Augmenting Graph Convolution with Distance Preserving Embedding for Improved Learning

Guojing Cong, Seung-Hwan Lim, Steven Young
Oakridge National Laboratory
Oakridge, TN, 37830
Email: {congg,lims1, youngsr}@ornl.gov

***Abstract*—Graph convolution incorporates topological information of a graph into learning. Message passing corresponds to traversal of a local neighborhood in classical graph algorithms. We show that incorporating additional global structures, such as shortest paths, through distance preserving embedding can improve performance. Our approach, *Gavotte*, significantly improves the performance of a range of popular graph neural networks such as *GCN*, *GAT*,*GraphSAGE*, and *GCNII* for transductive learning. *Gavotte* also improves the performance of graph neural networks for full-supervised tasks, albeit to a smaller degree. As high-quality embeddings are generated by *Gavotte* as a by-product, we leverage clustering algorithms on these embeddings to augment the training set and introduce *Gavotte+*. Our results of *Gavotte+* on datasets with very few labels demonstrate the advantage of augmenting graph convolution with distance preserving embedding.**

1. Introduction

Graph convolution provides a mechanism to apply deep learning to problems that are naturally represented as graphs (e.g., see Hamilton *et al.* [2017]; Kipf and Welling [2017]; Veličković *et al.* [2017]). Graph convolutional network (*GCN*) has been employed for a wide range of machine learning tasks such as text classification Yao *et al.* [2019], traffic pattern prediction Chen *et al.* [2020a], material design Xie and Grossman [2018] and drug discovery Sun *et al.* [2019], where superior performance is achieved in comparison to prior approaches. *GCN*s have become more sophisticated with extensions including for example edge convolution Wang *et al.* [2019], neighborhood sampling Hamilton *et al.* [2017], and attention mechanism Veličković *et al.* [2017]; Yang *et al.* [2019].

*GCN*s achieve impressive results for semi-supervised learning (or transductive learning), where limited amount of data is labeled. Transductive learning reflects the practical difficulty of obtaining labels for data, and the amount of training samples in the dataset is far less than that in supervised learning. For example, in several popular citation network datasets, the number of training samples is below 6%. The success of *GCN*s in the transductive setting is due to their capability to simultaneously leverage input features and graph structures for learning.

Through message passing, the local neighborhood of a node contributes features to learning. The neighborhood expands as the network gets deeper. However, deeper *GCN*s have limitations in their expressiveness due to over-smoothing Li *et al.* [2019]. As a result, many high performing *GCN*s are shallow in practice. This suggests that in many *GCN*s global structures in a graph are not captured in learning. Some studies attempt to introduce constructs other than local neighborhoods to learning Yang *et al.* [2019]; You *et al.* [2019]. Unfortunately, either the network or training procedure becomes much more complicated. In addition, it is unclear if these extensions are applicable to recent deeper neural networks Chen *et al.* [2020b]; Rong *et al.* [2020a]; Xu *et al.* [2018] that address the over-smoothing issue.

Structures such as spanning trees, shortest paths, articulation points, min-cuts, and expanders lend unique properties to a graph. They can be helpful to learning, especially transductive learning. We explore incorporating one such global structure, that is, shortest paths, through distance preserving embedding to learning. Our approach, *Gavotte* (GrAph conVOlution wiTh disTance Embedding), learns an embedding that preserves the distance between a pair of nodes in the graph in the embedding space, and at the same time learns a function for node classification. Learning an embedding is unsupervised while classification is supervised. *Gavotte* unifies the two by employing one *GCN* for two tasks. It can be considered as a meta learning approach as it works with most existing *GCN* flavors such as vanilla *GCN*, *GAT*, *GraphSAGE*, and *GCNII*. The capability to capture local neighborhood information through graph convolution and global structure through embedding yields improved performance for vanilla *GCN*, *GAT*, *GraphSAGE*, and *GCNII* on our benchmark datasets.

The embedding generated by *Gavotte* enables further optimizations for transductive learning on datasets with very few labels. In *Gavotte+*, we augment the labeled set with nodes from unlabeled sets that are closest to the training samples in the embedding space. We then run *Gavotte* again with the expanded training set. Significant improvement is observed for datasets with very few labeled nodes.

Our main contributions are as follows.

- We incorporate global structures, for example, shortest paths, through distance preserving embedding in conjunction with *GCN*s for learning. *Gavotte* improves the performance of both semi-supervised and supervised learning, and motivates future *GCN*s to employ other elaborate constructs from classical graph algorithms.

- *Gavotte* as a meta learning approach works with a range of *GCN*s such as vanilla *GCN*, *GAT*, *GraphSAGE*, and *GCNII*. Minimal changes are made to these neural networks with significant performance improvement.
- *Gavotte+* further improves learning, and is especially effective for datasets with very few labeled samples. For example, *Gavotte+* achieves 82.0% test accuracy for Pubmed with 0.3% labeled data.

2. Preliminaries and Related work

Consider a graph $G = (V, E)$, where V is the set of vertices (or nodes) and E is the set of edges. $|V| = n$, and $|E| = m$. There is an optional feature vector associated with each $v \in V$, denoted as $\mathbf{v}$. $A = [a_{u,v}]$, $u, v \in V$, is the adjacency matrix, $\Delta = A - D$ is the Laplacian matrix, where D is the diagonal matrix with $D_{v,v} = \sum_{u \in V} a_{u,v}$. For our discussion, we use M to denote the similarity matrix and its entry $m_{u,v}$ measures the similarity between two vertices $u, v \in V$ or their associated vectors $\mathbf{u}$ and $\mathbf{v}$. $\|\mathbf{u}\|_2$ denotes the L_2 norm of $\mathbf{u}$, and $\cdot$ is dot product.

We next introduce some prior studies relevant to our work on graph embedding, semi-supervised learning, and graph convolution.

Graph Embedding

An embedding function $f : V \to \mathbb{R}^t$ maps a node $v \in V$ or its associated vector $\mathbf{v}$ to a t-dimensional vector $f(v)$ or $f(\mathbf{v}) \in \mathbb{R}^t$. Note we use t instead of d for the number of dimensions as d is reserved for distances in graph and embedding space. With the similarity matrix $M = [m_{u,v}]$, a natural objective function for graph embedding is given in Yan *et al.* [2007]

$$\min_f \sum_{u \neq v} (\|f(u) - f(v)\|_2 m_{u,v}) \tag{2.1}$$

As in practice it is difficult to compute M, approximations are used (see, e.g., Cai *et al.* [2018]; Goyal and Ferrara [2018]; Xu [2020]).

DeepWalk Grover and Leskovec [2016] and *Node2Vec* Perozzi *et al.* [2014] are two of the earliest graph embedding approaches that employ deep learning. In *DeepWalk* and *Node2Vec*, graph embedding is done by random walk together with SkipGram modeling Mikolov *et al.* [2013]. Take *Node2Vec* for example. For every source node $v \in V$, its neighborhood $N_S(v) \subset V$ is generated through a neighborhood sampling strategy S with random walks. The objective is formulated as

$$\max_f \sum_{v \in V} \log Pr(N_S(v)|f(v))$$

where Pr is the conditional probability.

GraphSAGE generates embeddings for nodes in a graph by encouraging nearby nodes (e.g., those touched by a random walk) to have similar embeddings or representations and disparate nodes to have highly distinct representations Hamilton *et al.* [2017]. The objective function for *GraphSAGE* is

$$-\log(\sigma(\mathbf{z}_u \cdot \mathbf{z}_v)) - \mathcal{Q}\mathbb{E}_{v_n \approx P_n(v)} \log(\sigma(-\mathbf{z}_u \cdot \mathbf{z}_{v_n}))$$

Here σ is the sigmoid function; P_n is a negative sampling distribution; $\mathbf{z}_u$ and $\mathbf{z}_v$ are the projections for u and v, respectively; and $\mathcal{Q}$ defines the number of negative samples.

Semi-supervised Learning

In semi-supervised learning with graphs, the nodes are partitioned into labeled and unlabeled sets L and U, respectively. $L \cup U = V$. Generally, the objective function of semi-supervised learning has the following form

$$\sum_{v \in L} \mathcal{L}(y_v, f(\mathbf{v})) + \lambda \sum_{u,v \in V} m_{u,v} \|f(\mathbf{u}) - f(\mathbf{v})\|_2 \tag{2.2}$$

The first term in Formula 2.2 is the supervised loss with y_v as the label for v. The second term imposes similarity regularization, and yields a large penalty when similar nodes u and v with a large $m_{u,v}$ are predicted to have different labels $f(\mathbf{u})$ and $f(\mathbf{v})$. λ is a constant weighting factor.

In approximating M, most prior methods assume that nodes close to each other in the graph tend to have the same labels (e.g., see Weston *et al.* [2008]; Yang *et al.* [2016]; Zhu *et al.* [2003]). This is known as *clustering assumption*. Replacing M with Laplacian Δ for example gives the following objective

$$\sum_{v \in L} \mathcal{L}(y_v, f(\mathbf{v})) + \lambda \mathbf{f}^T \Delta \mathbf{f}$$

Label propagation (*LP*) by Zhu *et al.* [2003] minimizes the similarity regularization term with

$$m_{u,v} = exp\left(-\sum_{i=1}^{t} \frac{(\mathbf{u}_i - \mathbf{v}_i)^2}{\rho_i^2}\right)$$

where $\rho_1, \cdots, \rho_t$ are length scale hyperparameters for each dimension.

Using a similar M, Zhou *et al.* [2004] regularize supervised learning by minimizing

$$\sum_{u,v \in V} m_{u,v} \left\| \frac{\mathbf{u}}{\sqrt{D_{u,u}}} - \frac{\mathbf{v}}{\sqrt{D_{v,v}}} \right\|_2$$

Yang *et al.* [2016] employ context based embedding such as *DeepWalk* in their semi-supervised learning approach, and minimize an unsupervised loss of predicting the graph context.

GCNs

The vanilla *GCN* is a localized first-order approximation of spectral graph convolution Kipf and Welling [2017].

$$h_v^{l+1} = \sigma\left(b^l + \sum_{u \in N(v)} C_{u,v} h_u^l W^l\right)$$

Here h_v^l is the hidden feature at node v at convolution level l. For each node v, its hidden feature at next level h_v^{l+1} is computed by aggregating hidden states of its neighbors $u \in N(v)$ and possibly of the incident edges as well. $C_{u,v}$ is a constant, b is the bias, and W is the weight to be learned.

Graph attention network (*GAT*) augments regular graph convolution with a multi-head attention mechanism that improves

performance Veličković *et al.* [2017]. In *GAT*, the multi-head attention mechanism allows it to attend to all its neighbors when the hidden feature of v is being computed. Formally, h_v is computed as

$$h_v^{l+1} = \sum_{u \in N(v)} \alpha_{u,v}^l W^l h_u^l$$

where $\alpha_{u,v}^l = softmax_u(e_{u,v}^l)$, W is the weight to be learned, and $e_{u,v}$ is the edge between nodes u and v. *GAT* computes h_v^l and eventually the prediction with the input of attention on the edges $e_{u,v}^l$, $u \in N(v)$.

Initial residual connection and identity mapping are introduced in in *GCNII* Chen *et al.* [2020b] to overcome over-smoothing and leverage deep neural networks. They extend *GCN* to express a high order polynomial filter with arbitrary coefficients.

3. *Gavotte*– Graph Convolution with Distance Embedding

Gavotte aims to leverage structures in a graph beyond the local neighborhood of a node for learning. Plain message passing alone is unlikely to capture global structures. First, due to over-smoothing, most *GCN*s are shallow, and the neighborhood that contributes to the learning of a node's feature is oftentimes small. Even in the absence of over-smoothing, for a node $v \in V$, the contribution from a node u that is outside of v's immediate neighborhood gets diluted during message passing. Some *GCN*s attempt to explicitly utilize constructs beyond local neighborhood. In these studies, the network architecture or the training procedure becomes much more complicated Yang *et al.* [2019]; You *et al.* [2019], reflecting the challenge of marrying classical graph algorithms with graphical neural networks.

Gavotte exploits shortest paths in the graph in addition to local neighborhoods through distance preserving embedding, and for a node it produces an embedding as well as a prediction of its property (e.g., in the case of classification, a predicted label). Of the many structures in graphs such as cycles, paths, minors, covers, expanders and so on, we choose shortest paths as they form a natural metric space satisfying the properties such as symmetry, $d(u,v) = d(v,u)$, and triangle inequality, $d(u,v) + d(v,x) \geq d(u,x)$. The embedding minimizes the unsupervised loss

$$\mathcal{L}_{embedding} = \sum_{u,v \in V} (d(u,v) - \|f(\mathbf{u}) - f(\mathbf{v})\|_2)^2 \qquad (3.1)$$

for all pairs of $u, v \in V$, where $d(u,v)$ is the (shortest path) distance between u and v.

The resulting embedding does not depend on the similarity matrix M or its approximation derived with *clustering assumption*. The distance preserving embedding generated with Equation 3.1 in theory can retain all topology information of the original graph. To see this, let us assume the embedding is isometric, and show the original graph can be reconstructed from its embedding. Let $f(V)$ be the embedding function for V. Obviously $|f(V)| = |V|$. For each $u' \in f(V)$, we create a node u. We create all edges incident to u to reconstruct the graph. For that, we find N_u, that is, all neighbors of u. As the embedding is isometric, we simply let $N_u = \{v | d(v', u') = 1\}$, $v', u' \in f(V)$, and v is node created for v'. Here we slightly abuse the notation d to denote the distance between two vectors in the embedding space. Thus the structure of the original graph is fully present in the embedding. In practice there of course may be distortions in the embedding. However, our experiments show that enough structural information is preserved for our learning purpose.

Gavotte employs a *GCN* to optimize $\mathcal{L}_{embedding}$ and another *GCN* to optimize a regular training loss $\mathcal{L}_{training}$ (e.g., cross-entropy loss) on the labeled set $L \subset V$. Equation 3.2 computes the features for $v \in V$ with *Gavotte* when the *GCN* is the vanilla *GCN*.

$$h_v^l = \begin{cases} \mathbf{v} & l = 0 \\ \sigma\left(b^{l-1} + \sum\limits_{u \in N(v)} C_{u,v} h_u^{l-1} W^{l-1}\right) & 1 \leq l \leq H \\ b_o^H + \sum\limits_{u \in N(v)} C_{u,v} h_u^H W_o^H & v \in L, l = H+1 \\ b_e^H + \sum\limits_{u \in N(v)} C_{u,v} h_u^H W_e^H & v \in V, l = H+1 \end{cases} \qquad (3.2)$$

In the neural network represented by Equation 3.2, there are H hidden layers. The biases b, b_o, and b_e are for hidden layers, the output layer, and embedding output (for computing distance-preserving loss), respectively. The weight matrices are similarly subscripted. After the H^{th} layer, it computes an output feature for each $v \in L$ and an embedding feature for each $v \in V$. Note $v \in L$ gets both the output feature and the embedding feature. The architecture is demonstrated in Figure 1.

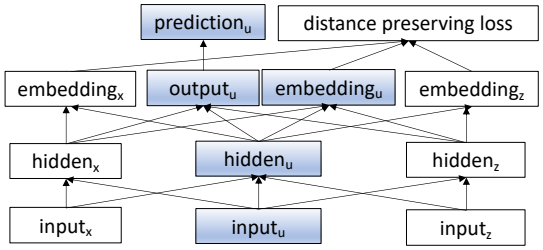

Fig. 1: An example graph of three nodes. Node u has two neighbors x and z. The hidden feature for u is computed with the input features of u, x, and z, and the output and embedding for u are computed with the hidden features of u, x, and z

In Figure 1, the supervised and unsupervised learning are unified through graph convolution. The figure shows the learning architecture of *Gavotte* for a graph with $V = \{u, x, z\}$, $E = \{(u,x), (u,z)\}$, $L = \{u\}$. Data and operations associated with the labeled node are shaded. *Gavotte* computes a hidden feature h using graph convolution for each node. From the same h, *Gavotte* computes a predicted output for evaluating $\mathcal{L}_{training}$ and an embedding for evaluating

$\mathcal{L}_{embedding}$. Combining them, the objective is to minimize $\mathcal{L}_{training} + \lambda\mathcal{L}_{embedding}$.

Yang *et al.* in their semi-supervised learning framework, *Planetoid*, use context prediction for embedding Yang *et al.* [2016]. As *Planetoid* employs SkipGram for embedding without using feature vectors, it has two separate paths of information flow for learning. *Planetoid* needs an additional algorithm to coordinate the learning from the two paths.

Many *GCN*s have been proposed in the literature, for example, vanilla *GCN*, *GraphSAGE*, *GAT*, and *GCNII*, and to the best of our knowledge, *Gavotte* works with all of them. *Gavotte* may be viewed as a meta method to boost the performance of *GCN*s. In *Gavotte*, unsupervised learning (i.e., distance preserving embedding) shares much of the same neural network structure with the supervised learning. The quality of the embedding directly impacts the prediction performance of *Gavotte*. We evaluate the produced embedding by *Gavotte* in Section 4.

To compute $\mathcal{L}_{embedding}$ in Equation 3.1, we need to compute all pair shortest paths (APSP). APSP is notoriously time consuming. As is common in prior studies, we devise a simple sampling scheme to reduce the computational complexity. In each training step, we randomly sample $s \ll n$ nodes as representatives, and compute the distance from all nodes in V to these s representative nodes.

$$\mathcal{L}_{embedding} = \sum_{\substack{u\in V \\ v_i \approx P(V), i=1,\cdots,s}} (d(u,v_i) - \|f(\mathbf{u}) - f(\mathbf{v}_i)\|_2)^2 \tag{3.3}$$

Here $P(V)$ is a uniform distribution on V. The number of terms in $\mathcal{L}_{embedding}$ is reduced from $\Theta(n^2)$ to $\Theta(sn)$.

Computing $d(u,v)$ is straightforward with breadth-first search (BFS) if G is connected. For disconnected graph, $d(u,v)$ is usually defined to be ∞ and is problematic for learning. To facilitate a well-behaved embedding, in *Gavotte*, we set $d(u,v)$ to be one larger than the largest diameter of the connected component in G. Instead of complex algorithms Chechik *et al.* [2014] and heuristics Crescenzi *et al.* [2013], we approximate the diameter with a simple and fast 2-approximation algorithm, where we randomly select a node in the graph (e.g., the hub), and compute a BFS tree from that node. The diameter of the graph is no larger than twice the depth of the BFS tree.

4. *Gavotte* PERFORMANCE

We first consider transductive learning with three citation network datasets Sen *et al.* [2008]: Cora, Citeseer, and Pubmed. In these datasets, nodes correspond to documents, edges correspond to citation links, and each node has a sparse bag-of-words feature vector as well as a class label. The characteristics of these datasets are summarized in Table I. In this setting, only a small fraction of nodes, that is, 20 per class, are included in the training/labeled set. Pubmed is the largest network with the smallest percentage of labeled samples. Pubmed is connected, while Cora and Citeseer have 78 and 438 connected components, respectively. In both Cora and Citeseer, there is one big component and many tiny components (e.g., singletons or structures with two or three nodes). Pubmed is representative of emerging large datasets with few labels. Pubmed is the most challenging instance for *GCN* extensions to improve upon vanilla *GCN*. It is shown in Chen *et al.* [2020b] that minor improvement or even degradation is observed for Pubmed with some of the state-of-art *GCN*s such as *GAT*, *APPNP* Rozemberczki *et al.* [2021], *JKNet* Xu *et al.* [2018], and *Incep* Rong *et al.* [2020a].

	Cora	Citeseer	Pubmed
# nodes	2,708	3,327	19,717
# edges	5,278	4,552	44,324
# features	1,433	3,703	500
# classes	7	6	3
# nodes in training	140	120	60
# nodes in test	1,000	1,000	1,000

TABLE I: Characteristics of the three citation networks

We implement *Gavotte* in Pytorch version 1.8, and experiment with various graph convolutions including vanilla *GCN*, *GAT*, *GraphSAGE*, and *GCNII*. For *GCN*, *GAT* and *GraphSAGE*, we use implementations from Deep Graph Library (DGL) version 0.6.1 Rong *et al.* [2020b]. For *GCNII*, we use the author's own implementation and hyperparameter settings Chen *et al.* [2020b]. The experiments are run on a NVIDIA DGX-2 with V100 GPUs.

Embedding Quality

As embedding impacts prediction performance, we examine the quality of the embedding *Gavotte* produces. For that, we run *Gavotte* with only unsupervised learning to produce an embedding. Specifically, we minimize $\mathcal{L}_{embedding}$ in equation 3.3, and then feed the embedding vectors to a downstream node classification task.

For illustration purposes, we embed Zachary's Karate Club Zachary [1977] graph, and we set $t = 2$ for easy visualization. The embedding result is shown in Figure 2. The nodes are painted in two colors representing the two communities that are formed (each node belongs to one community). That is, the nodes are classified into two categories, and the training set contains two nodes, node 0 and node 32 with colors purple and green, respectively. The embedding clearly separates the two communities except for node 8, as shown by the dotted line in the figure.

Recall that this graph embodies the relationship (interactions outside the club) among 34 members of a karate club. After a conflict between the administrator (node 0) and the instructor (node 32), the club is split into two. Half of the members formed a new club around the instructor, and the rest formed a new club with a new instructor or gave up karate. Zachary Zachary [1977] applies a max-flow, min-cut algorithm with node 0 as source and node 32 as sink on the graph, and correctly assigns all but one member of the club to the new groups. That group member is node 8[1]. The correlation with

[1]In Zachary's study, the nodes are numbered from 1 to 34. Here our numbering starts from 0

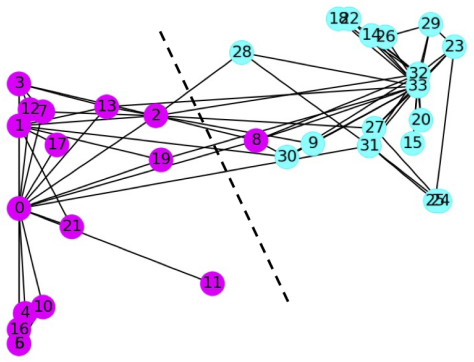

Fig. 2: Embedding of Zachary's Karate Club dataset produced by *Gavotte*

Zachary's result suggests a connection between learning on graphs and classical graph algorithms.

We run *Gavotte* for 400 epochs to generate an embedding. To emphasize the quality of the embedding and the contribution to learning from graph topology, we exclude node features from this experiment so no hints from the bag-of-words vectors are taken during embedding. Each node is encoded as a one hot vector. We set $t = 128$ and $s = 13$. We use the Adam optimizer with a weight decay of $5e{-}4$. After the embedding is completed, for the downstream task, the input data is $n = |V|$ 128-dimensional vectors, with $|L|$ of them in the training set. Test is done on the embeddings produced for the test set. Each test set contains 1000 vectors. Note the downstream task no longer has access to the original graph. Learning for the downstream task uses a simple fully connected neural network with one hidden layer of 256 units. ReLU is used as the activation function. Training for the downstream task takes 100 epochs with the Adam optimizer with weight decay $5e - 4$. The classification results are shown in Table II.

	Cora	Citeseer	Pubmed
Planetoid	0.691	**0.493**	0.664
GCN	0.592 ± .009	0.372 ± .01	0.550 ± .008
MLP	**0.701 ± .005**	0.482 ± .006	**0.690 ± .005**

TABLE II: Performance comparison of various learning methods on plain graphs without node features.

In Table II, the numbers for *Planetoid* are reproduced from Yang *et al.* [2016]; *GCN* is with the vanilla *GCN*; and *MLP* is the downstream task with the fully connected network operating on the embeddings generated by *Gavotte*. *GCN* and *MLP* numbers are collected with 10 runs. It is striking that *MLP* outperforms *GCN* by a significant margin even though it does not has access to the input graph. This suggests that the graph topology is well preserved in the embedding generated by *Gavotte*, and vanilla graph convolution does not fully utilize the structural information in the graph.

Transductive Learning Results

We now present the transductive learning results with *Gavotte* using node features. We experiment with *GCN*, *GAT*, *GraphSAGE* and and *GCNII*. Unless noted otherwise, the hyperparameters, that is, number of layers, hidden layer size, learning rate, and weight decay, are set as those proposed by the authors in their original studies. *GraphSAGE* uses the "gcn" aggregator function. In all experiments $t = 128$, $s = 13$. The most important hyperparameter to tune for *Gavotte* is λ. Recall that λ balances the weighting of the supervised and unsupervised losses during training. If $\lambda = 0$, *Gavotte* becomes standalone training with *GCN*s. $\mathcal{L}_{embedding}$ can be viewed as a regularization for the supervised learning on labeled data. Enforcing distance based embedding regularizes by enforcing the solution to "heed" the global graph structures (here the shortest paths). We find the best λ by a simple search in the range $[0.001, 1]$. Note that we explicitly exclude 0 in the search.

The results with the three citation network datasets are shown in Table III. The numbers are collected with 10 runs for each implementation. In Table III, the performance numbers are in general agreement with the numbers reported by the each method's authors except for *GAT*. In our runs, *GAT* improves the performance over *GCN* for Cora, but not as much on Citeseer and Pubmed. There can be many subtle causes to this behavior. Since the performance of *GAT* is surpassed by *GCNII*, we do not further investigate this discrepancy.

	Cora	Citeseer	Pubmed
GCN	0.815 ± .004	0.711 ± .006	0.790 ± .003
w. *Gavotte*	**0.830 ± .004**	**0.720 ± .005**	**0.793 ± .003**
GAT	0.826 ± .007	0.709 ± .006	0.785 ± .004
w. *Gavotte*	**0.834 ± .006**	**0.718 ± .006**	**0.792 ± .005**
GraphSAGE	0.816 ± .005	0.710 ± .007	0.780 ± .006
w. *Gavotte*	**0.834 ± .004**	**0.718 ± .005**	**0.792 ± .006**
GCNII	0.855 ± .003	0.721 ± .005	0.802 ± .007
w. *Gavotte*	**0.862 ± .004**	**0.734 ± .006**	**0.811 ± .007**

TABLE III: Test accuracy on citation networks

Gavotte improves the test accuracy across the board for *GCN*, *GAT*, *GraphSAGE*, and *GCNII*. For example, it improves over *GCN* for Cora, Citeseer, and Pubmed by 1.5, 0.8, and 0.3 percentage points, respectively. Moreover, the performance of *Gavotte* with *GCN* is better than standalone *GAT* and *GraphSAGE*. This makes *Gavotte* with vanilla *GCN* a strong candidate for transductive learning in comparison with some of the most popular implementations. In this case *Gavotte* has the advantage of a very simple neural network architecture, and it generates high quality embedding as a side product.

GCNII is different from vanilla *GCN*, *GAT*, and *GraphSAGE* as it explicitly addresses over-smoothing and is able to leverage deep networks (e.g., up to 64 layers). Supposedly *GCNII* is able to capture features from a large neighborhood of a node. Indeed *GCNII* outperforms *GCN* and *GAT*. The fact that *Gavotte* with *GCNII* improves upon standalone *GCNII*

	Chameleon	Cornell	Texas	Wisconsin
# nodes	2,277	183	183	251
# edges	36,101	295	309	499
# features	2,325	1,703	1,703	1,703
# classes	4	5	5	5

TABLE IV: Characteristics of four additional networks

shows that it is still beneficial to explicitly incorporate global structures into learning even when the network is deep.

Of the three datasets, Cora benefits the most from increasingly sophisticated architectures. Classification accuracy with Cora increases by about 5 percentage points from *GCN* to *GCNII*. Much more modest improvement, about 1.2 percentage points is observed for Pubmed. *Gavotte* with *GCNII* brings the accuracy to over 81%.

Full-supervised Learning Results

Gavotte works for full-supervised learning. In addition to citation networks, for full-supervised learning we include four more datasets, Chameleon, Cornell, Texas, and Wisconsin, as is done in some prior studies (e.g., see Chen *et al.* [2020b]). These datasets are web networks with nodes and edges representing web pages and hyperlinks, respectively. Similar to the citation network datasets, the feature associated with each node is the bag-of-words representation of the corresponding web page. Table IV summarizes the characteristics of these datasets.

Table V shows the results of full-supervised learning with seven datasets: Cora, Citeseer, Pubmed, Chameleon, Cornell, Texas, and Wisconsin. For each dataset, we randomly split nodes of each class into 60%, 20%, and 20% for training, validation and test, respectively, and measure the performance on the test sets over 10 random splits. *Gavotte* shows the biggest improvement over *GCNII* for Wisconsin. The overall improvement is more modest in comparison to the transductive learning setup. This is expected as more training data becomes available, the influence of graph structure wanes over increased amount of node features.

5. *Gavotte+* FOR DATASET WITH VERY SMALL L

GAT, *GraphSAGE*, and *GCNII* extend vanilla *GCN* with improved learning performance. Yet the improvement is not uniform across all datasets, as shown in Table III. For all three *GCN*s, without the *Gavotte* augmentation, the biggest improvement is observed with Cora. Smaller improvement is observed with Citeseer and Pubmed. The improvement appears to correlate with the amount (percentage) of labeled samples in the dataset. Take the best performing *GCN* of the three, *GCNII*, for example. About 4%, 1%, and 0.8% improvement over vanilla *GCN* is achieved for Cora, Citeseer, and Pubmed, respectively. The corresponding percentages of training samples are 5%, 3%, and 0.3%, respectively. The Pubmed dataset represents the quintessential challenge of training with very few labeled samples.

We propose *Gavotte+*, that leverages the embedding generated by *Gavotte* for datasets with a small L. The main mechanism of *Gavotte+* is to "expand" the very limited training set by adding samples from the original unlabeled set. Such a mechanism is made possible by the high-quality embeddings that *Gavotte* generates as a by-product for prediction tasks. We cluster the embeddings generated for the nodes and add nodes closest to the center of each category to the training set. We give a brief sketch of *Gavotte+* as follows.

For a graph $G = (V, E)$ where $V = L \cup U$, we first run *Gavotte* with a certain *GCN* flavor, say *GCNII*, using the labeled set L until stopping (e.g., patience is reached). *Gavotte* generates an embedding vector for all $v \in V$. For each label category $i \in \{1, 2, \cdots, c\}$, we compute a center $\mathcal{C}_i$ for all embedding vectors of labeled samples in category i. The center can be computed by any number of clustering algorithms. Let Q_i be the set of q nodes in U with embeddings closest to $\mathcal{C}_i$, and let $Q = \cup_{i=1}^{c} Q_i$. Obviously, $|Q| = cq$.

So far we have identified cq nodes in the unlabeled dataset. In order to add them to L, we need to provide labels to them. We give a node $p \in Q$ label i if the embedding of p is closest to $\mathcal{C}_i$, $i \in \{1, 2, \cdots, c\}$, with ties broken arbitrarily. Training resumes with new $L = L \cup Q$ and $U = U - Q$.

We experiment with *Gavotte+* with both *GCN* and *GCNII* on the citation network datasets. In our experiments, we add $q = 10$ new nodes per class to the training set except for *GCNII* with Pubmed. For Pubmed, we add 40 nodes per class. The results are shown in Table VI.

First note that the labels we predicted for these nodes may not always be correct when we add them to the training (labeled) set. In the table, *mis* shows the average number of mislabeled nodes in Q for 10 runs. Apparently there are more mislabeled nodes in Q for Citeseer. The labeling is quite accurate for Cora and Pubmed. Even in the case of mislabeled nodes that might pollute training, *Gavotte+* still improves the accuracy of both *GCN* and *GCNII* for all datasets. The improvement is bigger than previously achieved with *Gavotte*. The improvement is significant with *GCN* for Cora and with *GCNII* for Pubmed. *Gavotte+* with *GCNII* achieves 0.820 test accuracy with Pubmed. This is extraordinary considering how *GAT*, *GraphSAGE*, and *GCNII* barely improve over vanilla *GCN* for Pubmed.

6. CONCLUSION AND FUTURE WORK

We present *Gavotte*, an approach that leverages global structures, i.e., shortest paths, for learning by unifying graph convolution with distance preserving embedding. *Gavotte* successfully embeds topological information for learning as evidenced by the superior performance of the downstream node classification task using only the embeddings in comparison to vanilla *GCN* with full access to the graph. As a meta method, *Gavotte* improves the accuracies of *GAT*, *GraphSAGE*, *GCNII*, and vanilla *GCN* for a wide range of datasets for transductive learning. *Gavotte* improves the performance of *GCN*s for full-supervised learning as well. We expect *Gavotte* will work with many *GCN*s in addition to the ones used in our study. In comparison to other *GCN*s that attempt to incorporate structures other than local neighborhoods into learning, *Gavotte* does

Method	Cora	Citeseer	Pubmed	Chameleon	Cornell	Texas	Wisconsin
GCNII	0.885 ± .003	0.771 ± .005	0.896 ± .004	**0.606 ± .003**	0.749 ± .007	0.695 ± .006	0.741 ± .005
w. *Gavotte*	**0.885 ± .003**	**0.774 ± .004**	**0.898 ± .005**	0.602 ± .003	**0.754 ± .006**	**0.697 ± .004**	**0.753 ± .005**

TABLE V: Test accuracy for full-supervised learning on all datasets

		Cora	Citeseer	Pubmed
GCN	acc	0.815 ± .004	0.711 ± .006	0.790 ± .003
w. *Gavotte+*	acc	**0.835 ± .004**	**0.724 ± .005**	**0.794 ± .005**
	mis	1	8.6	1.1
	cq	70	60	30
GCNII	acc	0.855 ± .003	0.734 ± .005	0.802 ± .007
w. *Gavotte+*	acc	**0.863 ± .005**	**0.741 ± .005**	**0.820 ± .007**
	mis	0	10	7
	cq	70	60	120

TABLE VI: *Gavotte+* performance. Of *cq* nodes added to the training set, the average number of mislabeled ones is shown by *mis*.

not introduce complicated structures or training routines into learning.

The high quality embedding generated by *Gavotte* is further leveraged in *Gavotte+* for learning with few labeled samples. Pubmed has been challenging for many *GCN*s including *GAT* and *GraphSAGE*. *Gavotte+* improves upon the current best implementation, *GCNII*, on the citation network dataset. For Pubmed, it achieves an accuracy of 82.0%.

As there are numerous structures from classical graph theory that determine the properties of the graph, our work motivates future research of incorporating them to learning. Related is the question of what structural information current *GCN*s capture, especially the recent ones that adopt deep convolution.

7. Ackowledgement

This material is based upon work supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725, and in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC.

This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

H. Cai, V. W. Zheng, and K. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge & Data Engineering*, 30(09):1616–1637, Sep 2018.

S. Chechik, D.H. Larkin, L. Roditty, and et al. Better approximation algorithms for the graph diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, page 1041–1052, USA, 2014. Society for Industrial and Applied Mathematics.

F. Chen, Z. Chen, S. Biswas, S. Lei, and et al. Graph convolutional networks with kalman filtering for traffic prediction. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '20, page 135–138, New York, NY, USA, 2020. Association for Computing Machinery.

M. Chen, Z. Wei, Z. Huang, and et al. Simple and deep graph convolutional networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1725–1735. PMLR, 13–18 Jul 2020.

P. Crescenzi, R. Grossi, M. Habib, and et al. On computing the diameter of real-world undirected graphs. *Theoretical Computer Science*, 514:84–95, 2013. Graph Algorithms and Applications: in Honor of Professor Giorgio Ausiello.

P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

W.L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

T.N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

G. Li, M. Muller, A. Thabet, and B. Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9266–9275, Los Alamitos, CA, USA, Nov 2019. IEEE Computer Society.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

Y. Rong, W. Huang, T. Xu, and J. Huang. DropEdge: Towards

deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Y. Rong, T. Xu, J. Huang, and et al. Deep graph learning: Foundations, advances and applications. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 3555–3556, New York, NY, USA, 2020.

B. Rozemberczki, C. Allen, and R. Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2), 05 2021. cnab014.

P. Sen, G. Namata, M. Bilgic, and et al. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

M. Sun, S. Zhao, C. Gilvary, and et al. Graph convolutional networks for computational drug development and discovery. *Briefings in Bioinformatics*, 21(3):919–935, Jun 2019.

P. Veličković, G. Cucurull, A. Casanova, and et al. Graph attention networks. *6th International Conference on Learning Representations*, 2017.

Y. Wang, Y. Sun, Z. Liu, and et al. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1168–1175, New York, NY, USA, 2008. Association for Computing Machinery.

T. Xie and J.C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120:145301, Apr 2018.

K. Xu, C. Li, Y. Tian, and et al. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR, 10–15 Jul 2018.

M. Xu. Understanding graph embedding methods and their applications. *CoRR*, abs/2012.08019, 2020.

S. Yan, Dong Xu, B. Zhang, and et al. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:40–51, 2007.

Z. Yang, W.W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 40–48. JMLR.org, 2016.

Y. Yang, X. Wang, M. Song, and et al. SPAGAN:shortest path graph attention network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, Jul 2019.

L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *AAAI*, 2019.

J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 7134–7143. PMLR, 2019.

W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

D. Zhou, O. Bousquet, T. Lal, and et al. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 912–919. AAAI Press, 2003.