Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Transfer Learning for Deep Learning on Graph-Structured Data

Jaekoo Lee, Hyunjae Kim, Jongsun Lee, Sungroh Yoon
Electrical and Computer Engineering
Seoul National University
Seoul 08826, Republic of Korea
sryoon@snu.ac.kr

Abstract

Graphs provide a powerful means for representing complex interactions between entities. Recently, new deep learning approaches have emerged for representing and modeling graph-structured data while the conventional deep learning methods, such as convolutional neural networks and recurrent neural networks, have mainly focused on the grid-structured inputs of image and audio. Leveraged by representation learning capabilities, deep learning-based techniques can detect structural characteristics of graphs, giving promising results for graph applications. In this paper, we attempt to advance deep learning for graph-structured data by incorporating another component: transfer learning. By transferring the intrinsic geometric information learned in the source domain, our approach can construct a model for a new but related task in the target domain without collecting new data and without training a new model from scratch. We thoroughly tested our approach with large-scale real-world text data and confirmed the effectiveness of the proposed transfer learning framework for deep learning on graphs. According to our experiments, transfer learning is most effective when the source and target domains bear a high level of structural similarity in their graph representations.

Introduction

Recently, many deep neural network models have been adopted successfully in various fields (LeCun, Bengio, and Hinton 2015; Schmidhuber 2015). In particular, convolutional neural networks (CNN) (Krizhevsky, Sutskever, and Hinton 2012) for image and video recognition and recurrent neural networks (RNN) (Sutskever, Vinyals, and Le 2014) for speech and natural language processing (NLP) often deliver unprecedented levels of performance. Deep learning has also triggered advances in implementing human-level intelligence (e.g., in the game of Go (Silver et al. 2016)).

CNN and RNN extract data-driven features from input data (e.g., image, video, and audio data) structured in typically low-dimensional regular grids (see Fig. 1, top). Such grid structures are often assumed to have statistical characteristics (e.g., stationarity and locality) to facilitate the modeling process. Learning algorithms then take advantage of this assumption and boost performance by reducing the complexity of parameters (Schmidhuber 2015; Bruna et al. 2013; Henaff, Bruna, and LeCun 2015).

In reality, there exist a wide variety of data types in which we need more general non-grid structures to represent and model complex interactions among entities. Examples include social media mining and protein interaction studies. For such applications, a graph can provide a natural way of representing entities and their interactions (Deo 2016). For graph-structured input, it is more challenging to find the statistical characteristics that can be assumed for grid-structured input (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015).

Theoretical challenges including the above and practical limitations, such as data quantity/quality and training efficiency, make it difficult to apply conventional deep learning approaches directly, igniting research on adapting deep learning to graph-structured data (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015; Jain et al. 2015; Li et al. 2015). In many graph analysis methods, the structural properties derived from input graphs play a crucial role in uncovering hidden patterns (Koutra, Vogelstein, and Faloutsos 2013; Lee, Kim, and Yoon 2015). The representation learning capability of deep networks is useful for automatically detecting data-driven structural features, and deep learning approaches have reported promising results.

In this paper, we attempt to advance deep learning for graph-structured data by incorporating another key component: transfer learning (Pan and Yang 2010). By overcoming the common assumption that training and test data should be drawn from the same feature space and distribution, transfer learning between task domains can alleviate the burden of collecting data and training models for a new task. Given the importance of structural characteristics in graph analysis, the core of our proposal is to transfer the data-driven structural features learned by deep networks from a source domain to a target domain, as informally shown in Fig. 1 (bottom). In the context of graphs, we call the transferred information the *intrinsic geometric* information.

Starting from this intuitive baseline, we need to fill in many details to implement transfer learning for deep learning on graph data. In particular, we need to answer two important questions: (**Q1**) under what condition can we expect a successful knowledge transfer between task domains and (**Q2**) how do we actually perform the transfer most effec-

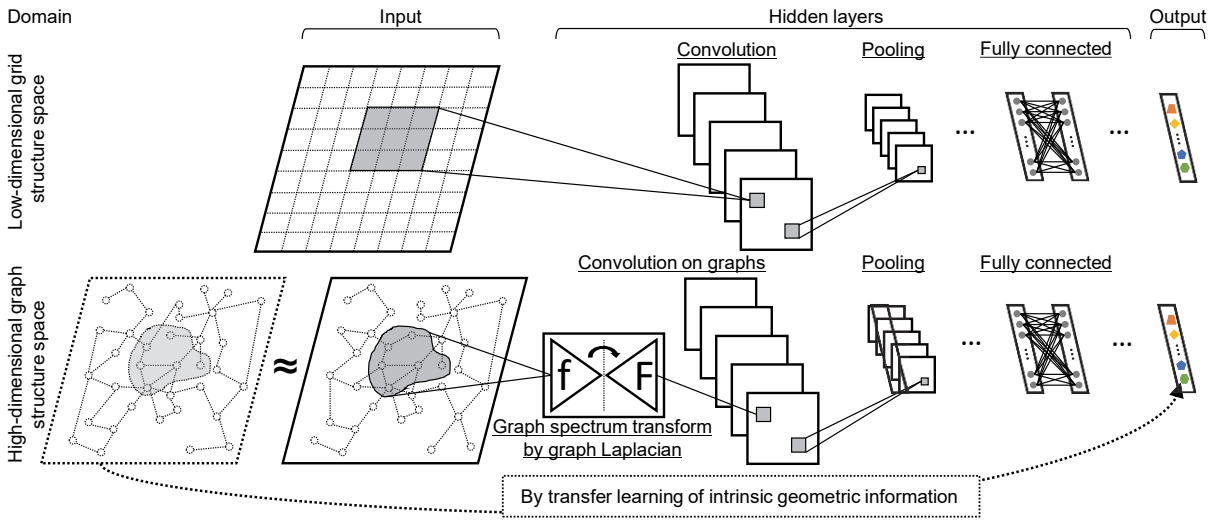

Figure 1: Conventional CNN works on a regular grid domain (top); proposed transfer learning framework for CNN, which can transfer intrinsic geometric information obtained from a source graph domain to a target graph domain (bottom).

tively? This paper tries to address these questions.

To demonstrate the effectiveness of our approach, we tested it with large-scale public NLP datasets for text classification (Zhang, Zhao, and LeCun 2015). Each dataset contained a corpus of news articles, Internet reviews, or ontology entries. We represented a dataset (e.g., Amazon reviews) with a graph to capture the interactions among the words in the dataset. We then used the spectral CNN (SCNN) (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015) to model the graph using neural networks. The learned model can be used for classifying unseen texts from the same data source (Amazon). Furthermore, our experimental results confirmed that our transfer learning methodology allows us to implicitly derive a model for classifying texts from another source (e.g., Yelp reviews) without collecting new data and without repeating all the learning procedures from scratch.

Our specific contributions can be summarized as follows:

- We proposed a new transfer learning framework for deep learning on input data in non-grid structure such as graphs. To the best of the authors' knowledge, this work is the first attempt of its kind. Adopting our approach will relieve the burden of re-collecting data and re-training models for related tasks.
- To address **Q1**, we investigated the conditions for successful knowledge transfers between graph domains. We conjectured that two graphs with similar structural characteristics would give better results and confirmed it by comparing graph similarity and transfer learning accuracy.
- To answer **Q2**, we tested diverse alternatives to the components of the proposed framework: graph generation, input representation, and deep network construction. In particular, to improve the SCNN model for extracting data-driven structural features from graphs, we analyzed and optimized the key factors that affect the performance of SCNN (e.g., the method to quantify spectral features of a graph).
- We performed an extensive set of experiments, using both synthetic and real-world data, to show the effectiveness of our approach.

Related Work

Graphs can provide a general way of representing the diverse interactions of entities and have been studied extensively (Sonawane and Kulkarni 2014). In addition to studies on representation and quantification of relations and similarities (Koutra, Vogelstein, and Faloutsos 2013), various studies focused on large-scale graph data and use of structural information. Recently, deep learning methods to automatically extract structural characteristics from graphs have been proposed (Duvenaud et al. 2015; Li et al. 2015).

Examples of deep learning applied to non-grid, non-Euclidean space include graph wavelets from applying deep auto-encoders to graphs and using the properties of automatically extracted features (Rustamov and Guibas 2013), analysis of molecular fingerprints of proteins saved as graphs (Duvenaud et al. 2015), and a CNN-based model for handling tree structures in the context of programming language processing (Mou et al. 2016).

Particularly relevant to our approach is the localized SCNN model (Boscaini et al. 2015), which is a deep learning approach that can extract the properties of deformable shapes. The generalized SCNN model (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015), a key component of our framework, borrowed the Fourier transform concept from the signal processing field in order to apply CNNs in a grid domain to a graph-structured domain. In this model, the convolutional operation was re-defined for graphs.

More recently, the restricted Boltzmann machine (LeCun, Bengio, and Hinton 2015) was used to learn structural features from graphs in an unsupervised manner for classification (Niepert, Ahmed, and Kutzkov 2016). For efficient and

scalable semi-supervised classification of graph-structured data, the first-order approximation of spectral graph convolutions was utilized (Kipf and Welling 2016). Our method could adopt these approaches as its base learning model to improve the effectiveness of transfer learning.

Proposed Method

Fig. 2 presents a diagram illustrating the overall flow of the proposed method, which consists of five steps, A–E. The first three steps are to produce a graph from input and to identify unique structural features from the graph. The last two steps are to apply transfer learning based on the learned features and graph similarity to carry out inference.

Step A: Graph Production

We represent data elements of input data and their interactions and relations as nodes and edges, respectively, in a graph. From an input dataset, we construct an undirected, connected, and weighted graph $G = (V, E, A)$, where V and E represent the sets of vertices and edges, respectively, and A denotes the weighted adjacency matrix. Assume that $|V| = N$ and $|E| = M$.

We utilize two recent techniques to derive a graph (more specifically, the edge set E) from input data: co-occurrence graph estimation (CoGE) (Sonawane and Kulkarni 2014) and supervised graph estimation (SGE) (Henaff, Bruna, and LeCun 2015). CoGE directly quantifies the closeness of data elements based on the frequency of co-occurrence, while SGE automatically learns a similarity features among elements through a fully connected network model.

B: Representation of Graphs in Spectral Domain

We extract the intrinsic geometric characteristics of the entire graph by deriving (non-)normalized Laplacian matrix L of the graph constructed in step A. For a graph domain, L provides the values for graph spectral bases in the convolution operation of SCNN (Mohar 1997; Koutra, Vogelstein, and Faloutsos 2013).

We consider three types of L: the non-normalized Laplacian (L^{basic}), the random walk-based normalized Laplacian (L^{rw}), and the random walk with restart based normalized Laplacian (L^{rwr}) given by (Tong, Faloutsos, and Pan 2006):

$$L^{\text{basic}} = D - A \tag{1}$$

$$L^{\text{rw}} = D^{-1}(D - A) = I - D^{-1}A \tag{2}$$

$$L^{\text{rwr}} = [I + \epsilon^2 D - \epsilon A]^{-1} \approx [I - \epsilon A]^{-1} \tag{3}$$

$$\approx I + \epsilon A + \epsilon^2 A^2 + \cdots \tag{4}$$

where D represents the degree matrix[1] of the graph, and ϵ represents the probability of restart. Note that the approximation in Eq. (3) is attained by attenuating neighboring influence, while the approximation in Eq. (4) is attained by belief propagation and its fast approximation.

L is a symmetric matrix that can be decomposed through the diagonalization by combining eigenvalues λ_l and the corresponding orthogonal eigenvectors $u_l(n)$, where l is the order of an eigenvalue, and $n \in [1, N]$ is the index of a node (Mohar 1997).

Recall that a function $f : V \mapsto \mathbb{R}$ defined on the nodes of graph G can be represented by a vector $f \in \mathbb{R}^N$ with the n-th dimension of f indicating the value at the n-th vertex in V (Shuman et al. 2013; Shuman, Ricaud, and Vandergheynst 2016). As in the Fourier transform, the eigenfunctions of L represent the function f defined by the nodes in the graph: $f_G(n) = \sum_{l=0}^{N-1} \hat{f}_G(\lambda_l) u_l(n) \leftrightarrow \hat{f}_G(\lambda_l) = \sum_{n=1}^{N} f_G(n) u_l(n)$, where $\hat{f}$, the transformed function of f, is represented by a set of basis eigenvectors. The Parseval's theorem also holds (i.e., the energy of the transformed function is the same as that of the original function), and $\langle f, g \rangle = \langle \hat{f}, \hat{g} \rangle$ for two functions f and g, verifying the consistency between the two domains (Chung 1997; Shuman, Ricaud, and Vandergheynst 2016).

This indicates that an input function defined on the vertex domain of a graph can be converted into the corresponding graph spectral domain by using the concept of Fourier analysis on graphs. The generalized convolutional operation (denoted by $*_G$) of functions f and g can be defined by the diagonalized linear multiplication in the spectral domain as follows (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015; Shuman, Ricaud, and Vandergheynst 2016):

$$(f *_G g)(n) = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) \hat{g}(\lambda_l) u_l(n)$$

which can also be expressed as

$$f *_G g = \hat{g}(L) f = U \begin{bmatrix} \hat{g}(\lambda_0) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{g}(\lambda_{N-1}) \end{bmatrix} U^T f \tag{5}$$

where U is a matrix having the eigenvectors of the graph Laplacian in its columns that quantify the intrinsic structural geometry of the entire graph domain and serve as the spectral bases of a graph. This matrix functions as the Fourier transform into the graph spectral domain. In this regard, a receptive filter learned through the convolution operation in a convolution layer of a CNN in a regular grid domain can be regarded as a matrix on g, which is diagonalized by $\hat{g}(\lambda_i)$ $(0 \le i \le N-1)$ elements on input f defined in the graph domain provided by Eq. (5).

For conventional CNNs, the j-th output $x_{k+1,j}$ from the k-th layer is defined as

$$x_{k+1,j} = h\left(\sum_{i=1}^{\phi_{k-1}} F_{k,i,j} * x_{k,i} \right),\ j = 1, \ldots, \phi_k \tag{6}$$

where ϕ_k is the number of feature maps, x_k is the input of k-th layer, h is a nonlinear function, and $F_{k,i,j}$ is the filter matrix from the i-th feature map to the j-th feature map.

For the SCNN, the transform of input x_k of size $n \times \phi_{k-1}$

[1] A diagonal matrix that shows the degree (i.e., the number of edges attached) of each node.

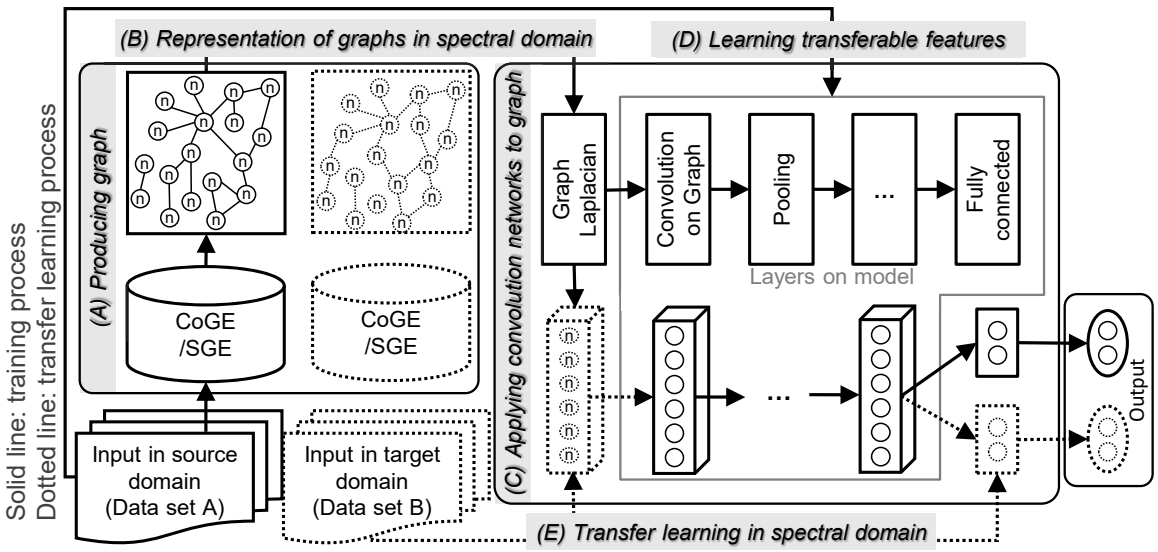

Figure 2: Overview of the proposed method.

into output x_{k+1} of size $n \times \phi_k$ is given by

$$x_{k+1,j} = h\left(U \sum_{i=1}^{\phi_{k-1}} F_{k,i,j} U^T x_{k,i}\right), \; j = 1, \ldots, \phi_k \quad (7)$$

where h is a nonlinear function, and $F_{k,i,j}$ is a diagonal matrix. This implies that training the weights of learnable filters are the same as training the multipliers on the eigenvalues of the Laplacian (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015). This characterizes the SCNN, a generalized CNN model that has several filter banks through generalized convolutional operations on a graph.

We augment the SCNN model so that it can support spatial locality, which is made independent of input size by using windowed smoothing filters. They are defined as $\hat{P}_k(l) = \sum_{k=0}^{K} a_k \lambda_l^k$ for $K < N$, based on the polynomial kernel a_k with degree K (Shuman, Ricaud, and Vandergheynst 2016). This is based on the fact (originally observed in signal processing) that the smoothness in the spectral domain can have spatial decay or local features in the original domain. We implement this idea using the eigenvectors of the subsampled Laplacian as the low-frequency eigenvectors of the Laplacian (Boscaini et al. 2015).

C: Applying Convolutional Networks to Graphs

We train the SCNN model by using the information obtained through the previous steps to represent the geometric information of local behaviors from the surface of a structural graph domain. The model has a hierarchical structure consisting of layers for convolutional and pooling, and a fully connected layer as shown in Fig. 2. The training determines the weights of each layer by minimizing the task-specific cost (loss) function. The model can learn various data-driven features by re-defining the convolution operation with the spectral information of the structural graph domain (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015).

D: Learning Transferable Features

Once the model training is completed, it contains data-driven features for the graph-structured data derived from the input in steps A and B. As stated in Introduction, the core of our proposal is to transfer the information on structural characteristics of a graph learned by deep learning. The features learned in step C provide this information.

E: Transfer Learning in Spectral Domain

According to (Pan and Yang 2010), a *domain* in the context of transfer learning consists of a feature space $\mathcal{X}$ and a probability distribution $P(X)$, where $X \in \mathcal{X}$. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, we can denote a *task* by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ with a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$ that is learned from training data $\{x, y\}$ where $x \in X$ and $y \in \mathcal{Y}$. The objective of general transfer learning is then to improve learning $f_T(\cdot)$ in the target domain $\mathcal{D}_T$ by exploiting the knowledge in the source domain $\mathcal{D}_S$ and task $\mathcal{T}_S$.

In the present context, we transfer the intrinsic geometric information learned from the graph G_S encoding the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$ in steps A–D. We skip the steps to generate G_T for $\mathcal{T}_T$ in $\mathcal{D}_T$ as well as the steps to extract the structural characteristics therefrom. Under the condition that G_S and G_T bear structural similarities, we can directly build a model for $\mathcal{T}_T$ by (1) copying the convolutional and pooling layers that contain the features trained for $\mathcal{T}_S$ in $\mathcal{D}_S$, and by (2) training the fully connected layer of the model for fine tuning weights for $\mathcal{T}_T$ in $\mathcal{D}_T$.

This way of transfer learning provides efficiency in learning and also helps to minimize the problems resulting from lack of data and imperfect structural information for the new task. Note that the proposed method guarantees the spectral homogeneity of graphs by using the union of node sets on the heterogeneous source and target datasets. In our method, it is possible to utilize the spectral features of graphs from heterogeneous datasets.

Table 1: Details of the real-world datasets used

	AG	DBP	YELP	AMAZ
Train	$120,000$	$560,000$	$580,000$	$3,600,000$
Test	$7,600$	$70,000$	$38,000$	$400,000$
#class	4	14	2	2
Name	$\mathrm{sim}(G_1, G_2)^*$ [$\mathrm{corr}(wv_1, wv_2)^\dagger$]			
AG		0.37[0.45]	0.28[0.36]	0.35[0.42]
DBP	0.37[0.45]		0.23[0.29]	0.33[0.40]
YELP	0.28[0.36]	0.23[0.29]		0.50[0.58]
AMAZ	0.35[0.42]	0.33[0.40]	0.50[0.58]	

AG: a corpus of news articles on the web; DBP: ontology data from DBpedia; YELP: reviews from Yelp; AMAZ: reviews from Amazon.

* $\mathrm{sim}(G_1, G_2) = 0$ indicates that two graphs G_1 and G_2 are structurally complementary, whereas the value of 1 means that they are identical.

† $\mathrm{corr}(wv_1, wv_2)$ represents the correlation between the log-normalized bag of words extracted from each of the text corpora.

Results and Discussion

We tested the proposed method by performing topic classification of text documents. Text data carry information on not only individual words but also on their relationships, and graph-based methods are widely used for text mining. We utilized large-scale public NLP data (Zhang, Zhao, and LeCun 2015), which contained multiple corpora of news articles, Internet reviews, or ontology entries (Table 1). For controlled experiments, we also generated two pairs of synthetic datasets by random sampling of the real corpora. One pair consisted of two corpora with high similarity, and the other pair consisted of two corpora with low similarity.

For measuring the structural similarity between graphs, as shown in Table 1 and Fig. 3, we used the methods reported by existing studies (Koutra, Vogelstein, and Faloutsos 2013; Lee, Kim, and Yoon 2015); refer to the note below Table 1 for more details. Note that YELP and AMAZ bear the highest similarity in terms of the metrics used.

We implemented the deep networks with Torch and Cuda using AdaGrad as the optimizer and ReLU as the activation. We carried out 10-fold cross validation. Note that the proposed method can offer an efficient training scheme with relatively low computation cost of $O(n^{2.376})$ by leaving out the eigenvalue decomposition on the SCNN and re-using the model trained by the data in the source domain. In our experiments, the proposed method provided more than 10% reduction in the average training time.

Using the above setting, we first carried out comprehensive experiments to determine what factors affected the performance of the SCNN model for graph modeling. Table 2 lists part of the results we obtained by varying the net architecture, the method to generate graphs, and the type of Laplacian matrix along with the resulting classification accuracy for each combination. We can observe from Table 2 that the Laplacian methods do not significantly affect the performance, but L^{rwr} had the benefit in terms of computational complexity. SGE tended to give more accurate results than CoGE, which implies that the initial graph generation affected the model training more critically than structural

Table 2: Performance of SCNN model with various hyperparameters for text topic classification task

Model architecture*	Graph gener.	L type	Classification accuracy AG	DBP	YELP	AMAZ
GC8-FC500	CoGE	L^{basic}	0.89	0.95	0.91	0.88
GC8-FC500	CoGE	L^{rw}	0.89	0.95	0.91	0.88
GC8-FC500	CoGE	L^{rwr}	0.89	0.93	0.90	0.88
GC8-FC500	SGE	L^{basic}	0.91	0.97	0.91	0.89
GC8-FC500	SGE	L^{rw}	0.91	0.97	0.91	0.88
GC8-FC500	SGE	L^{rwr}	0.89	0.95	0.91	0.89
GC8-FC1K	CoGE	L^{basic}	0.90	0.95	0.93	0.88
GC8-FC1K	CoGE	L^{rw}	0.90	0.97	0.92	0.89
GC8-FC1K	CoGE	L^{rwr}	0.89	0.96	0.92	0.89
GC8-FC1K	SGE	L^{basic}	0.91	0.97	0.92	0.88
GC8-FC1K	SGE	L^{rw}	0.91	0.97	0.92	0.88
GC8-FC1K	SGE	L^{rwr}	0.89	0.96	0.91	0.88
GC8-GC8-FC1K	CoGE	L^{basic}	0.89	0.96	0.92	0.89
GC8-GC8-FC1K	CoGE	L^{rw}	0.89	0.97	0.92	0.89
GC8-GC8-FC1K	CoGE	L^{rwr}	0.89	0.97	0.92	0.88
GC8-GC8-FC1K	SGE	L^{basic}	0.91	0.97	0.92	0.88
GC8-GC8-FC1K	SGE	L^{rw}	0.91	0.97	0.92	0.88
GC8-GC8-FC1K	SGE	L^{rwr}	0.89	0.97	0.92	0.89

* For training, we set the kernel degree $K = 60$, learning rate to 0.01 and used cross-entropy cost function with AdaGrad optimizer. GC8 means the use of graph convolutional layers with 8 feature maps, and FC500/FC1K means the use of fully connected layer with 500/1000 hidden units.

feature extraction. For the experiments shown in Table 2, the GC8-GC8-FC1K model (refer to the note below Table 2 for notation) gave the best results, and we used this model as our main learning model in the following experiments.

We then performed experiments to determine the effectiveness of transfer learning using the synthetic datasets. The results are shown in Fig. 3; the plots in the top row are from the pair of synthetic corpora with high similarity [$\mathrm{sim}(G_1, G_2) = 0.75$ and $\mathrm{corr}(wv_1, wv_2) = 0.95$] for varying quantities of fine-tuning data for training the transferred model in the target domain (1%, 3%, 5%, and 10% of the entire target data). The plots in the bottom row of Fig. 3 correspond to the results from the pair of synthetic corpora with low similarity [$\mathrm{sim}(G_1, G_2) = 0.30$ and $\mathrm{corr}(wv_1, wv_2) = 0.50$]. We can observe that transfer learning is more effective for the higher similarity case, in which the test accuracy of the transferred model increased significantly faster than that of the source domain model. Using only 1% of the target domain data was sufficient for training, and using more data did not provide a noticeable difference. For the lower similarity case, the training in the target domain was limited and could not deliver the same level of accuracy in the source domain due to discrepancies in the underlying structure between the source and target domains.

Finally, we tested our approach with four corpora (AG, DBP, YELP, and AMAZ) as shown in Fig. 4. The plots in the top row represent the test accuracy of the model trained with the original data (solid line) and those of the transferred model trained with each of the other data (dotted line). The bottom plots represent the test loss. For the two corpora with the highest level of similarity (YELP and AMAZ), the effect

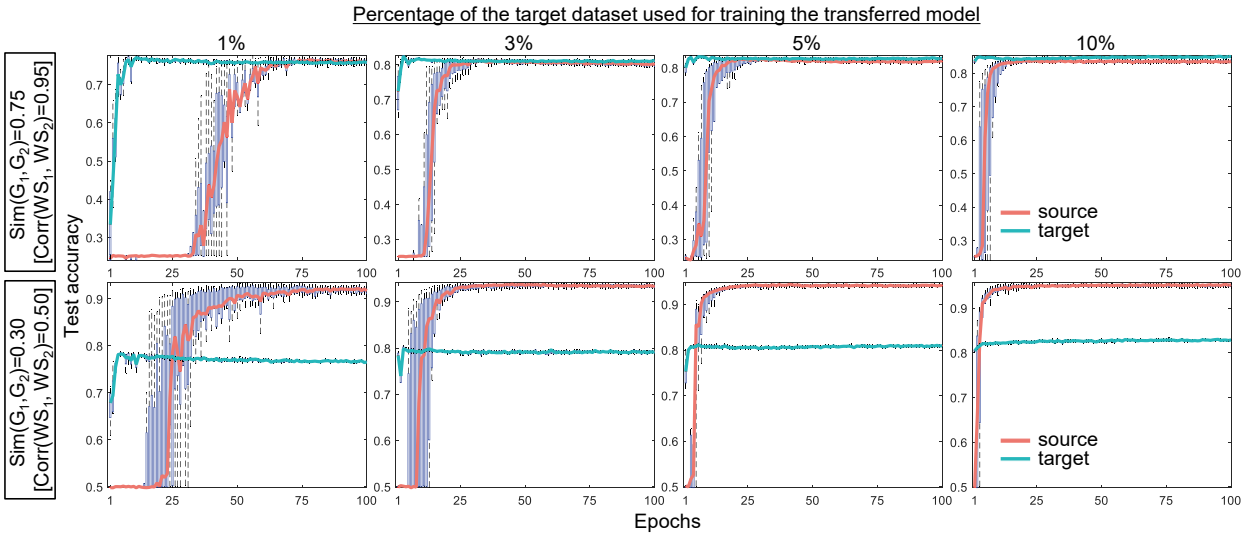

Figure 3: Results of intrinsic geometric information transfer learning for synthetic datasets (best viewed in color). Top: source and target datasets have high similarity in graph representations. Bottom: source and target datasets have low similarity. Each column: the percentage (1%, 3%, 5% and 10%) of the target dataset used for training the transferred model (fine tuning the fully connected layer). We repeated every experiment 10 times, and each data point shows a boxplot; red (source domain) and blue (target domain) lines connect the median locations of the boxplots.

of transfer learning was most salient. The test accuracy of the transferred model was comparable to that of the source model (for YELP) or was only 5–8% lower (for AMAZ). For the other cases with lower similarity than YELP and AMAZ, transfer learning was less effective. These results again confirmed our observation that the knowledge transfer is most successful when the source and target domains have high level of structural similarity between underlying graph representations.

Conclusion

We have proposed a new transfer learning framework for deep learning on graph-structured data. Our approach can transfer the intrinsic geometric information learned from the graph representation of the source domain to the target domain. We observed that the knowledge transfer between tasks domains is most effective when the source and target domains possess high similarity in their graph representations. We anticipate that adoption of our methodology will help extend the territory of deep learning to data in non-grid structure as well as to cases with limited quantity and quality of data. To prove this, we are planning to apply our approach to diverse datasets in different domains.

Acknowledgments

This work was supported by Ministry of Science, ICT & Future Planning [No.2016M3A7B4911115, No.PA-C000001], the Institute for Information Communications Technology Promotion [No.B0101-16-0307], the Brain Korea 21 Plus Project in 2016 grant funded by the Korea government, and Samsung Research Funding Center of Samsung Electronics [No.SRFC-IT1601-05].

References

Boscaini, D.; Masci, J.; Melzi, S.; Bronstein, M. M.; Castellani, U.; and Vandergheynst, P. 2015. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, 13–23. Wiley Online Library.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chung, F. R. 1997. *Spectral graph theory*, volume 92. American Mathematical Soc.

Deo, N. 2016. *Graph theory with applications to engineering and computer science*. Courier Dover Publications.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, 2215–2223.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Jain, A.; Zamir, A. R.; Savarese, S.; and Saxena, A. 2015. Structural-RNN: Deep learning on spatio-temporal graphs. *arXiv preprint arXiv:1511.05298*.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Koutra, D.; Vogelstein, J. T.; and Faloutsos, C. 2013. DeltaCon: A principled massive-graph similarity function. SIAM.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012.

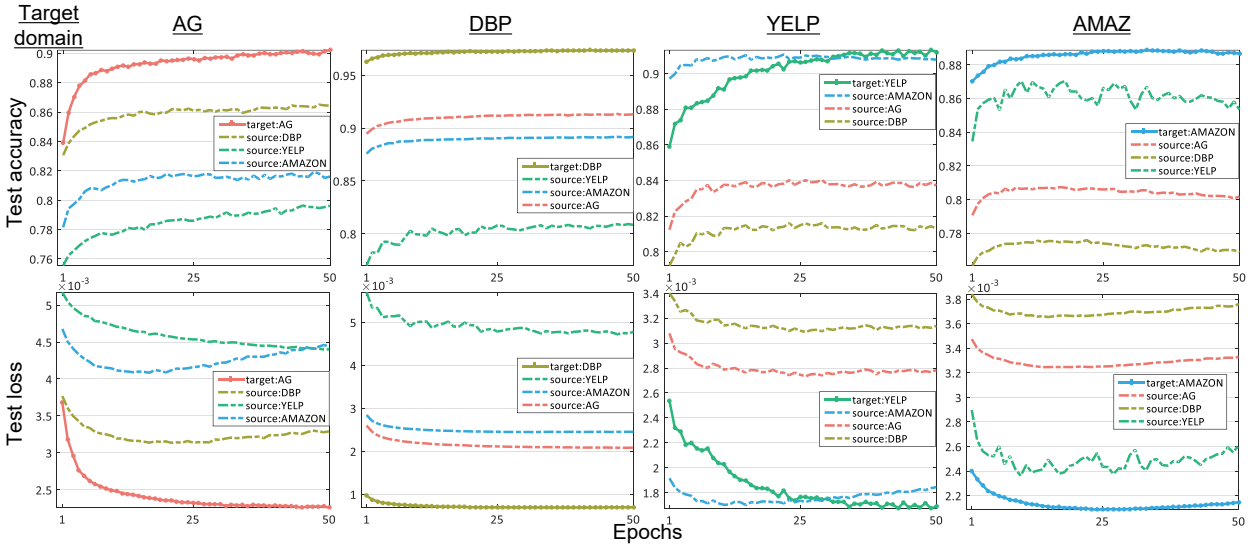

Figure 4: Results of the proposed method on real-world datasets (best viewed in color). Each plot in the top row: test accuracy of the model trained with the original data (solid lines) and those of the models trained with the other data sources and transferred (dotted lines). Each plot in the bottom row: test loss. For YELP and AMAZ, transfer learning was most effective, given that they have the highest level of structural similarity of all the cases (see Table 1).

Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Lee, J.; Kim, G.; and Yoon, S. 2015. Measuring large-scale dynamic graph similarity by RICom: RWR with intergraph compression. In *IEEE International Conference on Data Mining*, 829–834.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Mohar, B. 1997. *Some applications of Laplace eigenvalues of graphs*. Springer.

Mou, L.; Li, G.; Zhang, L.; Wang, T.; and Jin, Z. 2016. Convolutional neural networks over tree structures for programming language processing. In *AAAI Conference on Artificial Intelligence*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. *arXiv preprint arXiv:1605.05273*.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Rustamov, R., and Guibas, L. J. 2013. Wavelets on graphs via deep learning. In *Advances in Neural Information Processing Systems*, 998–1006.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30(3):83–98.

Shuman, D. I.; Ricaud, B.; and Vandergheynst, P. 2016. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis* 40(2):260–291.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sonawane, S., and Kulkarni, P. 2014. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications* 96(19).

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Tong, H.; Faloutsos, C.; and Pan, J.-Y. 2006. Fast random walk with restart and its applications. In *IEEE International Conference on Data Mining*, 613–622.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 649–657.