex approach

In-Size	L1-Nodes	L1a-AF	L1b-AF	Out-Nodes	LossFun	c_1	c_2
1433	16	ReLU6	ReLU6	3	CrossEntropy	0.8	0.2

From now we will call the two enhanced versions with CCs as Clustering Coefficients Graph Convolutional Networks (CCGCN) and Clustering Coefficients Confidence based Graph Convolutional Networks (CCConfGCN).

The structure of the baseline network achieved high accuracy as presented in Table 1. The CC matrix was added to the Adjacency matrix while pre-processing of the input and the combined matrix was considered as input to the neural network. The new resulting matrix $'C'_n$ replaces the Identity matrix with the same size.

It is worth highlighting that the information relating to the CC is added to give more weight to the structural features of the graph. This will not result in low efficiency during the iterative update of the nodes. However, it can fail when the graph is sparse or poorly connected in some of its parts.

4.4 GCN and dense layer combination

Some deep learning research has shown that, rather than adding a new layer, if one can properly redesign existing layers, activation functions, regularization methods, etc., it can result in improved performance relative to the initial models [23]. To this end, We have added dense layers to GCN and created a network that gave us better results. A dense layer, also known as a fully-connected layer, is represented as:

$$y_u^{l_n} = f_{l_n}\left(\sum_{i=1}^{I}\left(\left(w_{(i,v)}^{l_n} \cdot y_{(i)}^{l_{n-1}}\right) + b_{(1,v)}^{l_n}\right)\right) \tag{13}$$

Here, $y_u^{l_n}$ represents the neuron at layer n, $w_{i,v}^{l_n}$ represents the weight (i, v) for that neuron multiplied with input neuron $y_i^{l_{n-1}}$, and $b_v^{l_n}$ represents that bias that is added to the weighted sum. The resultant weighted sum value is passed through an activation function f_{l_n}.

Table 3 shows the architecture of this network. We used this network on all four datasets. It shows the baseline models where the 'In-Nodes' represents the input nodes to a layer, 'Out-Nodes' represents the output nodes of a layer, 'AF' represents the activation function, whereas drop out rate is represented by 'DO'.

The baseline models in Table 3 are then enhanced using various combinations of changing the parameters and using proposed enhancements. After extensive experiments, their best results are shown in Table 5.

This combination provides a mixture of both GCN and Dense layers and results in better performance compared to individual GCN or Dense model.

In training, we used the Adam optimizer, as was used for all other networks. In each layer, we used the *ReLU6* AF. Therefore, from now on, we will call these two enhanced versions the Dense Graph Convolutional Networks (DGCN) and Dense Confidence based Graph Convolutional Networks (DConfGCN), respectively.

Table 3 Networks having both GCN and dense layer

Layer	In-Nodes	Out-Nodes	AF	DO
Input	1433	-	-	-
GCN	1433	32	Relu6	0.5
Dense-1	32	16	Relu6	0.5
Dense-2	16	32	Relu6	0.5
GCN	32	48	Relu6	0.5
GCN	48	7	-	0.5
Output	7	7	Softmax	-

5 Results

This section describes the results of applying the proposed enhancements to public datasets. We compare our results with the state-of-the-art competitors in the literature. The most common evaluation metrics such as accuracy and execution time are widely used in the literature, hence, we use these metrics evaluate our proposed network. In all the experiments for a convex combination of activation over all datasets, the optimal results that we achieved are with the following combination of F:

$$F := \{ReLU6, ReLU6\}. \tag{14}$$

5.1 Datasets

The concept of similarity between data can be expressed through the creation of graphs. Specifically, the edges describe a certain degree of similarity through associated edge weights. In the cases handled, the datasets adopted are stored in the form of a graph and, therefore, the processing phase was not carried out by us. Therefore, as required by GCN models, graphs were adopted directly as input for processing.

For performance evaluation, we make use of several state-of-art semi-supervised classification datasets. The datasets are Cora, Citeseer, Pubmed [21], and Cora-ML [3]. The setup is the same that was followed in [24]. We aim to classify documents into one of the predefined classes. Datasets represent citation networks in which each document is encoded using bag-of-words features with undirected edges between nodes. As an example, Fig. 2 left visualizes a Citeseer dataset, whereas right side shows its zoomed-in version to show few nodes and how they are connected with others. The dataset statistics are summarized in Table 4. Here, *Label Mismatch* is the fraction of edges

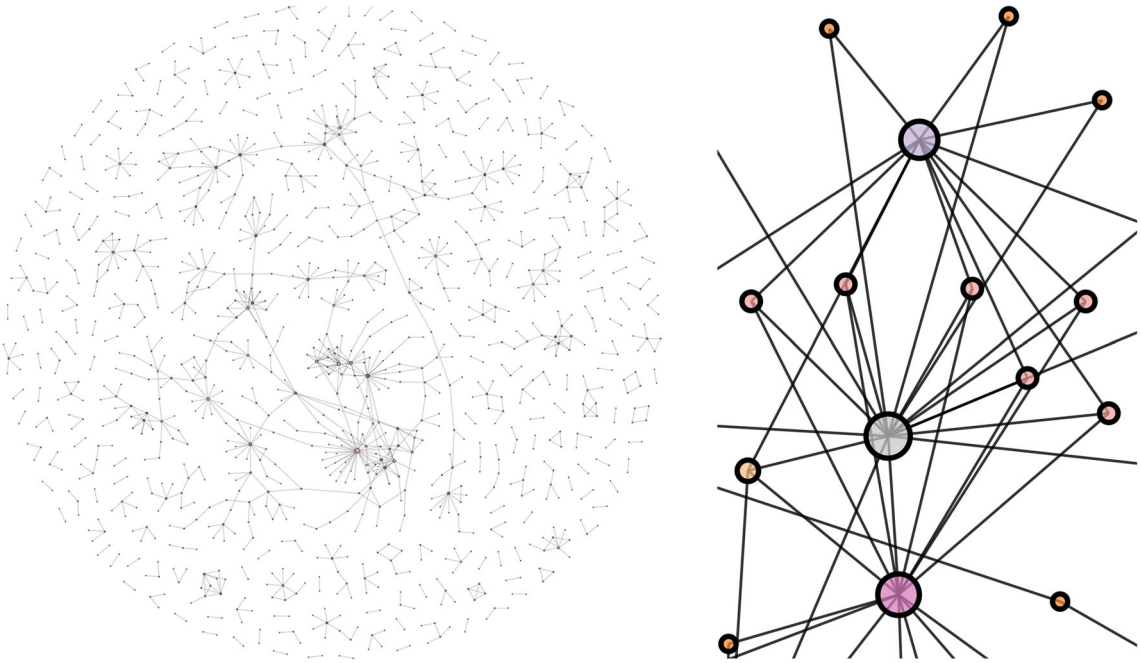

Fig. 2 Left side illustrates a structural graph of Citeseer dataset, whereas right side is the zoomed-in version of specific nodes and edges. As can be seen, the graph is very sparse and includes more density in specific areas as shown in the right side

between nodes with different labels in the training data. Except for Cora-ML, the datasets have quite low label mismatch rates.

5.2 Competing approaches

We compare our method with competitor approaches that can be divided into four groups. The first group includes networks based on extensions of the GCN. G-GCN [15] adopts edge-wise gating to remove noisy edges during aggregation. GAT [25] provides a method based on attention that gives different weights to different nodes by allowing nodes to attend their respective neighborhood. GAT network [25] learns both vertex and edge features to generalize. LGCN [7] works based on a learnable graph convolutional layer (LGCL), using 1D-CNN. Therefore, to make the data readable for the network, its LGCL converts the graph data into a fixed 1D structure by selecting a fixed number of neighbouring nodes from each feature based on their ranking. Fast-GCN [11] is an accelerated and optimized tool for constructing gene co-expression networks that can fully harness the parallel nature of GPU (Graphics Processing Unit) architectures. SGC [27] reduces complexity through the subsequent removal of non-linearities and collapsing the weight matrices between consecutive layers.

The second group includes networks based on extensions of the GNN [20]. GGNN [10] generalizes the RNN framework for graph-structured data applications. GPNN [12] adopts a partition approach to spread the information after the subdivision of large graphs into subgraphs.

The third group includes algorithms based on embeddings. SemiEmb [26] is a framework that provides semi-supervised regularization to improve training. DeepWalk [18] adopts random walks to learns node features. Planetoid [29] adopts a transductive and inductive approach for class label prediction using neighborhood information.

The fourth group includes other approaches. LP [31] is a label propagation algorithm that spreads labels information to the neighborhood following the proximity. ManiReg [2] provides geometric regularization on data. Feat [29] works based on node features ignoring the structure information.

Table 4 Dataset statistics

Dataset	Nodes	Edges	Classes	Features	Label Mismatch	$\frac{V_l}{V}$
Cora	2708	5429	7	1433	0.002	0.052
Cora-ML	2995	8416	7	2879	0.018	0.166
Citeseer	3327	4372	6	3703	0.003	0.036
Pubmed	19717	44338	3	500	0.0	0.003

5.3 Comparison

We have analyzed and explored the following activation functions: ReLU, ReLU6, ELU, and SELU. Of these, only ReLU6 was found to be the most suitable for the proposed model structure. Therefore, all the optimal results reported in this and following section uses ReLU6. Compared to GCN, ConfGCN has better accuracy on the same datasets but has higher execution time.

We have summarized experiments by showing the best results of all our enhancements for all the datasets in Table 5. We have been successful in getting the state-of-the-art result on one dataset as well as very close to the other three, as presented in Table 5. On the Cora_ML dataset, we achieved the current best accuracy of 86.9% ± 0.4 using the DConfGCN. This is the current state-of-the-art based on our knowledge as the relevant recent papers (LGCN, and Fast-GCN) did not report their results on the Cora_ML dataset. In case of the Citeseer dataset, we achieved our best result of 73.26%. This makes our accuracy with ConvConfGCN the second-best to date by only 0.3% less than LGCN.

We have achieved the 3^{rd} best accuracy for the Pubmed dataset i.e. 79.8% ± 0.4. Finally, on the Cora dataset, we achieved 82.1% ± 0.6 accuracy with ConvConfGCN, which is better than baseline GCN and ConfGCN by a slight margin, but at 4^{th} position overall in the list.

One of the reasons for not having the best result for the Citeseer, Cora, and Pubmed could be that the best reported results in LGCN [7] cannot be directly compared with ours as LGCN uses regular convolutional kernels in their network. Rather than designing new kernels to work on graph data, in LGCN the authors organized the graph data in a way that normal convolutional kernels can operate over it and learn features from them. Our enhancements and results are reported to provide a baseline for future works to be done in the field of SSL for graphs.

In Table 6, the execution time for the PubMed dataset is shown, where all runs were performed on the same computer.

The time (in seconds) per epoch varies for each dataset because the size of the features in each dataset varies. Overall, GCN and its enhancements are faster than confGCN and its enhancements. While optimizing based on hyper-parameters, we found that the main reduction in computational cost was due to usage of the *cross-entropy softmax V2* function rather than simple *cross-entropy*. Therefore, in all our subsequent experiments, we used this

Table 5 Performance comparisons of different methods on described datasets. The accuracy in brackets shows the single best result in the 100 runs

Method	Citeseer	Cora	Pubmed	Cora ML
LP [31]	45.3	68.0	63.0	-
ManiReg [2]	60.1	59.5	70.7	-
SemiEmb [26]	59.6	59.0	71.1	-
Feat [29]	57.2	57.4	69.8	-
DeepWalk [18]	43.2	67.2	65.3	-
GGNN [10]	68.1	77.9	77.2	-
Planetoid [29]	64.9	75.7	75.7	-
G-GCN [15]	69.6 ± 0.5	81.2 ± 0.4	77.0 ± 0.3	86.0 ± 0.2
GPNN [12]	68.1 ± 1.8	79.0 ± 1.7	73.6 ± 0.5	69.4 ± 2.3
GAT [25]	72.5 ± 0.7	83.0 ± 0.7	79.0 ± 0.3	83.0 ± 0.8
GCN [9]	69.4 ± 0.4	80.9 ± 0.4	76.8 ± 0.2	85.7 ± 0.3
OpGCN	70.1± 0.7	80.3± 0.4	79.1± 0.3	85.3± 0.4
ConvGCN	70.1 ± 0.3	80.1± 0.2	79.0± 0.2	84.3± 0.3
CCGCN	53.1 ± 0.6	55.3 ± 2.4	71.1 ± 0.7	63.3 ± 0.4
DGCN	70.9 ± 0.7	82.1 ± 1.2 (83.1)	79.10 ± 0.4	86.3 ± 0.3
ConfGCN [24]	72.7 ± 0.8	82.0 ± 0.3	79.5 ± 0.5	86.5 ± 0.3
OpConfGCN	70.1 ± 1.4	80.9 ± 0.8	**79.8 ± 0.4** (80.1)	84.6 ± 0.5
ConvConfGCN	**73.1± 0.2** (73.26)	**82.1± 0.6**	**79.8± 0.4** (80.1)	86.4± 0.3
CCConfGCN	70.8 ± 0.3	82.1 ± 0.6	78.2 ± 0.4	83.4 ± 0.5
DConfGCN	58.03 ± 0.9	81.0 ± 1.4	78.8 ± 0.6	**86.9 ± 0.4** (87.01)
SGC [27]	71.9 ± 0.1	81.0 ± 0.0	78.9 ± 0.0	-
LGCN [7]	73.4	83.3	79.7	-
Fast-GCN [11]	-	86	88	-

Table 6 Execution time on pubmed dataset

Method	Time (sec)
GCN	0.8
OpGCN	0.415
ConvGCN	0.585
CCGCN	0.417
DGCN	0.662
ConfGCN	1.344
OpConfGCN	1.93
ConvConfGCN	1.96
CCConfGCN	1.93
DConfGCN	1.99

loss function. The best found network in terms of execution time is OpGCN.

The PubMed dataset is a denser and more complex graph to classify. OpConfGCN and ConvConfGCN provide better results because they are two versions oriented to the optimization of performances compared to CCConfGCN and DConfGCN that are oriented to the identification of the structural information within the graph. The Cora and Cora-ML datasets have fewer nodes and more edges and classes, which makes the classification phase more complex. Nonetheless, due to the dense layers in DConfGCN and DGCN, good results are achieved.

The Citeseer dataset is the simplest of the datasets with the fewest edges. It can be seen that the trend of the results is much lower than the others. Out of the four proposed approaches, only ConvConfGCN shows good accuracy on Citeseer. We conclude that ConvConfGCN is the best proposed model among all we evaluated, based on its optimal performance on three out of the four datasets as shown in bold in Table 5.

5.4 Over-smoothing in GCN

GCN includes the message-passing mechanism to exploit the information encapsulated in the graph structure. However, this can lead to limitations when combined with the depth of the neural network. The message-passing mechanism provides two main functions: (1) aggregation, which collects spatial neighborhood information in the graph structure and node features; (2) updating, to combine them to update the representation of the node. This mechanism works to represent interacting nodes in a similar way. The search for an expressive and representative model for the structure of the graph, through the addition of more deep graph convolutional layers, could produce repetitive nodes in the new embedding for the new deep layer. This behavior is called over-smoothing. An important aspect concerns the quantification of over-smoothing, adopted for tracking during the training of the model. This is an approach adopted as a form of numerical penalty by adding it as a regularization term in the objective function.

In [5] Mean Average Distance (MAD) and Mean Average Distance Gap (MAD-Gap) are introduced, to measure the smoothness and over-smoothness of the graph nodes representations. MAD and MADGap calculate the Mean Average Distance among node representations, also known as embeddings, in the graph with a purpose to show the smoothing as a natural effect of adding more layers to the neural model.

In [30] Group Distance Ratio, which computes the ratio of two average distances, is introduced. First, nodes are associated with their specific group label. Then, to construct the nominator of the ratio, the pairwise distance between every two groups of nodes is calculated, averaged over the resulting distances. For the denominator, the average distance for each group is calculated. Although the quantification phase can be performed, it is not enough to add the metrics described as a regularization term.

The remaining problem is that calculating the metrics at each iteration of training may be computationally expensive, since it is necessary to access all the training nodes of the graph. For this reason, the problem of over-smoothing is addressed with different solutions that affect training. In [30] the neural model assigns nodes to groups and normalizes them independently to generate a new embedding matrix for the next layer. This additional layer is built to optimize the Group Distance Ratio. In fact, normalizing embedded nodes within a group makes their representation similar, and this scaling, using trainable parameters, provides varied embedding belonging to different groups labels. In our case, rather than increasing the depth of graph convolution layers, we added dense layers after the first graph convolution layer which avoids the creation of almost similar embedding between nodes. Hence, we avoid the over smoothing issue that arises from the depth of graph convolution layer networks.

6 Conclusions

We have presented enhancements of GCN and ConfGCN for the task of semi-supervised learning with graph convolutions. In particular, we have focused on four main changes: parameter configuration; adding more structural information to adjacency matrices for graph representation; convex optimization of activation functions; and combination of base networks with dense layers. Through these enhanced graph networks, we have been able to show that the addition of the layers can help to increase accuracy, unlike in the baseline networks where addition of new layers reduces accuracy. Currently, all of the Graph

Convolutional Layers use 1D convolutions to operate the network, but there can be 2D or 3D weighting schemes that can be implemented on the concurrent networks. GCN was initially proposed as a novel approach for SSL, and implemented the layer-wise propagation rule, while ConfGCN was subsequently proposed as a network that estimates label scores with labels' confidences. We have proposed six different network configurations and validated them on four benchmark datasets. The selection of optimal parameters is done through a grid search for exploring their complete space. This helps in successfully achieving high accuracy and low execution times for all networks in all four datasets.

Acknowledgements The first two authors acknowledge the guidance and supervision of their late Prof. Alfredo Petrosino. May he rest in peace. This research is supported by the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No. 700264, ROCSAFE.

References

1. Atwood J, Towsley D (2016) Diffusion-convolutional neural networks. In: Advances in neural information processing systems, pp 1993–2001
2. Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7:2399–2434
3. Bojchevski A, Günnemann S (2018) Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In: International conference on learning representations, pp 1–13
4. Bruna J, Zaremba W, Szlam A, Lecun Y (2014) Spectral networks and locally connected networks on graphs. In: International conference on learning representations (ICLR 2014), CBLS, 2014
5. Chen D, Lin Y, Li W, Li P, Zhou J, Sun X (2020) Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 3438–3445
6. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852
7. Gao H, Wang Z, Ji S (2018) Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, pp 1416–1424
8. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. CoRR arXiv:1506.05163
9. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5Th international conference on learning representations, ICLR 2017. Conference Track Proceedings, Toulon
10. Li Y, Tarlow D, Brockschmidt M, Zemel RS (2016) Gated graph sequence neural networks. In: 4Th international conference on learning representations, ICLR 2016. Conference Track Proceedings, San Juan
11. Liang M, Zhang F, Jin G, Zhu J (2015) Fastgcn: a gpu accelerated tool for fast gene co-expression networks. PloS one 10(1):e0116,776
12. Liao R, Brockschmidt M, Tarlow D, Gaunt AL, Urtasun R, Zemel RS (2018) Graph partition neural networks for semi-supervised classification. In: 6Th international conference on learning representations, ICLR 2018. Workshop Track Proceedings, Vancouver
13. Lu L, Shin Y, Su Y, Karniadakis GE (2020) Dying relu and initialization: Theory and numerical examples. Commun Comput Phys 28(5):1671–1706. https://doi.org/10.4208/cicp.OA-2020-0165. http://global-sci.org/intro/article_detail/cicp/18393.html
14. Manessi F, Rozza A (2018) Learning combinations of activation functions. In: 2018 24Th international conference on pattern recognition (ICPR). IEEE, pp 61–66
15. Marcheggiani D, Titov I (2017) Encoding sentences with graph convolutional networks for semantic role labeling. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp 1506–1515
16. Opsahl T (2013) Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. Soc Netw 35(2):159–167
17. Orbach M, Crammer K (2012) Graph-based transduction with confidence. In: Joint european conference on machine learning and knowledge discovery in databases. Springer, pp 323–338
18. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 701–710
19. Rozza A, Manzo M, Petrosino A (2014) A novel graph-based fisher kernel method for semi-supervised learning. In: 2014 22Nd international conference on pattern recognition. IEEE, pp 3786–3791
20. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
21. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93
22. Strang A, Haynes O, Cahill ND, Narayan DA (2018) Generalized relationships between characteristic path length, efficiency, clustering coefficients, and density. Soc Netw Anal Min 8:14
23. Ullah I, Petrosino A (2016) About pyramid structure in convolutional neural networks. In: 2016 International joint conference on neural networks (IJCNN), pp 1318–1324. https://doi.org/10.1109/IJCNN.2016.7727350
24. Vashishth S, Yadav P, Bhandari M, Talukdar P (2019) Confidence-based graph convolutional networks for semi-supervised learning. In: The 22nd international conference on artificial intelligence and statistics, AISTATS 2019, Naha, pp 1792–1801
25. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: 6Th international conference on learning representations, ICLR 2018. Conference Track Proceedings, Vancouver
26. Weston J, Ratle F, Mobahi H, Collobert R (2012) Deep learning via semi-supervised embedding. In: Neural networks: Tricks of the trade. Springer, pp 639–655

27. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on machine learning, pp 6861–6871
28. Yadav P, Nimishakavi M, Yadati N, Vashishth S, Rajkumar A, Talukdar P (2019) Lovász convolutional networks. In: The 22nd international conference on artificial intelligence and statistics, pp 1978–1987
29. Yang Z, Cohen WW, Salakhutdinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48. JMLR.org, pp 40–48
30. Zhou K, Huang X, Li Y, Zha D, Chen R, Hu X (2020) Towards deeper graph neural networks with differentiable group normalization. Adv Neural Inf Process Syst:33
31. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th International conference on Machine learning (ICML-03), pp 912–919