Multi-view adaptive graph convolutions for graph classification

Nikolas Adaloglou, Nicholas Vretos, and Petros Daras

The Visual Computing Lab - Information Technologies Institute, Centre for Research and Technology Hellas, 57001 Thessaloniki, Greece
{adaloglou,vretos,daras}@iti.gr

Abstract. In this paper, a novel multi-view methodology for graph-based neural networks is proposed. A systematic and methodological adaptation of the key concepts of classical deep learning methods such as convolution, pooling and multi-view architectures is developed for the context of non-Euclidean manifolds. The aim of the proposed work is to present a novel multi-view graph convolution layer, as well as a new view pooling layer making use of: a) a new hybrid Laplacian that is adjusted based on feature distance metric learning, b) multiple trainable representations of a feature matrix of a graph, using trainable distance matrices, adapting the notion of views to graphs and c) a multi-view graph aggregation scheme called graph view pooling, in order to synthesise information from the multiple generated "views". The aforementioned layers are used in an end-to-end graph neural network architecture for graph classification and show competitive results to other state-of-the-art methods.

Keywords: Distance metric learning, graph neural networks, graph classification, multi-view, view pooling, adaptive graph convolution

1 Introduction

Graph theory enabled us, among other things, to powerfully represent relationships between data. Due to their avast application field, ranging from biological structures to modern social networks, graph representations, along with the recently exploding field of Graph Neural Networks (GNNs), assisted us to re-address challenging graph related tasks under the prism of neural networks. To that end, the task of adapting operations from classical convolutional neural networks (CNNs), such as convolution and pooling, to non-Euclidean domains (e.g., graphs and manifolds), gave rise to the emerging field of geometric deep learning [4].

A typical CNN consists of different layers of convolutions and pooling that are usually combined in a sequential manner [26], [13], [24]. The idea of adapting these key operations of CNNs to irregular grids is not new. Until recently, a considerable amount of research effort has been invested to adapt the classical convolution layers in many different ways towards graph-based convolutions

[5] - [12] giving birth to the Graph-based Neural networks. GNNs have shown promising results on representation learning mainly due to the graphs' ability to encode the structure of the data, in contrast to regular grids [39], by encapsulating information in the graph's vertices [28].

Many different graph theoretical domains have made use of the above ideas to this point. Among them, graph classification has attracted a lot of attention lately due to its wide application areas [15], [42], [49]. Graph classification is considered as the task to find a mapping $f : G \rightarrow T$ that maps each graph in G to a label from a set of target labels T. Nonetheless, in graph classification, the continuous-valued vertex attributes, such as the ones that can be found in biological measurements [11], are rarely exploited.

In this paper, a flexible multi-view GNN architecture for the task of graph classification is proposed, which propagates both structural and signal information throughout the network. To that end, a novel multi-view graph convolutional layer is presented making use of a hybrid Laplacian, which combines information of the feature space with the structure of the input graph. Subsequently, the use of multiple trainable distance metrics is proposed to cope with training instabilities and overfitting, since it has been proven that single distance metric learning can be prone to overfitting [17]. Finally, by applying a spectral filtering on the graph's signal with different hybrid Laplacians associated to each "view", a per "view" projected signal is calculated. Similar to [41], a batch normalized max view pooling layer is proposed to aggregate the different per "view" projected signals. This leads our model to learn from compact generalized representations.

The remainder of this paper is organized as follows: in Section 2, related work is briefly described. In Section 3, the different components of the proposed methodology are outlined as well as their application to a GNN architecture is detailed. In Section 4, the performed experimental results in several and diverse datasets are reported. Finally, conclusions are drawn in Section 5.

2 Related work

Graph convolutional filters can be divided in two main approaches: a) the spatial and b) the spectral ones. Spatial approaches operate directly in the vertices' neighborhoods while spectral approaches operate on the graph's Laplacian eigenspace [56]. In PSCN [34], the authors presented a generalized spatial approach in order to generate local normalized neighborhood regions while in DGCNN [53] an end-to-end architecture is proposed that keeps extra vertex information through sorted graph representations from spatial graph convolutions. In order to process graph data with classical CNNs, KGCNN [35] uses a 2-step approach, starting from extracting patches from graphs via common community detection algorithms, embedding them with graph kernels and finally feed them in a classical CNN. Furthermore, another approach proposed in GIN [47] developed an architecture based on Weisfeiler-Lehman test [38], stating that its discriminating power is equal to the power of the Weisfeiler-Lehman isomorphism test. Finally, in

DGK [50] the authors proposed to leverage the dependency information between sub-structures used in graph kernels by learning their latent representations.

A major breakthrough in graph spectral convolutions was proposed in [10]. Therein, the authors showed the ability of the spectral GNN to extract local features through graph convolutional layers. Moreover, they proposed an efficient pooling strategy on graphs, based on a binary tree structure of the rearranged vertices. Several recently developed graph pooling methods attempt to reduce the number of vertices [52], [10], [6], [14], producing coarser and sparser representations of a graph. However, for graph classification, where the number of vertices is relatively small, it is difficult to design such models, while sometimes it could lead to unstable training behaviour [52]. Furthermore, the coarsened vertices are not always arranged in a meaningful way. Moreover, graph coarsening processes are usually keeping a fixed number of vertices, which results in training and inference bias [10].

There exist various attempts that define pooling operation on graphs. In Graph U-Nets [14], the authors propose a max pooling scheme using a trainable projection vector of the graph signal and then assign the corresponding indices to the adjacency matrix. In DIFFPOOL [52], the authors try to learn the hierarchical structure through a trainable cluster assignment matrix, providing a flexible architecture that can be applied in a plethora of GNNs. In the same direction, CLIQUEPOOL [30] also attempts to coarsen the vertices of graph by aggregating maximal cliques. In another recent work called SAGPool [27], the authors define an alternative graph pooling method based on self-attention that considers both the vertex features and the graph structure. Finally, in HO-GNN [32], a GNN architecture is used that takes into account higher-order graph structures at multiple scales.

Some recent approaches have attempted to design architectures that explore different graph structures. In this direction, MGCN [22] is developed to capture multi-relational graph relationships through multi-dimensional Chebyshev polynominals. In GCAPS-CNN [45], the authors introduced the notion of capsules in spectral GNNs, in order to encapsulate local higher order statistical information per vertex feature. In a proximal work, CAPS-GNN [46] explores additional vertex information by computing hand-crafted statistics. However, both approaches compute extra vertex information explicitly, while they do not involve any trainable components.

Moreover, one of the first works that introduced single supervised metric learning on graph related tasks is [29], which is closer to our approach. Nonetheless, this work explores only one graph structure through the graph's Laplacian as opposed to our work where multiple graphs are used. Besides, distance metric learning has also been applied in [25] to resolve the irregular graph matching task, using spectral graph convolutional networks in the field of biomedical applications. Finally, WKPI [55] employ a metric learning scheme, which learns a positive semi-definite weighted-kernel for persistence summaries from labelled data.

In [41] the notion of "view" was introduced, where each "view" represents a $2D$ projection of a $3D$ object from a different angle, in order to produce

informative representations for the tasks of object recognition and retrieval. In [54], the authors used the idea of "views" to construct different brain graphs, where structure was determined by a structural MRI scan and the feature matrices were calculated from multiple tractography algorithms based on diffusion MRI acquisition, to fuse information scattered in different medical imaging modalities for pairwise similarity prediction. However, none of the above view-based approaches introduced a trainable task-driven generation of views, as is the case in our approach.

The majority of graph convolution methods are based on projected graph features into a single graph structure. The motivation behind our approach is that "views" are artificially created from a common input. These, can extract different learning representations, the same way 3D objects are projected onto 2D images and processed independently. Nevertheless, the fact that the generated "views" are trainable, as opposed to previous approaches, is a novel and unexplored domain.

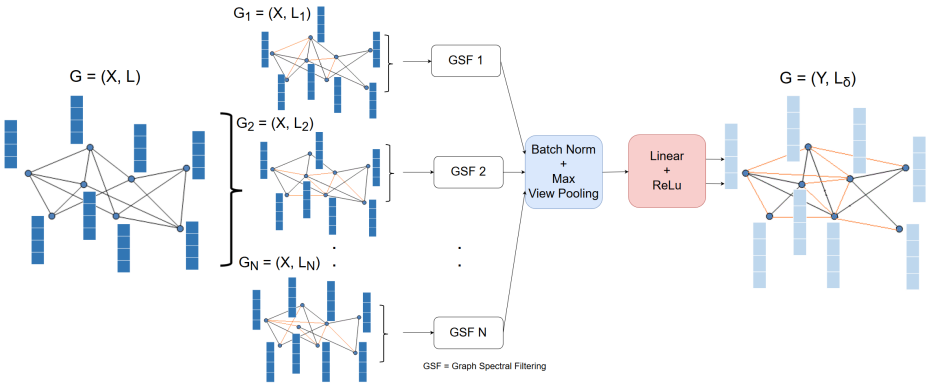

Fig. 1. An overview of the proposed layers (multi-view graph convolution and view pooling)

3 Proposed method

The proposed work is inspired from [41] and [54] in using the notion of "views" produced by a shared input (i.e. human vision). The key idea of the proposed multi-view graph convolution (MV-GC) layer is that in each "view" a different n complete graph (K_n) is constructed from the pairwise Mahalanobis distances [8] of the vertices in the feature space. Each "view" is thus encapsulating a different relation between vertices in a distance metric learning context. This information is encoded in the MV-GC layer via the so-called hybrid Laplacian. The hybrid Laplacian is a linear combination of the input graph's Laplacian

and a non-Euclidean distance-based Laplacian term, called from now on the "view" Laplacian, derived from the aforementioned K_n graph. As a result, the spectral graph convolution is approximated by a different hybrid Laplacian eigenspace, producing multiple learnable projections of the same graph signal. Therefore, applying batch normalization as in [19] and a view-wise max pooling operation, called from now on "view pooling" layer (VPOOL), an aggregated graph output signal is produced. In contrast to other existing methods [21] that keep the structure of the graph constant throughout the network, in the proposed approach the intrinsic graph's structure is altered during the learning process of the GNN. The whole process can be illustrated in Figure 1.

3.1 Notations and prerequisites

To ameliorate the readability of this paper, a summary of the notations and some initial background theory to be used is provided. A graph G is defined as $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of vertices and $\mathbf{E}$ a set of edges, with $e_{ij} \in \mathbf{E}$ defined as (v_i, v_j) and v_i, $v_j \in \mathbf{V}$. An alternative representation of a graph of n vertices, taking into account the vertices' feature space can be $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_n] \in \mathbb{R}^{n \times d}$ is the graph's feature matrix where each row contains the corresponding vertex d-dimensional feature vector $\mathbf{x}$. The feature matrix is also called the signal of graph G. Consequently, the graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} \in R^{n \times n}$ is the diagonal degree matrix with $(\mathbf{D})_{ii} = \sum_j (\mathbf{A})_{ij}$. Finally, a commonly used normalized version of the Laplacian operator is defined as follows [7]:

$$\mathbf{L}_{norm} \triangleq \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{1}$$

$\mathbf{L}_{norm}$ is a real symmetric, positive semi-definite matrix, with a complete set of orthonormal eigenvectors and their associated ordered real non-negative eigenvalues λ_i, with $i = [0, n-1]$. In addition, $\mathbf{L}_{norm}$ is utilized in graph spectral convolution, due to the fact that its eigenvalues lie in the range $[0, \lambda_{max}]$, with $\lambda_{max} \leq 2$ and is widely adopted in most spectral GNN architectures.

The generalized Mahalanobis distance, with transformation matrix $\mathbf{M}$, for any positive semi-definite matrix and vectors $\mathbf{x}$, $\mathbf{y}$, is defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y})} \quad \forall \mathbf{x}, \mathbf{y}, \tag{2}$$

where ^{T} denotes the transpose operation. The above equation can represent a quantitative measure of dissimilarity between vectors $\mathbf{x}$ and $\mathbf{y}$, which are drawn from the same distribution with covariance matrix $\mathbf{C} = \mathbf{M}^{-1}$. For $\mathbf{M} = \mathbf{I}$, eq. 2 reduces to the Euclidean distance. The latter assumes that the variances among different dimensions to be one and covariances to be zero, which is rarely the case in real life applications.

Furthermore, let *diag* be the operator that maps the main diagonal of an $c \times c$ matrix into a vector $\in \mathbb{R}^c$ and $\mathbf{1}_d$ be a d element vector of ones. A useful

equation, derived from the properties of the Hadamard product [18], denoted as $\odot$, is that for given matrices $\mathbf{A}, \mathbf{B} \in R^{c \times d}$:

$$diag(\mathbf{A}\mathbf{B}^T) = (\mathbf{A} \odot \mathbf{B})\mathbf{1}_d \in \mathbb{R}^c. \tag{3}$$

Batch normalization across a set of N representations (or views) of X_v, where $\mathbf{X}_v \in R^{n \times d}$, can be defined as an element-wise operation as:

$$BN(x_v) = \gamma_v(\frac{x_v - \mu(\mathbf{X}_v)}{\sigma(\mathbf{X}_v)}) + \beta_v, \tag{4}$$

where x_v denotes the elements of $\mathbf{X}_v$, while γ_v and β_v are trainable scalars that correspond to a single view. $\mu_v(\mathbf{X}_v)$ is the mean value of the elements of matrix $\mathbf{X}_v$ and σ their standard deviation.

Similarly, given N representations $\mathbf{Z} = [\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_N]$ that may correspond to "views", max view-wise pooling [41] can be computed by simply taking the maximum across the different views:

$$(\mathbf{Z})_{i,j} = \max((\mathbf{X}_1)_{i,j}, (\mathbf{X}_2)_{i,j}, .., (\mathbf{X}_N)_{i,j}) \tag{5}$$

3.2 Construction of the hybrid Laplacian

For each graph $G = (\mathbf{A}, \mathbf{X})$, N "views" are initially created. Each "view" is associated to a feature transformation matrix $\mathbf{M}_v \in \mathbb{R}^{d \times d}$ with $v \in [1, .., N]$. Each $\mathbf{M}_v$ is a positive semi-definite matrix defined as $\mathbf{M}_v = \mathbf{Q}_v\mathbf{Q}_v^T \in \mathbb{R}^{d \times d}$, where $\mathbf{Q}_v$ is randomly initialized at the beginning of the learning process. In particular, taking the generalized Mahalanobis distance between feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ of the i_{th} and j_{th} vertices respectively, it follows that:

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sqrt{\left(\mathbf{x}_i - \mathbf{x}_j\right)^T \mathbf{M}_v \left(\mathbf{x}_i - \mathbf{x}_j\right)} \quad \forall \mathbf{x}_i, \mathbf{x}_j. \tag{6}$$

Moreover, a feature difference matrix $\mathbf{F} \in \mathbb{R}^{c \times d}$, with $c = \frac{1}{2}n(n-1)$, is defined as the matrix of all feature differences between unique pairs of vertices $(v_i, v_j) \in \mathbf{V}$, without taking into account their connectivity in $\mathbf{E}$. Based on all the above, eq. 6 can be written, using eq. 3 as:

$$\mathbf{d}_v = diag(((\mathbf{F}\mathbf{M}_v\mathbf{F}^T) \odot \mathbf{I}_c)^{1/2}) = ((\mathbf{F}\mathbf{M}_v \odot \mathbf{F})\,\mathbf{1}_d)^{1/2} \quad \in \mathbb{R}^c \tag{7}$$

The above step is necessary as it significantly reduces the required memory (see Sec. 3.4), and allows the network to be trained in a end-to-end way. During the learning process, it is attempted to learn optimized supervised distance metrics, from the given data, that minimize the GNN's graph classification loss, through backpropagation. The unique distance pairs of $\mathbf{d}_v$ are placed back in a "view" distance matrix $\mathbf{H}_v \in \mathbb{R}^{n \times n}$. The per "view" similarity matrix, using the Gaussian kernel, is defined as:

$$\mathbf{S}_v = exp\left(-\mathbf{H}_v/2\sigma^2\right), \tag{8}$$

where σ is the standard deviation and the hybrid Laplacian, $\mathbf{L}_h$ with $h \in [1, .., N]$, is calculated as:

$$\mathbf{L}_h = \mathbf{L}_{in} + \alpha \mathbf{L}_v, \tag{9}$$

with $\mathbf{L}_v = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{S}_v\mathbf{D}^{-1/2}$ and $\mathbf{L}_{in}$ being the input graph's Laplacian. The contribution of each "view" is controlled by the scalar value α, which is set to 1, to avoid extra hyperpameter tuning. $\mathbf{L}_v$ represents a n complete graph (K_n) with $c = \frac{1}{2}n(n-1)$ weighted edges based on the feature differences and the trainable distance matrix $\mathbf{M}_v$. Thus, a dense hybrid Laplacian is produced for each "view", encoding different relation types between vertices.

For each view v, we compute L_v by taking into account the different trainable Q_v, based only on the features of the graph; thus computing different distances. However, since we want to maintain the original graph's structure, we propagate input-graphs' connectivity by adding L_{in} to L_v. Doing so, we also imitate the successful Res-Net architecture that propagates previous layer outputs to subsequent layers (the so-called "identity connection shortcut"). All the above contribute to faster convergence and encounter for the vanishing gradient problem in the early layers.

3.3 Construction of the graph output signal

As it is described in [10], convolution of graph signal $\mathbf{X}$ can be defined in the spectral domain and can therefore be approximated by applying a filter g_θ in the eigenvalues of the Laplacian of a graph as:

$$\mathbf{Y} = g_\theta(\mathbf{L})\mathbf{X} = g_\theta(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)\mathbf{X} = \mathbf{U}g_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X}, \tag{10}$$

where $\mathbf{U}$ represents the eigenvectors of $\mathbf{L}$ and $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the corresponding eigenvalues. In order to make the spectral filter independent of the graph size and also restrict it in a local graph's region (simulating the CNNs' localization property), the filter is usually modeled as a polynomial function of powers of $\mathbf{\Lambda}$. This expansion can be approximated by using the recurrent Chebyshev expansion to speed up the computations as:

$$g_\theta(\mathbf{\Lambda}) = \sum_{p=0}^{K-1} \theta_p \mathbf{\Lambda}^p = \sum_{p=0}^{K-1} \theta_p \mathbf{T}_p(\tilde{\mathbf{\Lambda}}) \tag{11}$$

with $\mathbf{T}_p(\tilde{\mathbf{\Lambda}}) = 2\tilde{\mathbf{\Lambda}}\mathbf{T}_{p-1}(\tilde{\mathbf{\Lambda}}) - \mathbf{T}_{p-2}(\tilde{\mathbf{\Lambda}})$, $\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I}$ and $\boldsymbol{\theta} \in \mathbb{R}^K$ is the vector of the spectral coefficients. Given a decomposition, $\mathbf{L}^p = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)^p = \mathbf{U}\mathbf{\Lambda}^p\mathbf{U}^T$, g_θ can be parametrized as a polynomial function of $\mathbf{L}$, that can be calculated from a recurrent Chebyshev expansion of order K. Using the re-scaled Laplacian $\tilde{\mathbf{L}}_h = \frac{2}{\lambda_{max}}\mathbf{L}_h - \mathbf{I}_n$ the computation of decomposition is avoided. Each Laplacian power can be interpreted as expressing the graph constructed from the p-hops thus providing the desired localization property. In our approach, such a spectral filter is created for each "view", thus, for each hybrid Laplacian. For graph signal $\mathbf{X} \in \mathbb{R}^{n \times d}$ the projected features $\mathbf{X}_v \in \mathbb{R}^{n \times K \cdot d}$, where each $\mathbf{X}_v$ represents a different "view" signal, are calculated as:

$$\mathbf{X}_v = g_\theta(\tilde{\mathbf{L}}_h)\mathbf{X} = [\tilde{\mathbf{X}}_0, \tilde{\mathbf{X}}_1, .., \tilde{\mathbf{X}}_{K-1}]\boldsymbol{\theta}_v, \tag{12}$$

with $\boldsymbol{\theta}_v = [\theta_0, \theta_1, .., \theta_{K-1}]$ are the learnable coefficients shared across vertices in the same "view" and $\tilde{\mathbf{X}}_p = \mathbf{T}_p(\tilde{\mathbf{L}}_h)\mathbf{X} = 2\tilde{\mathbf{L}}_h\tilde{\mathbf{X}}_{p-1} - \tilde{\mathbf{X}}_{p-2}$. The first two recurrent terms of the polynominal expansion are calculated as: $\tilde{\mathbf{X}}_0 = \mathbf{X}$ and $\tilde{\mathbf{X}}_1 = \tilde{\mathbf{L}}_h\mathbf{X}$. Thus, the graph signal $\mathbf{X}$ is projected onto the Chebyshev basis $\mathbf{T}_p(\tilde{\mathbf{L}}_h)$ and concatenated for all orders $p \in [0, K-1]$, similar to [22].

As a result, multiple complex relationships between neighboring vertices from different "views" are gradually captured. The receptive field is controlled by K and the trainable parameters $\boldsymbol{\theta}_v$ adjust the contribution of each Chebyshev basis. The majority of spectral convolution methods are based on projected graph features into a single graph structure, including but not limited to Chebyshev approximation. The usage of several trainable hybrid Laplacians, enables spectral filters to capture information in different projected domains.

For an input graph G, N graph signals $\mathbf{Z}_s = [\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_N]$ are stacked, as well as N hybrid Laplacians $\mathbf{L}_s = [\mathbf{L}_1, \mathbf{L}_2, .., \mathbf{L}_N]$. Then, batch normalization is applied to $\mathbf{Z}_s$, followed by a view-wise max aggregation step (VPOOL). This way, the VPOOL layer becomes invariant to random initialization of $\mathbf{Q}_v$ thanks to the batch normalization and also fuses information captured from different "views" due to the aggregation step. The aggregated view-pooled signal has a shared feature matrix, which is then passed to a linear layer followed by a non-linear activation function σ and a Dropout layer (DROP) [40]. It has to be noted that applying batch normalization to $\mathbf{Z}_s$ the graph signals from the different views have zero-mean and an activation function in the aggregated output can be meaningfully applied. Less literally, all the above procedure can be summarized in the following equation:

$$\mathbf{Y} = DROP(\sigma(VPOOL(\mathbf{Z}_s)\mathbf{W} + \mathbf{b})), \tag{13}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times m}$ is the graph output signal, with m being the number of the output features. The linear layer parameters are $\mathbf{W} \in \mathbb{R}^{K \cdot d \times m}$, $\mathbf{b} \in \mathbb{R}^m$. Based on the above, the proposed method can handle a varying number of input vertices, given that the features space is of constant dimension between different graphs (i.e., d is constant between different graphs of the same dataset). In other words, in eq.13 the model parameters are only dependent to the Chebyshev degree K and the feature vector; thus independent of the vertices of the graph. In addition, the linear layer learns to process aggregated compact signals from multiple views which results in better generalization properties. Furthermore, based on the indices of VPOOL, the corresponding most frequent hybrid Laplacian is passed to the next layer, which we call "dominant" hybrid Laplacian, defined as $\mathbf{L}_\delta$. Max and mean view pooling have also been tested on $\mathbf{L}_s$, giving slightly inferior results.

Finally, the overall multi-view GNN architecture consists of three MV-GC layers, each one followed by a VPOOL and a linear layer as depicted in eq. 13. In the first MV-GC layer the input Laplacian is calculated from the adjacency matrix as in eq. 1, which we call intrinsic Laplacian. In the rest layers, the

"dominant" Laplacian $\mathbf{L}_\delta$ of the previous VPOOL layer is the input Laplacian of the next. The choice of m provides the flexibility to adjust the number of trainable parameters and model complexity. For $m > K \cdot d$ the linear layer behaves as an encoding layer (usually the first layer of our architecture), while for $m < K \cdot d$ acts as a decoding layer. In the last layer, mean and max operators are applied in the graph output signal across the vertex dimension, similar to [6] and [48], which are further concatenated. This step further aggregates the learned features, while concurrently reduces complexity. The concatenated output features are then fed to a 2 fully connected layers. For the prediction, a softmax function is used to produce the final output $\bar{\mathbf{y}}_{pred}$. The whole network architecture is trained end-to-end. As loss function the cross entropy is used with q classes, given that $\mathbf{y}_T$ is the target one-hot vector, defined as:

$$l_s = -\sum_{1}^{q} \mathbf{y}_T \log(\bar{\mathbf{y}}_{pred}). \tag{14}$$

3.4 Computational complexity analysis

The proposed method produces dense graph representations. The proposed MV-GC layer requires the calculation of all the differences between vertex features, which is of $O(n^2)$. As shown in eq. 7, exploiting distance matrix symmetry, we only used the feature differences between unique pairs of vertices. As a consequence, the time complexity of dense non-square matrix multiplications was reduced from $O(n^2d^2)$ to $0.5O(n^2d^2)$ and Hadamard element-wise multiplication from $O(n^2d)$ to $0.5O(n^2d)$ per view. Although time complexity remains quadratic, we were able to scale it down by a factor of 4. With the current implementation, the time complexity of the MV-GC layer is linearly dependent to the number of views. Nonetheless, the required operations for each view are completely independent and can be implemented in a parallel way. In addition, space complexity required for eq. 7 is reduced from $O(n^4)$ to $0.5O(n^2d)$ per view, using eq. 3. As mentioned in [29], the required learning space complexity is $O(d^2)$ per view and $O(K \cdot d \cdot m)$ per linear layer, which are independent of the number of vertices. Still, it remains a challenging task to scale our layers for big graph data analysis and further performance optimization is left for future work.

4 Experimental evaluation

4.1 Dataset description and modifications

The proposed model (MV-AGC) is evaluated on four biological datasets and four social network datasets, available at [20]. Each dataset has an arbitrary undirected set of graphs with number of n vertices and binary edges, as well as a categorical graph label T. A summary description can be found on Table 1. In MUTAG [9], edges correspond to atom bonds and vertices to atom properties. PTC-MR [43] is a dataset of 344 molecules graphs, where classes indicate carcinogenicity

Table 1. Graph classification datasets

Datasets	Graphs	Classes	Mean Number of Vertices	Mean Number of Edges	Vertex Labels	Vertex Attributes Dim
MUTAG	188	2	17.93	19.79	Yes	-
PTC-MR	344	2	14.29	14.69	Yes	-
SYNTHIE	400	4	95.00	172.93	No	15
ENZYMES	600	6	32.63	62.14	Yes	18
PROTEINS	1113	2	39.06	72.82	Yes	29
IMDB-B	1000	2	19.77	96.53	No	-
IMDB-M	1500	3	13.00	65.94	No	-
COLLAB	5000	3	74.49	2457.78	No	-

in rats. In ENZYMES [37], each enzyme is a member of one of the Enzyme Commission top level enzyme classes. PROTEINS [2] consists of 1113 graphs modeled with vertices being their secondary structure elements and edges between two vertices exists if they are neighbors in the $3D$ space. COLLAB [51] is a scientific collaboration dataset with $5K$ graphs of different researchers from three fields of Physics. IMDB-B and IMDB-M [50] consist of ego-networks of actors that have appeared together in a movie and the goal is to find the movies genre. SYNTHIE [31] is a synthetic dataset which consists of 400 graphs with four classes and each vertex has 15 continuous-valued attributes. All datasets are publicly available[1][20].

Some biological datasets had both discrete vertex labels and continuous-valued vertex attributes. For a fair comparison between other approaches we used the discrete vertex labels in Table 2. However, classification accuracy can be further improved using the continuous-valued attributes, as shown in Table 4. The proposed model is also tested in the aforementioned social network datasets, where vertex information is not provided, using the one hot encoding of the degree of each vertex as feature vector up to a certain degree, in our case from 30 to 50, as in [52]. Using the one hot encoding of the discrete vertex labels usually falls into the case of all the vertices in the graph to have the same label, which renders the computation of $\mathbf{L}_v$ to be infeasible. We included the mentioned data to train the model apart from the "view" transformation matrices $\mathbf{M}_v$. An experimental evaluation was conducted, following the conventional approach of 10 fold cross-validation, similar to [22], [50], [52] and mean classification accuracy across folds with standard deviation are reported in Tables 2-5. The continuous-valued attributes were preprocessed with mean/std normalization before plugged in the network. The datasets are presented in ascending order with respect to their containing number of graphs.

[1] https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

Table 2. Mean accuracies with standard deviations using discrete vertex labels - biological datasets. The reported methods are sorted in chronological order.

	Datasets			
Method	MUTAG	PTC-MR	ENZYMES	PROTEINS
WL [38]	86.0 ± 1.7	61.3 ± 1.4	59.05 ± 1.05	75.6 ± 0.4
WL-OA [23]	84.5 ± 1.7	63.6 ± 1.5	59.9 ± 1.1	76.4 ± 0.4
DGK [50]	87.44 ± 2.72	60.08 ± 2.55	53.43 ± 0.91	75.68 ± 0.54
PSCN [34]	92.63 ± 4.21	62.29 ± 5.68	-	75.89 ± 2.76
KGCNN [35]	-	62.94 ± 1.69	46.35 ± 0.23	75.76 ± 0.28
DGCNN [53]	85.83 ± 1.66	58.59 ± 2.47	-	76.26 ± 0.24
DIFFPOOL [52]	-	-	62.53	76.25
GIN [47]	90.0 ± 8.8	66.6 ± 6.9	-	76.2 ± 2.6
GCAPS-CNN[45]	-	66.01 ± 5.91	61.83 ± 5.39	76.40 ± 4.17
MGCN [22]	89.1 ± 1.4	-	61.7 ± 1.3	76.5 ± 0.4
Graph U-Nets [14]	-	-	-	77.68
SAGPool [27]	-	-	-	71.86 ± 0.97
CLIQUEPOOL [30]	-	-	60.71	72.59
WKPI [55]	88.3 ± 2.6	-	-	78.5 ± 0.4
MV-AGC (Ours)	**92.98 ± 5.12**	**74.45 ± 3.42**	**64.57 ± 5.27**	**78.81 ± 3.31**

Table 3. Mean accuracies with standard deviations using vertex degrees - social network datasets

	Datasets		
Method	IMDB-B	IMDB-M	COLLAB
DGK [50]	66.96 ± 0.56	44.55 ± 0.52	73.09 ± 0.25
KGCNN [35]	71.45 ± 0.15	47.46 ± 0.21	74.93 ± 0.14
DGCNN [53]	70.03 ± 0.86	47.83 ± 0.85	73.76 ± 0.5
PSCN [34]	71.00 ± 2.29	45.23 ± 2.84	72.60 ± 2.15
DIFFPOOL [52]	-	-	82.13
GIN [47]	75.1 ± 5.1	52.3 ± 2.8	80.6 ± 1.9
Graph U-Nets [14]	-	-	77.56
CLIQUEPOOL [30]	-	-	74.50
GCAPS-CNN[45]	71.69 ± 3.40	48.50 ± 4.10	77.71 ± 2.51
CAPS-GNN [46]	73.10 ± 4.83	50.27 ± 2.65	79.62 ± 0.91
HO-GNN [32]	74.2	49.5	-
WKPI [55]	75.1 ± 1.1	49.05 ± 0.4	-
MV-AGC (Ours)	**78.20 ± 3.05**	**53.47 ± 3.62**	**82.41 ± 1.09**

4.2 Implementation details

The described multi-view GNN architecture is implemented in the PyTorch [36] framework. Training is achieved using the stochastic gradient descent optimizer [3] with a constant learning rate ranging from $4 \cdot e^{-4}$ to $8 \cdot e^{-3}$ and single batch size. As non-linear activation function ReLU is used. Dropout was used to avoid overfitting, mostly on the small data sets, similar to [34]. Chebyshev degree K was set to 6 for all conducted experiments. The hidden fully connected layer has 128 units. $\mathbf{Q}_v$ is initialized with a random uniform distribution in range $(0, 1)$. $\mathbf{M}_v$ is regularized in each iteration divided with it's maximum element, although preserving it's desired properties. The number of views per MV-GC layer is set to 8 for the first layer and 6 for the others. The epochs per dataset vary from 30, in COLLAB, to 80, in the ENZYMES dataset. The number of output features m of each linear layer was set to 80, 128 and 256 for the bioinformatics datasets. For the COLLAB dataset, a smaller number of views per MV-GC layer was used, due to high computational demands. For all datasets the experiments were conducted in a NVIDIA GeForce GTX-1080 GPU with 12GB of memory and 16 GB of RAM.

An incremental training strategy was adopted to choose the model hyper-parameters e.g. number of views. First, we used a single MV-GC layer and we observed that multiple views helped significantly in graph classification. As a sanity check, we inspected the gradients of Q_v that roughly started from a mean value of 10^{-3} and reached a mean value of 10^{-8} at the end of the training. Then, due to complexity we could not fit more than three layers, which provided the best results. We made hyperparameter tuning and trained the network on two datasets and used the same model and hyperparameters in the other datasets.

Table 4. Mean accuracies with standard deviations using only the continuous-valued vertex attributes

	Datasets		
Method	SYNTHIE	ENZYMES	PROTEINS
HGK [31]	86.27 ± 0.72	66.73 ± 0.91	75.14 ± 0.47
GraphHopper [11]	-	69.60 ± 1.30	-
SP [1]	-	71.30 ± 1.30	75.50 ± 0.80
FGW [44]	-	71.00 ± 6.76	74.55 ± 2.74
PROPAK [33]	-	71.67 ± 5.63	61.34 ± 4.38
MV-AGC (Ours)	**90.02 ± 3.87**	**72.62 ± 2.63**	**77.66 ± 2.51**

4.3 Experimental results

We evaluate our model compared to other graph deep learning approaches, as well as classical graph kernel methods. As presented in Table 2, superior results

in 4 biological datasets are reported, using vertex labels as one hot encodings. A significant gain of 7.85 was reached in PTC-MR dataset, which is justified in the fact that the provided vertex labels of the dataset are of critical importance for classification. In social network datasets, as shown in Table 3, using vertices degrees as input features, we report state-of-the-art results in 3 social network datasets. Our method surpasses all previous approaches in the datasets that contain continuous-valued feature attributes as depicted in Table 4. As show in Table 5, where we vary the number of "views" per MV-GC layer without any hyperparameter tuning, there is a significant increase in generalization as the number of views per layer increases up to a certain value. Having only one trainable transformation matrix per layer would reduce the model close to [29], which is usually unstable to train and does not produce generalized representations. There is also a trade-off in the choice of views between training time and accuracy. Nevertheless, Table 5 clearly proves that the idea of multi-view metric learning for graph classification is non-trivial.

Table 5. Model (MV-AGC) comparison with varying number of views per layer

	Datasets		
# Views	MUTAG	ENZYMES	PROTEINS
2	87.71 ± 6.3	56.02 ± 6.3	75.36 ± 3.9
3	88.88 ± 5.4	57.61 ± 3.7	**78.08 ± 3.4**
6	90.43 ± 5.1	**62.27 ± 4.3**	77.83 ± 3.7
9	**90.52 ± 5.4**	61.10 ± 4.6	77.81 ± 3.4

4.4 Discussion

Some limitations that have to be taken account that influence the model's standard deviation and accuracy, based on our experimental study, are the following: a) the number of samples per dataset, b) the variability in the number of vertices, c) how informative is the graph signal for the prediction, and d) the choice of α in Eq. 9. It is observed that it is more difficult to estimate optimal global affine transformations as the number of samples increase and as the graphs have a wider range of vertices. On the other hand, in PTC-MR we observed a huge gain, because the graph signal is a significant feature for classification and the vertex variability is low. The choice of α to be equal to 1 introduces some extra standard deviation, but it still yields better accuracy for the majority of datasets. The reason behind this choice is to not scale down the gradient values on the affine transformation matrices. Moreover, large-scale graphs (with more than 3000 vertices) cannot be process in the referenced hardware. In a future work, we aim to encounter this limitation.

We claim that MV-AGC best fits continuous-valued datasets, due to the multi-view representation of the feature space. This is demonstrated in Table

4 by the higher gains in the continuous-valued datasets. As a future work, the convergence of the network with respect to the views will be further investigated.

5 Conclusions

In the present work, a graph analog of multi-view operations in CNNs was developed. We propose a novel multi-view GNN architecture able to exploit vertices information in an adaptive manner. The proposed MV-GC layer generates "views" representing multiple graph structures, based on a trainable non-Euclidean distance metric learning process. We explore pairwise feature relationships inside a graph via the newly introduced hybrid Laplacian. Spectral filtering is applied to the input graph signal with a different hybrid Laplacian per "view", producing multiple projected signals and, thus, encapsulating different relations between the data. A new view pooling layer is also introduced, able to fuse information from different views. The proposed multi-view graph distance metric learning methodology can also be applied in other graph convolutional schemes, which is out of the scope of this paper and left as a future research direction. Our model (MV-AGC) provides state-of-the-art results in 4 bioinformatics datasets with discrete vertices labels, as well as in 3 social network datasets. Finally, the proposed method outperforms previous approaches in all datasets with continuous-valued vertex attributes.

Acknowledgement The work presented in this paper was supported by the European Commission under contract H2020-822601 NADINE.

References

1. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining (ICDM'05). pp. 8–pp. IEEE (2005)
2. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21**(suppl_1), i47–i56 (2005)
3. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)
4. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
5. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
6. Cangea, C., Veličković, P., Jovanović, N., Kipf, T., Liò, P.: Towards sparse hierarchical graph classifiers. arXiv preprint arXiv:1811.01287 (2018)
7. Chung, F.R., Graham, F.C.: Spectral graph theory. No. 92, American Mathematical Soc. (1997)
8. De Maesschalck, R., Jouan-Rimbaud, D., Massart, D.L.: The mahalanobis distance. Chemometrics and intelligent laboratory systems **50**(1), 1–18 (2000)

9. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry **34**(2), 786–797 (1991)
10. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)
11. Feragen, A., Kasenburg, N., Petersen, J., de Bruijne, M., Borgwardt, K.: Scalable kernels for graphs with continuous attributes. In: Advances in Neural Information Processing Systems. pp. 216–224 (2013)
12. Fey, M., Eric Lenssen, J., Weichert, F., Müller, H.: Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 869–877 (2018)
13. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics **36**(4), 193–202 (1980)
14. Gao, H., Ji, S.: Graph u-net (2019), `https://openreview.net/forum?id=HJePRoAct7`
15. Gomez, L.G., Chiem, B., Delvenne, J.C.: Dynamics based features for graph classification. arXiv preprint arXiv:1705.10817 (2017)
16. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163 (2015)
17. Hoi, S.C., Liu, W., Chang, S.F.: Semi-supervised distance metric learning for collaborative image retrieval and clustering. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **6**(3), 18 (2010)
18. Horadam, K.J.: Hadamard matrices and their applications. Princeton university press (2012)
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
20. Kersting, K., Kriege, N.M., Morris, C., Mutzel, P., Neumann, M.: Benchmark data sets for graph kernels (2016), `http://graphkernels.cs.tu-dortmund.de`
21. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
22. Knyazev, B., Lin, X., Amer, M.R., Taylor, G.W.: Spectral multigraph networks for discovering and fusing relationships in molecules. arXiv preprint arXiv:1811.09595 (2018)
23. Kriege, N.M., Giscard, P.L., Wilson, R.: On valid optimal assignment kernels and applications to graph classification. In: Advances in Neural Information Processing Systems. pp. 1623–1631 (2016)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
25. Ktena, S.I., Parisot, S., Ferrante, E., Rajchl, M., Lee, M., Glocker, B., Rueckert, D.: Distance metric learning using graph convolutional networks: Application to functional brain networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 469–477. Springer (2017)
26. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision, pp. 319–345. Springer (1999)
27. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. arXiv preprint arXiv:1904.08082 (2019)

28. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
29. Li, R., Wang, S., Zhu, F., Huang, J.: Adaptive graph convolutional neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
30. Luzhnica, E., Day, B., Lio, P.: Clique pooling for graph classification. arXiv preprint arXiv:1904.00374 (2019)
31. Morris, C., Kriege, N.M., Kersting, K., Mutzel, P.: Faster kernels for graphs with continuous attributes via hashing. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 1095–1100. IEEE (2016)
32. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4602–4609 (2019)
33. Neumann, M., Garnett, R., Bauckhage, C., Kersting, K.: Propagation kernels: efficient graph kernels from propagated information. Machine Learning **102**(2), 209–245 (2016)
34. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International conference on machine learning. pp. 2014–2023 (2016)
35. Nikolentzos, G., Meladianos, P., Jean-Pierre Tixier, A., Skianis, K., Vazirgiannis, M.: Kernel Graph Convolutional Neural Networks. arXiv e-prints arXiv:1710.10689 (Oct 2017)
36. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
37. Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., Schomburg, D.: Brenda, the enzyme database: updates and major new developments. Nucleic acids research **32**(suppl_1), D431–D433 (2004)
38. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.v., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(Sep), 2539–2561 (2011)
39. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. arXiv preprint arXiv:1211.0053 (2012)
40. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
41. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: Proc. ICCV (2015)
42. Takerkart, S., Auzias, G., Thirion, B., Schön, D., Ralaivola, L.: Graph-based inter-subject classification of local fmri patterns. In: International Workshop on Machine Learning in Medical Imaging. pp. 184–192. Springer (2012)
43. Toivonen, H., Srinivasan, A., King, R.D., Kramer, S., Helma, C.: Statistical evaluation of the predictive toxicology challenge 2000–2001. Bioinformatics **19**(10), 1183–1193 (2003)
44. Vayer, T., Chapel, L., Flamary, R., Tavenard, R., Courty, N.: Optimal transport for structured data with application on graphs. arXiv preprint arXiv:1805.09114 (2018)
45. Verma, S., Zhang, Z.L.: Graph capsule convolutional neural networks. arXiv preprint arXiv:1805.08090 (2018)

46. Xinyi, Z., Chen, L.: Capsule graph neural network. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Byl8BnRcYm
47. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
48. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. arXiv preprint arXiv:1806.03536 (2018)
49. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
50. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374. ACM (2015)
51. Yanardag, P., Vishwanathan, S.: A structural smoothing framework for robust graph comparison. In: Advances in neural information processing systems. pp. 2134–2142 (2015)
52. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems. pp. 4805–4815 (2018)
53. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
54. Zhang, X., He, L., Chen, K., Luo, Y., Zhou, J., Wang, F.: Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease. In: AMIA Annual Symposium Proceedings. vol. 2018, p. 1147. American Medical Informatics Association (2018)
55. Zhao, Q., Wang, Y.: Learning metrics for persistence-based summaries and applications for graph classification. In: Advances in Neural Information Processing Systems. pp. 9855–9866 (2019)
56. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Sun, M.: Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434 (2018)