Pointspectrum: Equivariance Meets Laplacian Filtering for Graph Representation Learning

Marinos Poiitis
Aristotle University of Thessaloniki
mpoiitis@csd.auth.gr

Pavlos Sermpezis
Aristotle University of Thessaloniki
sermpezis@csd.auth.gr

Athena Vakali
Aristotle University of Thessaloniki
avakali@csd.auth.gr

***Abstract*—Graph Representation Learning (GRL) has revolutionized learning on graphs, by combining the graph's structural information with node/edge attributes. While Graph Neural Networks (GNNs) have been used in state-of-the-art GRL architectures, they suffer from over smoothing when many GNN layers need to be stacked. In a different approach, *spectral methods* based on graph filtering have emerged addressing over smoothing; however, up to now, they employ traditional neural networks that cannot efficiently exploit the structure of graph data. Motivated by this, we propose PointSpectrum, a spectral method that incorporates a set equivariant network to account for graph structure. PointSpectrum enhances the efficiency and expressiveness of spectral methods, while it outperforms or competes with state-of-the-art GRL methods. Overall, PointSpectrum addresses over smoothing through graph filtering and captures a graph's structure through set equivariance. Our findings are promising for this architectural shift for (spectral) GRL methods.**

I. Introduction

Graphs are extensively used to describe real-world data in various domains, e.g., citation and social networks [8], recommender systems [28], or adversarial attacks [29]. Due to the complex structure of graphs, traditional machine learning models are insufficient for graph-based tasks such as node and graph classification, link prediction and node clustering. This necessity has given rise to *Graph Representation Learning (GRL)*, which aims to capture the structure of input graph data and produce meaningful low-dimensional representations.

A main track of GRL methods is based on Graph Neural Networks (GNNs). GNNs and specifically Graph Convolutional Networks (GCN) [8] have advanced the research on GRL leading to outstanding performance as they capture a node's neighborhood influence. Nevertheless, each GNN layer considers only the local one-hop node relations leading to unavoidably deep layer stacking to efficiently account for the global graph information. Notwithstanding, vanilla GNNs cannot be arbitrarily deep as they lead to over smoothing and information loss [24], [31], [2]. To address over smoothing, spectral methods exploiting graph filters have been introduced [20], [30]. However, so far, these methods have been used in conjunction with traditional neural networks (e.g., MLPs or CNNs) that cannot efficiently exploit the *set equivariance* property of graph data (i.e., equivariance to permutations in the input data). In the graph domain there is no implicit data ordering, and thus the existing spectral methods waste an important portion of their computational capacity.

Contribution. In this work, we propose *PointSpectrum*, a GRL architecture that bridges the gap between GNN-based and spectral GRL methods (Section II). PointSpectrum is based on Laplacian smoothing (used in spectral methods to alleviate the problem of over-smoothing), while it maintains the set equivariance property of GNN-based approaches. Specifically, PointSpectrum is an unsupervised end-to-end trainable architecture, with the following main components:

- *Input*: a low-pass graph filter (Laplacian smoothing) is applied on the input graph data that enables the computation of k-order graph convolution for arbitrarily large k without over smoothing node features
- *Encoder*: the input data are fed to a set equivariant network that generates low-dimensional node embeddings
- *Decoder*: a joint loss is employed to account for the reconstruction of the input data and their better separation in the embedding space through a clustering metric.

To the best of our knowledge this is the first work that introduces set equivariance in spectral methods. Our experiments (Section III) show that: (i) set equivariant networks can increase the robustness (wrt. the model parameters) and efficiency (e.g., faster convergence, expressiveness) of spectral methods; (ii) PointSpectrum outperforms or competes with the state-of-the-art in all benchmark tasks and datasets. Overall, we showcase a new direction for GRL: the combination of spectral methods with set equivariance. Incorporating set equivariant networks in existing spectral methods can be straightforward (e.g., replacing MLPs or CNNs), and our results are promising for the efficiency of this approach.

II. Methodology

Definitions. We consider a non-directed attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the node set with $|\mathcal{V}| = n$, $\mathcal{E}$ is the edge set, and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times l}$ is the feature matrix consisting of feature vectors $\mathbf{x}_i \in \mathbb{R}^l$, $\forall v_i \in \mathcal{V}$. The structural information of graph $\mathcal{G}$ is represented by the adjacency matrix $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{n \times n}_{\geq 0}$.

Goals. The goal of GRL is to map nodes to low-dimensional embeddings, which should preserve both the structural ($\mathbf{A}$) and contextual ($\mathbf{X}$) information of $\mathcal{G}$. We denote as $\mathbf{Z} \in \mathbb{R}^{n \times l'}$, with $l' \ll l$, the matrix with the node embeddings.

Approach overview. We propose a methodology to compute an embedding matrix $\mathbf{Z}$, which combines structural and contextual information in a twofold way: on one hand, it uses

Laplacian smoothing (based on $\mathbf{A}$) of the feature matrix $\mathbf{X}$ (Section II-A) and, on the other hand, by considering the node features as a set of points it exploits global information using a permutation equivariant network architecture (Section II-B). Last, we leverage node clustering as a boosting component that further enhances performance by better separating nodes in the embedding space. The entire architecture that brings these components together is presented in Section II-C.

A. Graph Convolution and Laplacian Smoothing

The most important notion in the prevalent GNN-based embedding methods, such as GCN [8], is that neighboring nodes should be similar and hence their features should be smoother than non-neighboring nodes in the graph manifold. However, these methods capture deeper connections by stacking multiple layers, leading to deep architectures, which are known to overly smooth the node features [2]. Over smoothing occurs as each layer repeatedly smooths the original features so as to account for the deeper interactions. To address this problem, the domain of graph signal processing has been used; namely, graph convolution by Laplacian Smoothing [30], [3].

Specifically, "spectral methods" in GRL use a smoothed feature matrix $\bar{\mathbf{X}}$, instead of the original $\mathbf{X}$, which corresponds to a k-th order graph convolution of $\mathbf{X}$:

$$\bar{\mathbf{X}} = \mathbf{H}^{\mathbf{k}}\mathbf{X} \tag{1}$$

where matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ is the Laplacian Smoothing filter (discussed below). The multiplication of feature matrix $\mathbf{X}$ with filter $\mathbf{H}$ corresponds to a 1-order graph convolution (or 1-order graph smoothing). Stacking k filters together, i.e., k multiplications with $\mathbf{H}$, leads to a k-order convolution as in Eq. 1. Thus, deep network interactions are captured by the power of filter $\mathbf{H}$ instead of repeated convolutions of the input features and therefore over smoothing is avoided.

The Laplacian Smoothing Filter, H: The intuition behind k-order convolution of spectral methods is the following: Each column of feature matrix $\mathbf{X}(:, \mathbf{i}) \in \mathbb{R}^n$ (i.e., the vector with the values of all nodes for a single feature) can be considered as a graph signal. The smoothness of a signal depicts the similarity between the graph nodes. Since neighboring nodes should be similar, to capture this similarity, we would need to construct a smooth signal based on $\mathbf{X}(:, \mathbf{i})$ that captures node adjacency and takes into account the features of the most important neighboring nodes. It can be shown that the Laplacian Smoothing filter $\mathbf{H}$ as defined in Eq. 2 [18] can cancel high frequencies between neighboring nodes:

$$\mathbf{H} = \mathbf{I} - \mu\mathbf{L} \tag{2}$$

where $\mu \in \mathbb{R}$, $\mathbf{I}$ is the identity matrix and $\mathbf{L}$ is the graph Laplacian. The graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = diag(d_1, d_2, \ldots, d_n)$ is the degree matrix of $\mathbf{A}$, with $d_i = \sum_{v_j \in \mathcal{V}} a_{ij}$ being the degree of node v_i. In order for $\mathbf{H}$ to be low-pass, $1 - \mu\lambda$ should be a non-negative degressive function. [3] showed that the optimal value of μ is $1/\lambda_{max}$, with λ_{max} denoting the largest eigenvalue of $\mathbf{L}$; in the remainder, we use this value for μ.

Renormalization trick: In practice the renormalization trick [8], i.e., adding self-loops, has been shown to improve accuracy and shrink the graph spectral domain [23]. Qualitatively, this means that for every node the smoothing filter also considers its own features alongside the ones from its neighbors. Thus, we perform the following transformation: self-loops are added to the adjacency: $\bar{\mathbf{A}} = \mathbf{I} + \mathbf{A}$; we then use the symmetric normalized graph Laplacian $\bar{\mathbf{L}}_{\mathbf{sym}} = \bar{\mathbf{D}}^{-\frac{1}{2}}\bar{\mathbf{L}}\bar{\mathbf{D}}^{-\frac{1}{2}}$, where $\bar{\mathbf{D}}$ and $\bar{\mathbf{L}}$ are the degree and Laplacian matrices of $\bar{\mathbf{A}}$. Finally, the resulting Laplacian smoothing filter becomes $\mathbf{H} = \mathbf{I} - k\bar{\mathbf{L}}_{\mathbf{sym}}$.

B. Equivariance and PointNetST

Neural networks can approximate any continuous function f given sufficient capacity and expressive power [16], [26]. We denote a neural network as a function $f : \mathbf{X} \mapsto Y$ operating on the feature matrix $\mathbf{X}$ (or $\bar{\mathbf{X}}$).

Sets and permutation equivariance. Conventional neural networks such as Multilayer Perceptrons (MLPs) or Convolutional Neural Networks (CNNs) act on data where there is an implicit order (e.g., adjacent pixels in images). However, graphs do not have any implicit order: nodes can be presented in different order but the graph still maintains the same structure (isomorphism). Therefore, $\mathbf{X}$ can be seen as a set of points $\mathbf{x} \in \mathbb{R}^l$, and f as a function on a set.

Definition 1: A function $f : \mathbb{R}^{n \times l} \mapsto \mathbb{R}^{n \times l'}$ acting on a set $\mathbf{X} \in \mathbb{R}^{n \times l}$ is *permutation equivariant* when $f(\sigma \cdot \mathbf{X}) = \sigma \cdot f(\mathbf{X})$ for any permutation σ of the set $\mathbf{X}$.

In other words, a *permutation equivariant* function f produces the same output (e.g., embedding) for each set item (e.g., node) irrespective of the order of the given data. This means that if a transformation (e.g., a permutation of the input matrix's rows) is applied to the input data, the same transformation should be applied to the model's output as well.

Equivariance in GRL methods. Set equivariance is not captured by traditional neural networks such as MLPs (or CNNs), which are mainly used in the spectral methods [30], [3]. More specifically, [27] proved that a function f is permutation equivariant iff it can be decomposed in the form $\rho(\sum_{\mathbf{x} \in \mathbf{X}} \phi(\mathbf{x}))$, where ρ and ϕ are suitable transformations. Also, each MLP or CNN layer can be seen as a transformation ϕ, and a deep network with m such layers can be expressed as $f(\mathbf{X}) = \phi_m \circ \cdots \circ \phi_1(\mathbf{X})$. Hence, the necessary decomposition for permutation equivariance does not hold. On the contrary, GNNs can capture not only permutation equivariance but also graph connectivity; however, at a cost of oversmoothing. *This work aims to bring benefits from both GRL approaches: we employ a set equivariant network that accounts for the graph structure (through the matrix $\bar{\mathbf{X}}$) and avoids oversmoothing at the same time (see Section II-A).* This property is crucial for our model's prevalence over the traditional neural networks, as it will be shown in Section III.

PointNetST. In this work, PointNetST [17] is employed as the permutation equivariant neural network architecture. While different choices can be made, such as a deep self-attention network [21] or the constructions of [7] and [16], PointNetST

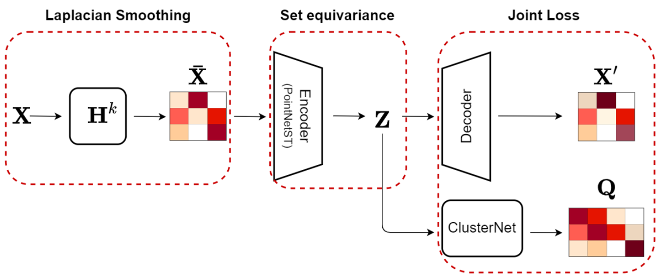


Fig. 1: PointSpectrum architecture. Encoder is a pointNetST network producing node embeddings in an equivariant manner, the pair-wise decoder reconstructs input $\bar{\mathbf{X}}$ and last ClusterNet is employed to better separate points in the embedding space.

is preferred as it is provably a universal approximator over the space of equivariant functions [17] and can be implemented as an arbitrarily deep neural network with the following form:

$$f(\mathbf{X}) = \phi_m \circ \sigma \cdots \circ \sigma \circ \phi_1(\mathbf{X}) \tag{3}$$

where σ is a non-linearity such as ReLU and ϕ_i, $i = 1, ..., m$, is the DeepSet layer [27]:

$$\phi_i(\mathbf{X}) = \mathbf{X}\mathbf{W}_1 + \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X}\mathbf{W}_2 + \mathbf{1}\mathbf{w} \tag{4}$$

with $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{l_i \times l_{i+1}}$ and $\mathbf{w} \in \mathbb{R}^{l_{i+1}}$ being the layer's parameters. *Remark:* while Eq. 4 is a generic form of an equivariant transformation, it is noted that PointNetST contains only a single layer with non-zero $\mathbf{W}_2$.

C. *PointSpectrum*

This work proposes PointSpectrum (Figure 1), an encoder-decoder architecture for GRL that consists of three main components: (i) Laplacian smoothing of the feature matrix, (ii) PointNetST as a permutation equivariant neural network of the encoder, (iii) a clustering module alongside the decoder.

Input. The input $\bar{\mathbf{X}}$ of the encoder-decoder network is the k-order graph convolution of node feature matrix $\mathbf{X}$, which is computed using a Laplacian filter $\mathbf{H}$ as described in Section II-A. The larger the convolution order k, the deeper node-wise interactions the model can capture.

Encoder. The encoder is the permutation equivariant PointNetST, which generates the node embeddings $\mathbf{Z}$, as described in Section II-B. The embeddings are fed to two individual modules: the decoder and the ClusterNet.

Decoder: The decoder aims to reconstruct a pairwise similarity value between the computed node embeddings based on $\bar{\mathbf{X}}$. Different choices for the reconstruction loss function can be made (e.g. minimum squared error [20] or noise-contrastive binary cross entropy [19]). However, as connectivity is incorporated in the smoothed signal, we use a pairwise decoder and cross entropy loss function with negative sampling (non-existing edges):

$$L_r = -\sum_{i \in \mathcal{V}} \left[\sum_{(i,j) \in \mathcal{E}} \log(\mathbf{z}_i \mathbf{z}_j) + \sum_{k \in \mathcal{N}_i} \log(1 - \mathbf{z}_i \mathbf{z}_k) \right] \tag{5}$$

where $\mathcal{N}_i$ are the negative samples for node i.

ClusterNet is a differentiable clustering module, which learns to assign nodes to clusters to better separate them in the embedding space [22]. ClusterNet learns a distribution of soft assignments $\mathbf{Q}$, where q_{ij} expresses the probability of node i belonging to cluster j, by optimizing KL-divergence loss:

$$L_c = KL(P||Q) = \sum_i \sum_j p_{ij} \cdot \log \frac{p_{ij}}{q_{ij}} \tag{6}$$

where $\mathbf{P}$ a target distribution (updated in every epoch) to emphasize more "confident" assignments [20]:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \tag{7}$$

Having computed $\mathbf{Q}$, cluster centers can be extracted directly by averaging the soft assignments for each cluster. Overall, PointSpectrum optimizes the following joint loss of the Decoder and ClusterNet (α, β are hyper parameters):

$$L = \alpha \cdot L_r + \beta \cdot L_c \tag{8}$$

III. Experimental Results

In this section, we demonstrate the efficiency of PointSpectrum in benchmark GRL datasets and tasks. We first present the evaluation setup (Section III-A). We provide experimental evidence for the gains that are introduced by the equivariance component of PointSpectrum over the traditional deep learning architectures in terms of efficiency (and robustness (Sections III-B and III-C). Finally, we compare PointSpectrum against baseline and state-of-the art GRL methods (Section III-D) and provide a qualitative visual analysis (Section III-E).

A. *Datasets & experimental setup*

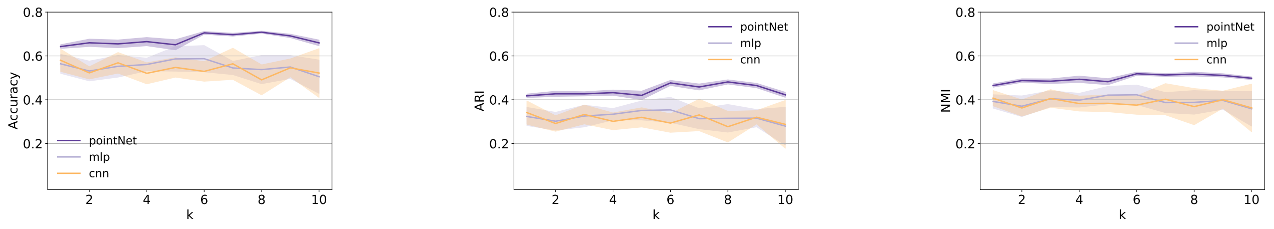
Datasets. We evaluated the performance of PointSpectrum on three widely used benchmark citation network datasets, namely Cora [12], Citeseer [4] and Pubmed [13] (Table I).

Dataset	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

TABLE I: Datasets detailed information

PointSpectrum setup. For all tasks, a PointSpectrum model is used for the encoder with a single PointNetST layer of dimension $l = 100$ (which is also the embedding dimension). For the clusterNet module, centers are randomly initialized and correspond to the number of distinct labels in each dataset. Last, weights are initialized according to [6] (He initialization).

Baseline methods. We compare PointSpectrum to several baseline methods (see details in Section IV), which we distinguish in four categories: (i) *feature-only* (K-Means), (ii) *traditional GRL* (DeepWalk, DNGR, TADW), (iii) *GNN-based* (VGAE, DGI, GIC), and (iv) *Spectral* (DAEGC, AGC, AGE). In Section III-D, we compare against all these methods on clustering, and only the most prevalent ones on link prediction.



(a) Accuracy (Cora) (b) ARI (Cora) (c) NMI (Cora)

Fig. 2: Accuracy, ARI and NMI metrics of the PointSpectrum with PointNetST, MLP and CNN-based networks in the encoder for the clustering task on the Cora dataset; the mean value and standard deviation of 10 experiment runs are depicted. *The set equivariant PointNetST achieves higher performance and is more robust than the MLP and CNN variants, irrespective of the convolution order.*

Results are obtained directly from [11]; in particular, for AGE, K-means is used as the clustering method (instead of spectral clustering as in [3]) to enable a fair comparison. Results in Sections III-B–III-C refer to node clustering, since it is the most natural task for the PointSpectrum architecture.

B. Efficiency of set equivariance

We investigate the efficiency of using a set equivariant network (PointNetST) along with Laplacian smoothing ($\bar{\mathbf{X}}$), by comparing the PointSpectrum architecture (Figure 1) against two variants with MLP and CNN neural networks in the encoder. Figure 2 shows the results for the different encoder types and convolution orders (k) for the clustering task.

The results for all metrics (Accuracy, ARI, NMI) in the Cora dataset suggest that *PointNetST achieves higher performance, is more robust (lower variance), and has less fluctuations wrt. the convolution order than the MLP and CNN variants.*

On the Citeseer dataset (detailed results are omitted), the superiority of PointNetST in terms of performance is even higher than on Cora. On the Pubmed dataset, all variants performs similarly: the added gains in a dataset with few features and simple graph structure may not be significant.

C. Performance on permuted data

As already shown, set equivariance offers performance and computational efficiency. Here, we focus on the main characteristic of set equivariance: its robustness on permuted inputs. To demonstrate this, we train PointSpectrum by randomly permuting the rows of $\bar{\mathbf{X}}$ (input) in every epoch, and evaluate the trained model on the original $\bar{\mathbf{X}}$. Table II presents the results of this experiment (in parentheses the initial results without permuted data in training). PointNetST achieves the highest performance, but more importantly, the drop in performance due to the corrupted input is significantly smaller compared to the MLP/CNN variants. This highlights that *PointSpectrum can capture data permutations on which it has not been explicitly trained* (e.g., graph isomorphism). On the contrary, the MLP/CNN variants perform poorly on permuted data (see, e.g., NMI/ARI metrics in Citeseer or Pubmed).

D. Comparison against baselines

In this section, we compare PointSpectrum's efficiency in clustering and link prediction tasks against baseline and state-of-the-art methods. We would like to stress that the main goal of this work is to introduce set equivariance in spectral GRL methods, and demonstrate the benefits it can bring. Hence, we do not extensively perform hyperparameter optimization. Here, we compare against baselines for completeness and for demonstrating its efficiency compared to the state-of-the-art in GRL. Nevertheless, the tested PointSpectrum implementation still outperforms spectral methods, and achieves top or near top performance in the evaluation tasks.

Clustering: The goal in clustering is to group similar nodes into m classes based on the computed embeddings. Similar to related literature, the number of classes is given, and in the evaluation the labels provided by the datasets are used. In Table III we report the best PointSpectrum results out of 10 experiment runs for 4 metrics: Accuracy (ACC), Normalized Mutual Information (NMI), Adjusted Randomized Index (ARI) and Macro-F1 score (F1).

Focusing first on the Spectral methods (bottom rows of Table III), we see that *the overall PointSpectrum performance (i.e., for the majority of metrics and datasets) is superior to Spectral methods*. When compared to all GRL methods, *PointSpectrum achieves state-of-the-art performance* on most metrics in the Cora and Pubmed datasets, and on the accuracy metric in the Citeseer dataset (in which the GNN-based models GIC and DGI perform best for the other metrics).

Link prediction: In link prediction, some graphs edges are hidden to the model during training and its goal is to predict these hidden interactions based on the computed node embeddings. For this task 10% of positive and negative edges are used as test and 20% as the validation set. Table IV presents the model performance on link prediction (mean value and standard deviation over 10 runs) as measured by the Area Under the Curve (AUC) and Average Precision (AP) metrics.

PointSpectrum outperforms all baselines by a significant margin in Cora ($\sim 5.5\%$) and Pubmed ($\sim 1.5\%$), while in Citeseer it is the second best method after GIC ($< 1\%$ margin; note also the much lower variance of PointSpectrum).

E. Qualitative Analysis

We evaluate qualitatively PointSpectrum by visualizing the computed embeddings using t-SNE [10] for Cora in Figure 3.

	Cora				Citeseer				Pubmed			
Method	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
MLP	0.275 (0.583)	0.247 (0.379)	0.170 (0.363)	0.145 (0.484)	0.308 (0.445)	0.053 (0.199)	0.043 (0.172)	0.272 (0.400)	0.433 (0.638)	0.022 (0.210)	0.002 (0.212)	0.269 (0.639)
CNN	0.301 (0.571)	0.090 (0.417)	0.514 (0.346)	0.208 (0.489)	0.362 (0.423)	0.098 (0.197)	0.091 (0.175)	0.314 (0.395)	0.457 (0.684)	0.063 (0.263)	0.018 (0.298)	0.323 (0.673)
PointNetST	**0.625** (0.715)	**0.431** (0.528)	**0.385** (0.493)	**0.538** (0.693)	**0.537** (0.703)	**0.316** (0.430)	**0.271** (0.451)	**0.463** (0.613)	**0.691** (0.710)	**0.298** (0.295)	**0.307** (0.329)	**0.687** (0.703)

TABLE II: Clustering results based on the true labels. The input is randomly permuted in every epoch during training. The reported metrics result from the best clustering assignments of the original (not permuted) input; in parentheses results for a model trained on the original input. *PointSpectrum captures data permutations on which it has not been explicitly trained, whereas the MLP/CNN variants perform poorly.*

	Cora				Citeseer				Pubmed			
Method	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
K-Means	0.492	0.321	0.230	0.368	0.540	0.305	0.279	0.409	0.398	0.001	0.002	0.195
DeepWalk	0.484	0.327	0.243	0.392	0.337	0.088	0.092	0.270	0.684	0.279	0.299	0.670
DNGR	0.419	0.318	0.142	0.340	0.326	0.180	0.044	0.300	0.458	0.155	0.054	0.467
TADW	0.560	0.441	0.332	0.481	0.455	0.291	0.228	0.414	0.511	0.001	0.001	0.335
GAE/VGAE	0.609	0.436	0.347	0.609	0.408	0.176	0.124	0.372	0.672	0.277	0.279	0.660
DGI	0.713	**0.564**	0.511	0.682	0.688	0.444	0.450	**0.657**	0.533	0.181	0.166	0.186
GIC	0.725	0.537	0.508	0.694	0.696	**0.453**	**0.465**	0.654	0.673	0.319	0.291	0.704
DAEGC	0.704	0.528	0.496	0.682	0.672	0.397	0.410	0.636	0.671	0.266	0.278	0.659
AGC	0.689	0.537	0.487	0.656	0.670	0.411	0.419	0.625	0.698	0.316	0.319	0.404
AGE	0.712	0.559	-	0.682	0.569	0.348	-	0.544	-	-	-	-
PointSpectrum (ours)	**0.736**	0.529	**0.516**	**0.711**	**0.703**	0.430	0.451	0.613	**0.776**	**0.375**	**0.444**	**0.768**

TABLE III: Clustering results based on the true labels. Horizontal lines discriminate feature-only, traditional GRL, GNN-based and Spectral methods. Underlined values indicate the best results among the spectral methods, and **bold** values the best results among all methods.

	Cora		Citeseer		Pubmed	
Method	AUC	AP	AUC	AP	AUC	AP
DeepWalk	0.846 ± 0.001	0.885 ± 0.000	0.805 ± 0.002	0.850 ± 0.001	0.842 ± 0.000	0.878 ± 0.000
GAE/VGAE	0.914 ± 0.001	0.926 ± 0.001	0.908 ± 0.002	0.920 ± 0.002	0.964 ± 0.000	0.965 ± 0.000
DGI	0.898 ± 0.08	0.897 ± 0.1	0.955 ± 0.1	0.957 ± 0.1	0.912 ± 0.06	0.922 ± 0.05
GIC	0.935 ± 0.06	0.933 ± 0.07	$\mathbf{0.970 \pm 0.05}$	$\mathbf{0.968 \pm 0.05}$	0.937 ± 0.03	0.935 ± 0.03
AGE	0.924 ± 0.000	0.932 ± 0.000	0.924 ± 0.000	0.930 ± 0.000	0.968 ± 0.000	0.971 ± 0.000
PointSpectrum (ours)	$\mathbf{0.989 \pm 0.000}$	$\mathbf{0.989 \pm 0.001}$	0.966 ± 0.000	0.961 ± 0.001	$\mathbf{0.987 \pm 0.001}$	$\mathbf{0.982 \pm 0.002}$

TABLE IV: Link prediction performance. Area Under the Curve (AUC) and Average Precision (AP) are reported. Best results on each dataset are shown in **bold**, and second best in underlined.

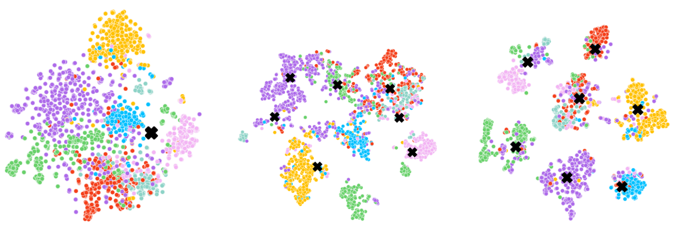

(a) Initial (Cora) (b) Intermediate (Cora) (c) Final (Cora)

Fig. 3: PointSpectrum's training behavior on the Cora dataset using t-SNE for visualization. ClusterNet's trainable centers are denoted as black 'x' marks.

On one hand, PointSpectrum separates well the nodes in the embedding space as the training proceeds. On the other hand, the ClusterNet component enables the PointSpectrum to learn the cluster centers as well (denotes as 'x' marks), as it pushes them from a random initial point towards each group's center. Last, since training is an iterative process, node embeddings and cluster centers attract each other in turns, explaining the formulation of these distinct clusters.

IV. Related Work

GRL has gained a lot of attention due to the need for automated processes that can analyze large volumes of structured data, with graphs being the hallmark of such structures.

Conventional graph embeddings: The first efforts exploited well known graph mechanisms to calculate node representations. DeepWalk [15] and Node2Vec [5] do random walks to sample the graph and train a Word2Vec model on these samples to extract the embeddings. TADW [25] applies non-negative matrix factorization on both the graph and node features to get a consistent partition. Last, DNGR [1] uses denoising auto encoders to find low dimensional representations and reconstruct the graph adjacency.

GNN-based methods: Graph Neural Networks are designed to capture graph structures, and thus have been used for learning node representations. VGAE [9] uses GCNs to form a variational autoencoder which learns to generate node embeddings, while ARVGA [14] uses adversarial learning to train the graph auto encoder. DGI [19] leverages both local and global graph information for representation learning using contrastive learning, while GIC [11] extends DGI by forming node clusters to better separate nodes in the embedding space. In particular ClusterNet [22] is also incorporated in PointSpectrum aiding in performance and validating GIC's design. Although efficient, to capture deep graph interactions GNN methods are inevitably led to over smoothing, where node representations converge to indistinguishable vectors [2], [32].

Spectral methods: On the other hand, spectral methods exploit graph filters to perform high-order graph convolution at once, thus bypassing GNNs' over smoothing. AGC [30] uses Laplacian filtering with spectral clustering to cluster nodes into groups, while AGE [3] employs an auto encoder to produce node embeddings through Laplacian smoothing. DAEGC [20] leverages an attentional network alongside soft-labeling for self supervision to construct the embeddings. While spectral methods address over smoothing, they used conventional neural networks (MLPs, CNNs) that cannot capture graph properties (e.g., equivariance) by design, but only learn the structural information contained in the smoothed input signal.

V. Conclusion

PointSpectrum is the first work to introduce into spectral GRL methods the set equivariance property, which is important when learning on unordered (i.e., graph) data. Our experimental results clearly demonstrated the performance benefits of using a set equivariant network (PointNetST) over the MLP or CNN layers that are used in spectral methods.

We deem PointSpectrum as an initial effort in the direction of integrating set equivariance with Laplacian smoothing. This is why we adopted a simple design for the model architecture, without exhaustively over-engineering its modules. Nevertheless, we showed that PointSpectrum can achieve state-of-the-art results in benchmark datasets. This brings a positive message for the efficiency, applicability and generalizability of our approach to other spectral or more generic GRL methods (e.g., VAE or GAN generative models), and motivates the need for a deeper theoretical analysis on the combination of spectral methods with set equivariant networks.

Acknowledgements

This research is co-financed by Greece and European Union through the Operational Program Competitiveness, Entrepreneurship and Innovation under the call RESEARCH-CREATE-INNOVATE (projects T2EDK-04937 and T2EDK-03898), and the European High-Performance Computing Joint Undertaking (GA No. 951732).

References

[1] Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Proc. AAAI Conference (2016)

[2] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proc. AAAI. vol. 34 (2020)

[3] Cui, G., Zhou, J., Yang, C., Liu, Z.: Adaptive graph encoder for attributed graph embedding. In: Proc. ACM KDD (2020)

[4] Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: An automatic citation indexing system. In: Proc. ACM conference on Digital libraries (1998)

[5] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proc. ACM KDD (2016)

[6] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. IEEE ICCV (2015)

[7] Keriven, N., Peyré, G.: Universal invariant and equivariant graph neural networks. NeurIPS (2019)

[8] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)

[9] Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)

[10] Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)

[11] Mavromatis, C., Karypis, G.: Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning. arXiv preprint arXiv:2009.06946 (2020)

[12] McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**(2), 127–163 (2000)

[13] Namata, G., London, B., Getoor, L., Huang, B., EDU, U.: Query-driven active surveying for collective classification. In: Workshop on Mining and Learning with Graphs (2012)

[14] Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. In: Proc. IJCAI (2018)

[15] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proc. ACM KDD (2014)

[16] Sannai, A., Takai, Y., Cordonnier, M.: Universal approximations of permutation invariant/equivariant functions by deep neural networks. arXiv preprint arXiv:1903.01939 (2019)

[17] Segol, N., Lipman, Y.: On universal equivariant set networks. In: ICLR (2019)

[18] Taubin, G.: A signal processing approach to fair surface design. In: Proc. annual conference on Computer graphics and interactive techniques (1995)

[19] Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: ICLR (2018)

[20] Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., Zhang, C.: Attributed graph clustering: A deep attentional embedding approach. In: IJCAI (2019)

[21] Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Pro. IEEE CVPR (2018)

[22] Wilder, B., Ewing, E., Dilkina, B., Tambe, M.: End to end learning and optimization on graphs. NeurIPS (2019)

[23] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: ICML (2019)

[24] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: ICML (2018)

[25] Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.: Network representation learning with rich text information. In: IJCAI (2015)

[26] Yarotsky, D.: Universal approximations of invariant maps by neural networks. Constructive Approximation pp. 1–68 (2021)

[27] Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J.: Deep sets. In: NeurIPS (2017)

[28] Zhang, M., Chen, Y.: Inductive matrix completion based on graph neural networks. In: ICLR (2019)

[29] Zhang, X., Zitnik, M.: Gnnguard: Defending graph neural networks against adversarial attacks. NeurIPS **33** (2020)

[30] Zhang, X., Liu, H., Li, Q., Wu, X.M.: Attributed graph clustering via adaptive graph convolution. In: IJCAI (2019)

[31] Zhao, L., Akoglu, L.: Pairnorm: Tackling oversmoothing in gnns. In: ICLR (2019)

[32] Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., Hu, X.: Towards deeper graph neural networks with differentiable group normalization. NeurIPS (2020)