A RECURRENT GRAPH NEURAL NETWORK FOR MULTI-RELATIONAL DATA

Vassilis N. Ioannidis[⋆] *Antonio G. Marques*[†] *Georgios B. Giannakis*[⋆]

[⋆] Digital Technology Center and Dept. of ECE, University of Minnesota, Minneapolis, USA
[†] Dept. of Signal Theory and Comms., King Juan Carlos University, Madrid, Spain

ABSTRACT

The era of "data deluge" has sparked the interest in graph-based learning methods in a number of disciplines such as sociology, biology, neuroscience, or engineering. In this paper, we introduce a graph *recurrent* neural network (GRNN) for scalable semi-supervised learning from *multi-relational data*. Key aspects of the novel GRNN architecture are the use of multi-relational graphs, the dynamic adaptation to the different relations via learnable weights, and the consideration of graph-based regularizers to promote smoothness and alleviate over-parametrization. Our ultimate goal is to design a powerful learning architecture able to: discover complex and highly non-linear data associations, combine (and select) multiple types of relations, and scale gracefully with respect to the size of the graph. Numerical tests with real datasets corroborate the design goals and illustrate the performance gains relative to competing alternatives.

***Index Terms*—** Deep neural networks, graph recurrent neural networks, graph signals, multi-relational graphs.

1. INTRODUCTION

A task of major importance in the interplay between machine learning and network science is semi-supervised learning (SSL) over graphs. In a nutshell, SSL aims at predicting or extrapolating nodal attributes given: i) the values of those attributes at a subset of nodes and (possibly) ii) additional features at all nodes. A relevant example is protein-to-protein interaction networks, where the proteins (nodes) are associated with specific biological functions, thereby facilitating the understanding of pathogenic and physiological mechanisms.

While significant progress has been achieved for this problem, most works consider that the relation among the nodal variables is represented by a single graph. This may be inadequate in many contemporary applications, where nodes may engage on multiple types of relations [1], motivating the generalization of traditional SSL approaches for *single-relational* graphs to *multi-relational* graphs[1]. In the particular case of social networks, each layer of the graph could capture a specific form of social interaction, such as friendship, family bonds, or coworker-ties [2]. Albeit their ubiquitous presence, development of SSL methods that account for multi-relational networks is only in its infancy, see, e.g., [1, 3].

Related work. A popular approach for graph-based SSL methods is to assume that the true labels are "smooth" with respect to the underlying network structure, which then motivates leveraging the topology of the network to propagate the labels and increase classification performance. Graph-induced smoothness may be captured by kernels on graphs [4, 5]; Gaussian random fields [6]; or low-rank parametric models based on the eigenvectors of the graph Laplacian or adjacency matrices [7, 8]. Alternative approaches use the graph to embed the nodes in a vector space, and classify the points [9–12]. More recently, another line of works postulates that the mapping between the input data and the labels is given by a neural network (NN) architecture that incorporates the structure of the graph [13–15]. The parameters describing the NN are then learned using labeled examples and feature vectors, and those parameters are finally used to predict the labels of the unobserved nodes. See, e.g., [15], for state-of-the-art results in SSL using a single-relational graph when nodes are accompanied with additional features.

Contributions. This paper develops a deep learning framework for SSL over multi-relational data. The main contributions are i) we postulate a (tensor-based) NN architecture that accounts for multi-relational graphs, ii) we define mixing coefficients that capture how the different relations affect the desired output and allow those to be learned from the examples, which enables identifying the underlying structure of the data; iii) at every layer we propose a recurrent feed of the data that broadens the class of (graph signal) transformations the NN implements and facilitates the diffusion of the features across the graph; and iv) in the training phase we consider suitable (graph-based) regularizers that avoid overfitting and further capitalize on the topology of the data.

2. MODELING AND PROBLEM FORMULATION

Consider a network of N nodes, with vertex set $\mathcal{V} := \{v_1, \ldots, v_N\}$, connected through I relations. The connectivity at the i-th relation is captured by the $N \times N$ matrix $\mathbf{S}_i$, and the scalar $S_{nn'i}$ represents the influence of v_n to v'_n under the i-th relation. In social networks for example, these may represent the multiple types of connectivity among people such as Facebook, LinkedIn, and Twitter; see Fig.1. The matrices $\{\mathbf{S}_i\}_{i=1}^I$ are collected in the $N \times N \times I$ tensor $\underline{\mathbf{S}}$. The graph-induced neighborhood of v_n for the i-th relation is

$$\mathcal{N}_n^{(i)} := \{n' : S_{nn'i} \neq 0, \;\; v'_n \in \mathcal{V}\}. \tag{1}$$

We associate an $F \times 1$ feature vector $\mathbf{x}_n$ to the n-th node, and collect those vectors in the $N \times F$ feature matrix $\mathbf{X} := [\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top]^\top$. The entry X_{np} may denote, for example, the salary of the n-th individual in the LinkedIn social network.

We also consider that each node n has a label of interest $y_n \in \{0, \ldots, K-1\}$, which may represent, for example, the education level of a person. In SSL we have access to the labels only at a subset of nodes $\{y_n\}_{n \in \mathcal{L}}$, with $\mathcal{L} \subset \mathcal{V}$. This partial availability may be attributed to privacy concerns (medical data); energy considerations (sensor networks); or unrated items (recommender systems). The $N \times K$ matrix $\mathbf{Y}$ is the "one-hot" representation of the true nodal labels, that is, if $y_n = k$ then $Y_{n,k} = 1$ and $Y_{n,k'} = 0, \forall k' \neq k$.

The work in this paper has been supported by USA NSF grants 171141, 1500713, and 1442686, and by the Spanish grants MINECO KLINILYCS (TEC2016-75361-R) and Instituto de Salud Carlos III DTS17/00158.

[1] Many works in the literature refer to these graphs as multi-layer graphs.

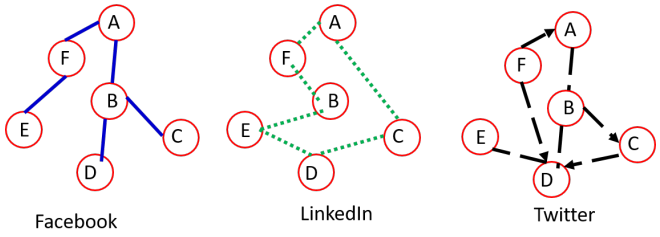


Fig. 1: A social multi-relational network of $N = 6$ people.

The goal of this paper is to develop a *deep learning architecture* based on *multi-relational graphs* that, using as input the features in $\mathbf{X}$, maps each node n to a corresponding label y_n and, hence, estimates the unavailable labels.

3. PROPOSED GRNN ARCHITECTURE

Deep learning architectures typically process the input information using a succession of L hidden layers. Each of the layers is composed of a conveniently parametrized linear transformation, a scalar nonlinear transformation, and, oftentimes, a dimensionality reduction (pooling) operator. The intuition is to combine nonlinearly local features to progressively extract useful information [16]. GNNs tailor these operations to the graph that supports the data [13], including the linear [17], nonlinear [17] and pooling [14] operators. In this section, we describe the architecture of our novel multi-relational GRNN, that inputs the known features at the first layer and outputs the predicted labels at the last layer. We first present the operation of the GNN, GRNN, and output layers, and finally discuss the training of our NN.

3.1. Single layer operation

Let us consider an intermediate layer (say the lth one) of our architecture. The output of that layer is the $N \times I \times P^{(l)}$ tensor $\check{\underline{\mathbf{Z}}}^{(l)}$ that holds the $P^{(l)} \times 1$ feature vectors $\check{\mathbf{z}}_{ni}^{(l)}, \forall n, i$, with $P^{(l)}$ being the number of output features at l. Similarly, the $N \times I \times P^{(l-1)}$ tensor $\check{\underline{\mathbf{Z}}}^{(l-1)}$ represents the input to the layer. Since our focus is on predicting labels on all nodes, we do not consider a dimensionality reduction (pooling) operator in the intermediate layers. As a result, the mapping from $\check{\underline{\mathbf{Z}}}^{(l-1)}$ to $\check{\underline{\mathbf{Z}}}^{(l)}$ can be split into two steps. First, we define a linear transformation that maps the $N \times I \times P^{(l)}$ tensor $\check{\underline{\mathbf{Z}}}^{(l-1)}$ into the $N \times I \times P^{(l)}$ tensor $\underline{\mathbf{Z}}^{(l)}$. The intermediate feature output $\underline{\mathbf{Z}}^{(l)}$ is then processed elementwise using a scalar nonlinear transformation $\sigma(\cdot)$ as follows

$$\check{\underline{Z}}_{inp}^{(l)} := \sigma(\underline{Z}_{inp}^{(l)}). \tag{2}$$

Collecting all the elements in (2), we obtain the output of the l-th layer $\check{\underline{\mathbf{Z}}}^{(l)}$. A common choice for $\sigma(\cdot)$ is the rectified linear unit (ReLU), i.e. $\sigma(c) = \max(0, c)$ [16].

Hence, the main task is to define a linear transformation that maps $\check{\underline{\mathbf{Z}}}^{(l-1)}$ to $\underline{\mathbf{Z}}^{(l)}$ and is tailored to our problem setup. Traditional convolutional NNs (CNNs) typically consider a small number of trainable weights and then generate the linear output as a convolution of the input with these weights [16]. The convolution combines values of close-by inputs (consecutive time instants, or neighboring pixels) and thus extracts information of local neighborhoods. GNNs have generalized CNNs to operate on graph data by replacing the convolution with a graph filter whose parameters are also learned [13]. This preserves locality, reduces the degrees of freedom of the transformation, and leverages the structure of the graph.

To that end, we first consider a step that combines linearly the information within a graph neighborhood. Since the neighborhood depends on the particular relation (1), we obtain for the i-th relation

$$\mathbf{h}_{ni}^{(l)} := \sum_{n' \in \mathcal{N}_n^{(i)}} S_{nn'i} \check{\mathbf{z}}_{n'i}^{(l-1)}. \tag{3}$$

While the entries of $\mathbf{h}_{ni}^{(l)}$ depend only on the one-hop neighbors of n (one-hop diffusion), the successive application of this operation across layers will increase the reach of the diffusion, spreading the information across the network. An alternative to account for multiple hops is to apply successively (3) within one layer, which is left as future work. Next, to learn features in the graph data, we combine the entries in $\mathbf{h}_{ni}^{(l)}$ using (trainable) parameters as follows

$$\underline{Z}_{inp}^{(l)} := \sum_{i'=1}^{I} R_{ii'p}^{(l)} \mathbf{h}_{ni}^{(l)\top} \mathbf{w}_{ni'p}^{(l)}, \tag{4}$$

where the $P^{(l-1)} \times 1$ vector $\mathbf{w}_{ni'p}^{(l)}$ mixes the features and $R_{ii'p}^{(l)}$ mixes the outputs at different graphs. The $P^{(l-1)} \times N \times I \times P^{(l)}$ tensor $\underline{\mathbf{W}}^{(l)}$ collects the feature mixing weights $\{\mathbf{w}_{ni'p}^{(l)}\}$, while the $I \times I \times P^{(l)}$ tensor $\underline{\mathbf{R}}^{(l)}$ collects the graph mixing weights $\{R_{ii'p}^{(l)}\}$. Another key contribution of this paper is the consideration of $\underline{\mathbf{R}}$ as a training parameter, which endows the GNN with the ability of learning how to mix (combine) the different relations encoded in the multi-relational graph. Clearly, if prior information on the dependence among relations exists, this can be used to constrain the structure $\underline{\mathbf{R}}$ (e.g., by imposing to be diagonal or sparse). Upon collecting all the scalars $\{\underline{Z}_{inp}^{(l)}\}$ in the $I \times N \times P^{(l)}$ tensor $\underline{\mathbf{Z}}^{(l)}$, we summarize (3), (4) as follows

$$\underline{\mathbf{Z}}^{(l)} := f(\check{\underline{\mathbf{Z}}}^{(l-1)}; \boldsymbol{\theta}_z^{(l)}), \quad \text{where} \tag{5}$$

$$\boldsymbol{\theta}_z^{(l)} := [\text{vec}(\underline{\mathbf{W}}^{(l)}); \text{vec}(\underline{\mathbf{R}}^{(l)})]^\top. \tag{6}$$

Recurrent GNN layer: Successive application of L GNN layers diffuses the input $\mathbf{X}$ across the L-hop graph neighborhood, cf. (3). However, the exact size of the relevant neighborhood is not always known a priori. To endow our architecture with increased flexibility, we propose a recurrent GNN (GRNN) layer that inputs $\mathbf{X}$ at each l and, thus, captures multiple types of diffusion. Hence, the linear operation in (5) is replaced by the recurrent (autoregressive) linear tensor mapping [16, Ch. 10]

$$\underline{\mathbf{Z}}^{(l)} := f(\check{\underline{\mathbf{Z}}}^{(l-1)}; \boldsymbol{\theta}_z^{(l)}) + f(\underline{\mathbf{X}}; \boldsymbol{\theta}_x^{(l)}) \tag{7}$$

where $\boldsymbol{\theta}_x^{(l)}$ encodes trainable parameters, cf. (6). When viewed as a transformation from $\underline{\mathbf{X}}$ to $\underline{\mathbf{Z}}^{(l)}$, the operator in (7) implements a broader class of graph diffusions than the one in (5). If $l = 3$ for example, then the first summand in (7) is a 1-hop diffusion of a signal that corresponded to a 2-hop (nonlinear) diffused version of $\mathbf{X}$ while the second summand diffuses $\mathbf{X}$ in one hop. At a more intuitive level, the presence of the second summand also guarantees that the impact of $\mathbf{X}$ in the output does not vanishes as the number of layers grow. The autoregressive mapping in (7) allows the architecture to further model time-varying inputs and labels, which motivates our future work towards predicting dynamic processes over multi-relational graphs[2].

[2]The recursive feed of $\mathbf{X}$ is also known as a skip connection [18].

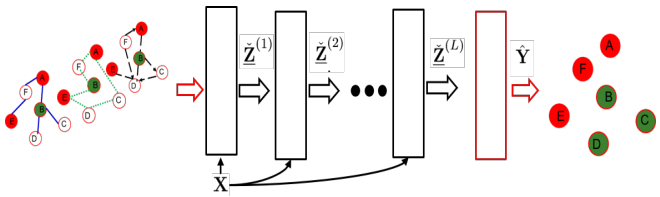


Fig. 2: GRNN; L hidden (black) and one output (red) layers

3.2. Initial and final layers

The operation of the first and last layers is very simple. Regarding layer $l = 1$, the input $\check{\underline{\mathbf{Z}}}^{(0)}$ is defined using $\underline{\mathbf{X}}$ as

$$\check{\mathbf{z}}_{ni}^{(0)} = \mathbf{x}_n \quad \text{for all} \quad (n, i). \tag{8}$$

On the other hand, the output of our graph architecture is obtained by taking the output of the layer $l = L$ and applying

$$\hat{\mathbf{Y}} := g(\check{\underline{\mathbf{Z}}}^{(L)}; \boldsymbol{\theta}_g), \tag{9}$$

where $g(\cdot)$ is a nonlinear function, $\hat{\mathbf{Y}}$ is an $N \times K$ matrix, $\hat{Y}_{n,k}$ represents the probability that $y_n = k$, and $\boldsymbol{\theta}_g$ are trainable parameters. The function $g(\cdot)$ depends on the specific application, with the normalized exponential function (softmax) being a popular choice for classification problems.

For notational convenience, the global mapping from $\mathbf{X}$ to $\hat{\mathbf{Y}}$ dictated by our GRNN architecture –i.e., by the sequential application of (7)-(9)– is denoted as

$$\hat{\mathbf{Y}} := \mathcal{F}(\mathbf{X}; \{\boldsymbol{\theta}_z^{(l)}\}, \{\boldsymbol{\theta}_x^{(l)}\}, \boldsymbol{\theta}_g), \tag{10}$$

and represented in the block diagram depicted in Fig. 2.

3.3. Training and graph-smooth regularizers

The proposed architecture depends on the weights in (7) and (9). We estimate these weights by minimizing the discrepancy between the estimated labels and the given ones. Hence, we arrive at the following minimization objective

$$\begin{aligned} \min_{\{\boldsymbol{\theta}_z^{(l)}\},\{\boldsymbol{\theta}_x^{(l)}\},\boldsymbol{\theta}_g} & \mathcal{L}_{tr}(\hat{\mathbf{Y}}, \mathbf{Y}) + \mu_1 \sum_{i=1}^{I} \mathrm{Tr}(\hat{\mathbf{Y}}^\top \mathbf{S}_i \hat{\mathbf{Y}}) \\ & + \mu_2 \rho(\{\boldsymbol{\theta}_z^{(l)}\}, \{\boldsymbol{\theta}_x^{(l)}\}) + \lambda \sum_{l=1}^{L} \|\underline{\mathbf{R}}^{(l)}\|_1 \\ \text{s.t.} \quad & \hat{\mathbf{Y}} = \mathcal{F}(\mathbf{X}; \{\boldsymbol{\theta}_z^{(l)}\}, \{\boldsymbol{\theta}_x^{(l)}\}, \boldsymbol{\theta}_g). \end{aligned} \tag{11}$$

In our classification setup, a sensitive choice for the fitting cost is to use $\mathcal{L}_{tr}(\hat{\mathbf{Y}}, \mathbf{Y}) := -\sum_{n \in \mathcal{L}} \sum_{k=1}^{K} Y_{nk} \ln \hat{Y}_{nk}$ the cross-entropy loss function over the labeled examples.

Note also that three regularizers have been considered. The first (graph-based) regularizer promotes smooth label estimates over the graphs [5], and $\rho(\cdot)$ is an $\mathcal{L}_2$ norm over the GRNN parameters that is typically used to avoid overfitting [16]. Finally, the $\mathcal{L}_1$ norm in the third regularizer promotes learning sparse mixing coefficients and, hence, promotes activating only a subset of relations at each l. The backpropagation algorithm [19] is employed to minimize (11). The computational complexity of evaluating (7) scales linearly with the number of nonzero entries in $\underline{\mathbf{S}}$ (edges), cf. (3).

To recap, while most of the works in the GNN literature use a single graph with one type of diffusion [13, 15], we have proposed a (recurrent) multi-relational GNN architecture that: adapts to each graph with $\underline{\mathbf{R}}$; uses a simple but versatile recurrent tensor mapping (7); and includes several types of graph-based regularizers.

4. NUMERICAL TESTS

We test the proposed GNN with $L = 2$, $P^{(1)} = 64$, and $P^{(2)} = K$. The regularization parameters $\{\mu_1, \mu_2, \lambda\}$ are chosen based on the performance of the GRNN in the validation set for each experiment. For the training stage, an ADAM optimizer with learning rate 0.005 was employed [20], for 300 epochs[3] with early stopping at 60 epochs[4]. The multiple layers of the graph in this case are formed using κ-nearest neighbors graphs for different values of κ (i.e., different number of neighbors). This method computes the link between n and n' based on the Euclidean distance of their features $\|\mathbf{x}_n - \mathbf{x}'_n\|_2^2$. The simulations were run using TensorFlow [21] and the code is available online[5].

4.1. Robustness of GRNN

This section reports the performance of the proposed architecture under perturbations. Oftentimes, the available topology and feature vectors might be corrupted (e.g. due to privacy concerns or because adversarial social users manipulate the data to sway public opinion). In those cases, the observed $\underline{\mathbf{S}}$ and $\mathbf{X}$ can be modeled as

$$\underline{\mathbf{S}} = \underline{\mathbf{S}}_{tr} + \underline{\mathbf{O}}_{\underline{\mathbf{S}}} \tag{12}$$

$$\mathbf{X} = \mathbf{X}_{tr} + \mathbf{O}_{\mathbf{X}}. \tag{13}$$

where $\underline{\mathbf{S}}_{tr}$ and $\mathbf{X}_{tr}$ represent the *true* topology and features and $\underline{\mathbf{O}}_{\underline{\mathbf{S}}}$ and $\mathbf{O}_{\mathbf{X}}$ denote the corresponding additive perturbations. We draw $\underline{\mathbf{O}}_{\underline{\mathbf{S}}}$ and $\mathbf{O}_{\mathbf{X}}$ from a zero mean white Gaussian distribution with specified signal to noise ratio (SNR). The robustness of our method is tested in two datasets: i) A synthetic dataset of $N = 1000$ points that belong to $K = 2$ classes generated as $\mathbf{x}_n \in \mathbb{R}^{F \times 1} \sim \mathcal{N}(\mu, 0.4)$, $n = 1, \ldots, 1000$ with $F = 10$ and $\mu = 0, 1$ corresponding to the different classes. ii) The ionosphere dataset, which contains $N = 351$ data points with $F = 34$ features that belong to $K = 2$ classes [22]. We generate κ-nearest neighbors graphs by varying κ, and observe $|\mathcal{L}| = 200$ and $|\mathcal{L}| = 50$ nodes uniformly at random.

With this simulation setup, we test the different GRNNs in SSL for increasing the SNR of $\underline{\mathbf{O}}_{\underline{\mathbf{S}}}$ (Figs. 3a, 3c) and $\mathbf{O}_{\mathbf{X}}$ (Figs. 3b, 3d). We deduce from the classification performance of our method in Fig. 3 that multiple graphs leads to learning more robust representations of the data, which testifies to the merits of proposed multi-relational architecture.

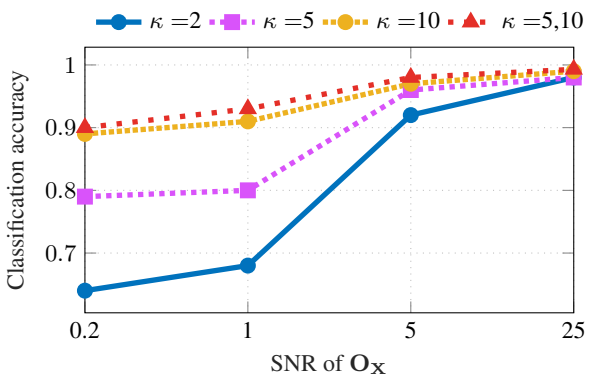
4.2. Classification of citations graphs

We test our architecture with three citation network datasets [23]. The citation graph is denoted as $\mathbf{S}_0$, its nodes correspond to different documents from the same scientific category, and $S_{nn'0} = 1$ implies that paper n cites paper n'. Each document n is associated with a feature vector $\mathbf{x}_n$ that measures the frequency of a set of words, as well as with a label y_n that indicates the document's subcategory.

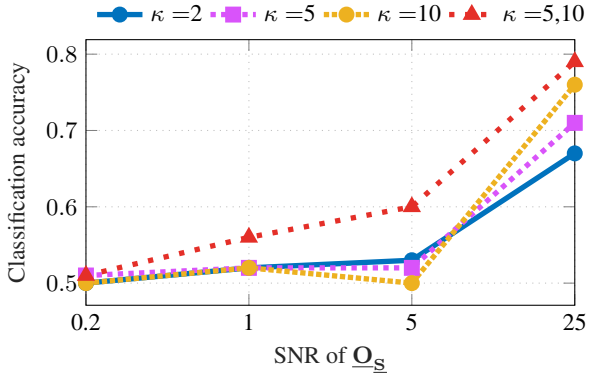
[3] An epoch is a cycle through all the training examples

[4] Training stops if the validation loss does not decrease for 60 epochs

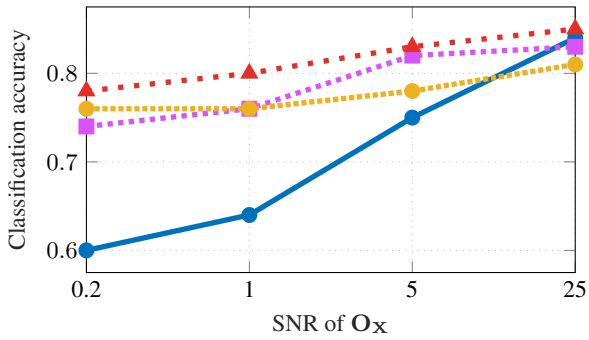
[5] https://sites.google.com/site/vasioannidispw/github



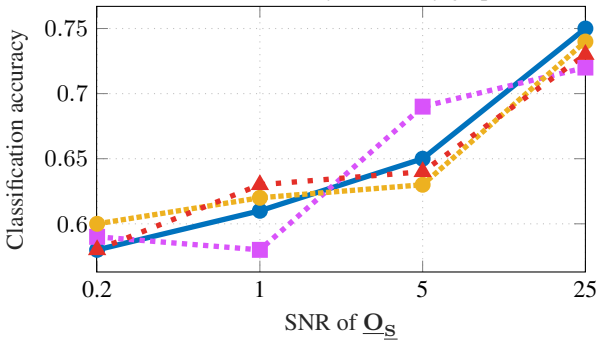
(a) Classification accuracy with noisy features.



(b) Classification accuracy with noisy graphs.



(c) Classification accuracy with noisy features.



(d) Classification accuracy with noisy graphs.

Fig. 3: Classification accuracy on synthetic (a), (b) and ionosphere (c), (d).

Dataset	**Nodes** N	**Classes** K	**Features** F	$\lvert\mathcal{L}\rvert$
Cora	2,708	7	1,433	140
Citeseer	3,327	6	3,703	120
Pubmed	19,717	3	500	30

Table 1: Citation datasets

Method	**Citeseer**	**Cora**	**Pubmed**
ManiReg [4]	60.1	59.5	70.7
SemiEmb [9]	59.6	59.0	71.7
LP [6]	45.3	68.0	63.0
Planetoid [10]	64.7	75.7	77.2
GCN [15]	70.3	81.5	79.0
GRNN[6]	**70.8**	**82.8**	**79.5**
GRNN (multi-relational)[7]	**70.9**	**81.7**	**79.2**

Table 2: Classification accuracy for citation datasets

"Cora" contains papers related to machine learning, "Citeseer" includes papers related to computer and information science, while "Pubmed" contains biomedical papers, see also Table 1.

To facilitate comparison, we reproduce the same experimental setup than in [15], i.e., the same split of the data in train, validation, and test sets. We test two architectures: a) a GRNN using only the original citation graph $\mathbf{S}_0$ (and, hence, with $I = 1$); and b) a multirelational GRNN that uses an extra 1-nearest neighbor graph (so that $I = 2$). Table 2 reports the classification accuracy of various SSL methods. It is observed that: i) GNN approaches (ours, as well as [15]) outperform competing alternatives, ii) our GRNN schemes always outperform [15] (either with $I = 1$ or $I = 2$). This illustrate the potential benefits of the recurrent feed in (7) –not used in [15]– as well as the use of multi-relational graphs.

[6] $\mu_1 = 5 \times 10^{-5}, \mu_2 = 5 \times 10^{-5}, \lambda = 10^{-4}$, dropout rate=0.9
[7] $\mu_1 = 2 \times 10^{-6}, \mu_2 = 5 \times 10^{-5}, \lambda = 10^{-4}$, dropout rate=0.9

5. CONCLUSIONS

This paper put forth a novel deep learning framework for SSL that utilized an autoregressive multi-relational graphs architecture to sequentially process the input data. Instead of committing a fortiori to a specific type of diffusion, our novel GRNN learns the diffusion pattern that best fits the data. The proposed architecture is able to handle scenarios where nodes engage in multiple relations, can be used to reveal the structure of the data, and is computationally affordable, since the number of operations scaled linearly with respect to the number of graph edges. Our approach achieves state-of-the-art classification results on graphs when nodes are accompanied by feature vectors. Future research includes investigating robustness to adversarial topology perturbations, predicting time-varying labels, and designing of pooling operators.

6. REFERENCES

[1] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.

[2] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge university press, 1994, vol. 8.

[3] V. N. Ioannidis, P. A. Traganitis, Y. Shen, and G. B. Giannakis, "Kernel-based semi-supervised learning over multilayer graphs," in *Proc. IEEE Int. Workshop Sig. Process. Advances Wireless Commun.*, Kalamata, Greece, Jun. 2018.

[4] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.

[5] V. N. Ioannidis, M. Ma, A. Nikolakopoulos, G. B. Giannakis, and D. Romero, "Kernel-based inference of functions on graphs," in *Adaptive Learning Methods for Nonlinear System Modeling*, D. Comminiello and J. Principe, Eds. Elsevier, 2018.

[6] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, Washington, USA, Jun. 2003, pp. 912–919.

[7] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[8] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Sig. Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[9] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.

[10] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn.*, New York, USA, Jun. 2016.

[11] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive diffusions for scalable learning over graphs," *arXiv preprint arXiv:1804.02081*, 2018.

[12] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv preprint arXiv:1612.07659*, 2016.

[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Sig. Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[14] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," in *IEEE Data Science Workshop*, Lausanne, Switzerland, Jun. 2018, pp. 1–5.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. on Learn. Represantions*, Toulon, France, Apr. 2017.

[16] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

[17] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.

[18] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," *arXiv preprint arXiv:1606.08921*, 2016.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[20] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learn. Represantions*, San Diego, CA, USA, May 2015.

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, Savannah, GA, USA, Nov. 2016, pp. 265–283.

[22] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[23] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.