GeniePath: Graph Neural Networks with Adaptive Receptive Paths

Ziqi Liu
Ant Financial Services Group
Hangzhou, China
ziqilau@gmail.com

Chaochao Chen
Ant Financial Services Group
Hangzhou, China
chaochao.ccc@antfin.com

Longfei Li
Ant Financial Services Group
Hangzhou, China
longyao.llf@antfin.com

Jun Zhou
Ant Financial Services Group
Beijing, China

Xiaolong Li
Ant Financial Services Group
Seattle, Washington

Le Song
Georgia Institute of Technology
Ant Financial Services Group
Atlanta, GA

ABSTRACT

We present, GeniePath, a scalable approach for learning adaptive receptive fields of neural networks defined on graph data. In GeniePath, we propose an adaptive path layer that consists of two complementary functions designed for breadth and depth exploration respectively, where the former learns the importance of different sized neighborhoods, while the latter extracts and filters signals aggregated from neighbors of different hops away. Extensive experiments on node classification tasks compared with state-of-the-art methods show that our approaches are competitive on both transductive and inductive settings.

KEYWORDS

Graph Neural Networks, Adaptive Receptive Fields

ACM Reference Format:
Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, and Le Song. 2018. GeniePath: Graph Neural Networks with Adaptive Receptive Paths. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In this paper, we study the representation learning task involving data that lie in an irregular domain, i.e. graphs. Many data have the form of a graph, e.g. social networks [17], citation networks [18], biological networks [21], and transaction networks [16]. Convolutional Neural Networks (CNN) have been proven successful in a diverse range of applications involving images [11] and sequences [6]. Recently, interests and efforts have emerged in the literature trying to generalize convolutions to graphs [4, 8, 10, 15], which also brings in new challenges.

Unlike image and sequence data which lie in regular domain, graph data are irregular in nature, making the receptive field of each neuron different for different nodes in the graph. Assuming an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $i \in \mathcal{V}$, $|\mathcal{E}|$ edges $(i, j) \in \mathcal{E}$, the sparse adjacency matrix $A \in \mathbb{R}^{N\times N}$, diagonal node degree matrix D ($D_{ii} = \sum_{ij} A_{ij}$), and a matrix of node features $X \in \mathbb{R}^{N,P}$. Considering the following calculations in typical graph convolutional networks [9, 15]: $H^{(t+1)} = \sigma\left(\phi(A)H^{(t)}W^{(t)}\right)$ at the t-th layer parameterized by $W^{(t)} \in \mathbb{R}^{K\times K}$, where $H^{(t)} \in \mathbb{R}^{N\times K}$ denotes the embeddings of N nodes at the t-th layer. In case $\phi(A) = D^{-1}A$ and we ignore the activation function σ, this leads to $H^{(T)} = \phi(A)^T H^{(0)} W^1$ with the T-th order transition matrix $(D^{-1}A)^T$ [5] as the pre-defined receptive field. That is, the depth of layers T determines the extent of neighbors to exploit, and any node j satisfies $d(i, j) \leq T$, where d is the shortest path distance, contributes to node i's embedding, with the importance weight pre-defined as $\phi(A)^T_{ij}$. Essentially, the receptive fields in graph domain are equivalent to subgraphs consist of paths.

Is there a specific path in the graph contributing mostly to the representation? Is there an adaptive and automated way of choosing the receptive field or path of a graph convolutional network? It seems that the current literature has not provided such a solution. For instance, graph convolution neural networks lie in spectral domain [2] heavily rely on the graph Laplacian matrix [3] $L = I - D^{-1/2}AD^{-1/2}$ to define the importance of the neighbors (and hence receptive field) for each nodes. Approaches lie in spatial domain define convolutions directly on the graph, with receptive field more or less hand-designed. For instance, GraphSage [9] used the mean or max of a fix-size neighborhood of each node, or an LSTM aggregator which needs a pre-selected order of neighboring nodes. These pre-defined receptive fields, either based on graph Laplacian in the spectral domain, or based on uniform operators like mean, max operators in the spatial domain, thus limit us from discovering meaningful receptive fields from graph data adaptively. For examples, the performance of GCN [15] based on graph Laplacian could deteriorate severely if we simply stack more and more layers to explore deeper and wider receptive fields (or paths).

To address the above challenges, (1) we propose adaptive path layer with two complementary components: adaptive breadth and depth functions, where the adaptive breadth function can adaptively select a set of significant important one-hop neighbors, and the

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] We collapse $\prod_{i=0}^{T} W^{(i)}$ as W, and $H^{(0)} = X \in \mathbb{R}^{N\times P}$.

adaptive depth function can extract and filter useful and noisy signals up to long order distance. (2) experiments on several datasets empirically show that our approaches are quite competitive especially on large graphs. Another remarkable result is that our approach is less sensitive to the depth of propagation layers.

Intuitively our proposed adaptive path layer guides the breadth and depth exploration of the receptive fields. As such, we name such adaptively learned receptive fields as receptive paths.

2 GRAPH CONVOLUTIONAL NETWORKS

Generalizing convolutions to graphs aims to encode the nodes with signals lie in the receptive fields. The output encoded embeddings can be further used in end-to-end supervised learning [15] or unsupervised learning tasks [7, 17].

The approaches lie in spectral domain heavily rely on the graph Laplacian operator $L = I - D^{-1/2}AD^{-1/2}$ [3]. The real symmetric positive semidefinite matrix L can be decomposed into a set of orthonormal eigenvectors which form the graph Fourier basis $U \in \mathbb{R}^{N\times N}$ such that $L = U\Lambda U^\top$, where $\Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \in \mathbb{R}^{N\times N}$ is the diagonal matrix with main diagonal entries as ordered nonnegative eigenvalues. As a result, the convolution operator in spatial domain can be expressed through the element-wise Hadamard product in the Fourier domain [2]. The receptive fields in this case depend on the kernel U. Kipf and Welling [15] further propose GCN to design the following approximated localized 1-order spectral convolutional layer:

$$H^{(t+1)} = \sigma\Big(\tilde{A}H^{(t)}W^{(t)}\Big), \tag{1}$$

where $\tilde{A}$ is a symmetric normalization of A with self-loops, i.e. $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$, $\hat{A} = A + I$, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, $H^{(t)} \in \mathbb{R}^{N,K}$ denotes the t-th hidden layer with $H^{(0)} = X$, $W^{(t)}$ is the layer-specific parameters, and σ denotes the activation functions.

GCN requires the graph Laplacian to normalize the neighborhoods in advance. This limits the usages of such models in inductive settings. One trivial solution (GCN-mean) instead is to average the neighborhoods:

$$H^{(t+1)} = \sigma\Big(\tilde{A}H^{(t)}W^{(t)}\Big), \tag{2}$$

where $\tilde{A} = \hat{D}^{-1}\hat{A}$ is the row-normalized adjacency matrix.

More recently, Hamilton et al. [8] proposed GraphSAGE, a method for computing node representation in inductive settings. They define a series of aggregator functions in the following framework:

$$H^{(t+1)} = \sigma\Big(\text{CONCAT}\big(\phi(A, H^{(t)}), H^{(t)}\big)W^{(t)}\Big), \tag{3}$$

where $\phi(\cdot)$ is an aggregator function over the graph. For example, their mean aggregator $\phi(A, H) = D^{-1}AH$ is nearly equivalent to GCN-mean (Eq. 2), but with additional residual architectures [11]. They also propose max pooling and LSTM (Long short-term memory) [13] based aggregators. However, the LSTM aggregator operates on a random permutation of nodes' neighbors that is not a permutation invariant function with respect to the ordering of neighborhoods, thus makes the operators hard to understand.

A recent work from Veličković et al. [19] proposed Graph Attention Networks (GAT) which uses attention mechanisms to parameterize the aggregator function $\phi(A, H; \Theta)$. This method is restricted to learn a direct neighborhood receptive field, which is not as general as our proposed approach which can explore receptive field in both breadth and depth directions.

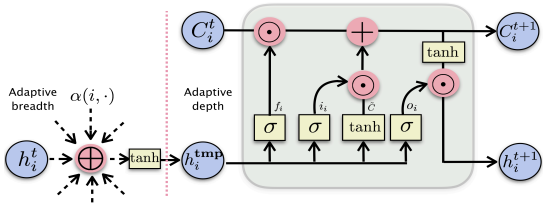

Figure 1: A demonstration for the architecture of GeniePath. Symbol $\bigoplus$ denotes the operator $\sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha(h_i^{(t)}, h_j^{(t)}) \cdot h_j^{(t)}$.

To summarize, the major efforts put on this area is to design effective functions that can propagate signals around the T-th order neighborhood for each node. Few of them try to learn the meaningful paths that direct the propagation. Our experiments in Figure 2 show that the mean operator, even the graph Laplacians cannot lead to indicative directions while propagating signals over the graph.

3 PROPOSED APPROACHES

In this section, we design neural network layers to learn "receptive paths" on graph.

3.1 Adaptive Path Layer

Our goal is to learn the breadths and depths of receptive paths adaptively. To explore the breadth of the receptive paths, we learn an adaptive breadth function $\phi(A, H^{(t)}; \Theta)$ parameterized by Θ, to aggregate the signals by adaptively assigning the importance of each neighbor. To explore the depth of the receptive paths, we learn an adaptive depth function $\varphi(\{f(h_i^{(t)}|h_i^{(t-1)})|t \in \{1, ...T\}\}; \Phi)$ parameterized by Φ that could further extract and filter the aggregated signals at the t-th order distance away from the anchor node i by modeling the dependencies of aggregated signals at various depths.

Now it remains to specify the adaptive breadth and depth function, i.e. $\phi(.)$ and $\varphi(.)$. We propose the following adaptive path layer.

$$h_i^{(\text{tmp})} = \tanh\left(W^{(t)\top} \sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha(h_i^{(t)}, h_j^{(t)}) \cdot h_j^{(t)}\right) \tag{4}$$

then

$$\begin{aligned} i_i &= \sigma(W_i^{(t)\top} h_i^{(\text{tmp})}), & f_i &= \sigma(W_f^{(t)\top} h_i^{(\text{tmp})}) \\ o_i &= \sigma(W_o^{(t)\top} h_i^{(\text{tmp})}), & \tilde{C} &= \tanh(W_c^{(t)\top} h_i^{(\text{tmp})}) \\ C_i^{(t+1)} &= f_i \odot C_i^{(t)} + i_i \odot \tilde{C}, \end{aligned}$$

and finally,

$$h_i^{(t+1)} = o_i \odot \tanh(C_i^{(t+1)}). \tag{5}$$

The i_i and f_i can be viewed as gated units that play the roles of extracting and filtering aggregated signals at the t-th order depth for node i. We maintain for each node i a memory C_i along the receptive paths for long dependencies of various depths, and ouput

Table 1: Dataset summary.

Dateset	Type	V	E	# Classes	# Features	Label rate (train / test)
Pubmed	Citation Network	19,717	88,651	3	500	0.3% / 5.07%
BlogCatalog[1]	Social Network	10,312	333,983	39	0	50% / 40%
BlogCatalog[2]	Social Network	10,312	333,983	39	128	50% / 40%
PPI	Protein-protein interaction	56,944	818,716	121	50	78.8% / 9.7%
Alipay	Account-Device Network	981,748	2,308,614	2	4,000	20.5% / 3.2%

the filtered singals as h_i^{t+1} given the memory. Furthermore, we parameterize the generalized linear attention operator $\alpha(\cdot,\cdot)$ that assigns the importance of neighbors as follows

$$\alpha(x,y) = \text{softmax}_y\Big(v^\top \tanh(W_s^\top\ x + W_d^\top\ y)\Big), \tag{6}$$

and $\text{softmax}_y f(\cdot,y) = \frac{\exp f(\cdot,y)}{\sum_{y'} \exp f(\cdot,y')}$. The overall architectures of adaptive path layer is illustrated in Figure 1. The first equation (Eq. (4)) corresponds to $\phi(.)$ and the rest gated units correspond to $\varphi(.)$. We let $h_i^{(0)} = W_x^\top X_i$ with weight matrix $W_x \in \mathbb{R}^{D,k}$ and node i's feature $X_i \in \mathbb{R}^D$.

Note that the adaptive path layer with only adaptive breadth function $\phi(.)$ reduces to the proposal of GAT, but with stronger non-linear representation capacities. Our generalized linear attention can also assign symmetric importance with constraint $W_s = W_d$.

A variant. Next, we propose a variant called "GeniePath-lazy" that postphones the evaluation of the adaptive depth functions $\varphi(.)$ at each depth. We rather propagate signals up to T-th order distance by merely stacking adaptive breadth functions. Given those T hidden units $\{h_i^{(t)}\}$, we add adaptive depth function $\varphi(.)$ on top of them to further extract and filter the signals at various depths. We initialize $\mu_i^{(0)} = W_x^\top X_i$, and feed $\mu_i^{(T)}$ to the final softmax or sigmoid layers for classification. Formally given $\{h_i^{(t)}\}$ we have:

$$\begin{aligned}
i_i &= \sigma\big({W_i^{(t)}}^\top \text{CONCAT}(h_i^{(t)}, \mu_i^{(t)})\big), \\
f_i &= \sigma({W_f^{(t)}}^\top \text{CONCAT}\big(h_i^{(t)}, \mu_i^{(t)})\big), \\
o_i &= \sigma({W_o^{(t)}}^\top \text{CONCAT}\big(h_i^{(t)}, \mu_i^{(t)})\big), \\
\tilde{C} &= \tanh({W_c^{(t)}}^\top \text{CONCAT}\big(h_i^{(t)}, \mu_i^{(t)})\big), \\
C_i^{(t+1)} &= f_i \odot C_i^{(t)} + i_i \odot \tilde{C}, \\
\mu_i^{(t+1)} &= o_i \odot \tanh(C_i^{(t+1)}).
\end{aligned} \tag{7}$$

4 EXPERIMENTS

In this section, we first discuss the experimental results of our approach evaluated on various types of graphs compared with strong baselines. We then study its abilities of learning adaptive depths. Finally we give qualitative analyses on the learned paths compared with graph Laplacian.

4.1 Transductive and Inductive Settings

We summarize the differences as follows.

Transductive setting. In transductive settings, we allow all the algorithms to access to the whole graph, i.e. all the edges among the nodes, and all of the node features.

Inductive setting. In inductive settings, all the nodes in test are *completely unobserved* during the training procedure.

4.2 Datasets

We conduct the comparison experiments on both transductive settings and inductive settings.

Transductive setting.

The *BlogCatalog*[1] [20] is a type of social networks, where nodes correspond to bloggers listed on BlogCatalog websites, and the edges to the social relationships of those bloggers. There are a total of 10,312 nodes, 333,983 edges, and 39 classes. We treat the problem as a multi-label classification problem. We randomly split 50% (10%) of nodes for training (validation), and the rest 40% for testing. Different from other datasets, the BlogCatalog[1] has no explicit features available, as a result, we encode node ids as one-hot features for each node, i.e. 10,312 dimensional features.

To further study the performance of learning on the same graph with features, we decompose the adjacency matrix A of BlogCatalog[1] by SVD (Singular-value decomposition). We use the 128 dimensional transformed singular-vectors as features for each node. Such features can be viewed as global features directly decomposed from the whole graph. We name this dataset as *BlogCatalog*[2].

The *Alipay* dataset [16] is a type of Account-Device Network, built for detecting malicious accounts in the online cashless payment system at Alipay. The nodes correspond to users' accounts and devices logged in by those accounts. The edges correspond to the login relationships between accounts and devices during a time period. Node features are counts of login behaviors discretized into hours and account profiles. There are 2 classes in this dataset, i.e. malicious accounts and normal accounts. The Alipay data set is random sampled during one week. There are a total of 981,748 nodes, 2,308,614 edges and 4,000 sparse features per node. The dataset consists of 82,246 subgraphs.

Inductive setting.

The *PPI* [8] is a type of protein-protein interaction networks, which consists of 24 subgraphs with each corresponds to a human tissue [21]. The node features are extracted by positional gene sets, motif gene sets and immunological signatures. There are 121 classes for each node from gene ontology. Each node could have multiple labels, then results into a multi-label classification problem. We use the exact preprocessed data provided by Hamilton et al. [8]. There are 20 graphs for training, 2 for validation and 2 for testing.

We summarize the statistics of all the datasets in Table 1.

4.3 Experimental Settings

We describe our experimental settings as follows.

Table 2: Summary of testing results on BlogCatalog and Alipay in the transductive setting. In accordance with former benchmarks, we report Macro-F1 for BlogCatalog, and F1 for Alipay.

Transductive			
Methods	BlogCatalog[1]	BlogCatalog[2]	Alipay
MLP	-	0.134	0.741
node2vec	0.136	0.136	-
Chebyshev	0.160	0.166	0.784
GCN	0.171	0.174	0.796
GraphSAGE*	0.175	0.175	0.798
GAT	**0.201**	0.197	0.811
GeniePath*	0.195	**0.202**	**0.826**

Table 3: Summary of testing Micro-F1 results on PPI in the inductive setting.

Inductive	
Methods	PPI
MLP	0.422
GCN-mean	0.71
GraphSAGE*	0.768
GAT	0.81
GeniePath*	**0.979**

Table 4: A comparison of GAT, GeniePath, and additional residual "skip connection" on PPI.

Methods	PPI
GAT	0.81
GAT-residual	0.914
GeniePath	0.952
GeniePath-lazy	0.979
GeniePath-lazy-residual	**0.985**

4.3.1 Comparison Approaches. (**1**) MLP, which is the classic multilayer perceptron consists of multiple layers of fully connected neurons. The approach only utilizes the node features but does not consider the structures of the graph.

(**2**) node2vec [7]. In our experiments, we feed the embeddings to a softmax layer or a sigmoid layer depends on the type of classification problem we are dealing with. Note that, this type of methods built on top of lookup embeddings cannot work on problems with multiple graphs, i.e. on datasets Alipay and PPI, because without any further constraints, the entire embedding space cannot guarantee to be consistent during training [8].

(**3**) Chebyshev [4], which approximate the graph spectral convolutions by a truncated expansion in terms of Chebyshev polynomials up to T-th order. This method needs the graph Laplacian in advance, so that it only works in the transductive setting.

(**4**) GCN [15], which is defined in Eq. (1). Same as Chebyshev, it works only in the transductive setting. However, if we just use the normalization formulated in Eq. (2), it can work in a inductive setting. We denote this variant of GCN as GCN-mean.

(**5**) GraphSAGE [8], which consists of a group of inductive graph representation methods with different aggregator functions. The GCN-mean with residual connections is nearly equivalent to GraphSAGE using mean for pooling. We will report the best results of GraphSAGEs with different pooling strategies as GraphSAGE*.

(**6**) Graph Attention Networks (GAT) [19] is similar to a reduced version of our approach with only adaptive breadth function. This can help us understand the usefulness of adaptive breadth and depth function.

We pick the better results from GeniePath or GeniePath-lazy as GeniePath*. We found out that the skip connections [12] are useful for stacking deeper layers for various approaches and will report the approaches with suffix "-residual" to distinguish the differences for better understandings of the contribution of different approaches.

4.3.2 Experimental Setups. In our experiments, we implement our algorithms in TensorFlow [1] with the Adam optimizer [14]. For all the graph convolutional network-style approaches, we set the hyperparameters include the dimension of embeddings or hidden units, the depth of hidden layers and learning rate to be same. For node2vec, we tune the return parameter p and in-out parameter q by grid search. Note that setting $p = q = 1$ is equivalent to DeepWalk [7]. We sample 10 walk-paths with walk-length as 80 for each node in the graph. Additionally, we tune the penalty of ℓ_2 regularizers for different approaches.

For pubmed, we set the number of hidden units as 16 with 2 hidden layers. For BlogCatalog[1], we set the number of hidden units as 128 with 3 hidden layers. For BlogCatalog[2], we set the number of hidden units as 128 with 3 hidden layers. For Alipay, we set the number of hidden units as 16 with 7 hidden layers. For PPI, we set the number of hidden units as 256 with 3 hidden layers.

4.4 Classification

We report the comparison results of transductive settings in Table 2.

In BlogCatalog[1] and BlogCatalog[2], we found both GAT and GeniePath work the best compared with other methods. Another suprisingly interesting result is that graph convolutional networks-style approaches can perform well even without explicit features.

Alipay [16] is a dataset with nearly 1 million nodes, the largest graph ever reported in the literatures of graph convolutional networks. The graph is relative sparse, and we stack 7 hidden layers. We found that GeniePath works quite promising on this large graph. The results show that adaptively learning the receptive paths are meaningful. Since the dataset consists of ten thousands of subgraphs, the node2vec is not applicable in this case.

We report the comparison results of inductive settings on PPI in Table 3. Because GCN does not work in inductive settings, we use the alternative "GCN-mean" by averaging the neighborhoods instead. GeniePath performs extremely promising results on this big graph, and shows that the adaptive depth function plays way important on this data compared to GAT with only adaptive breadth function.

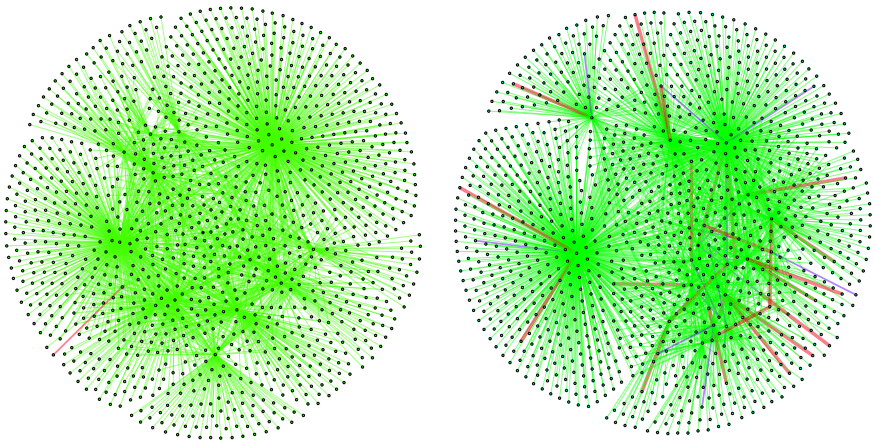

Figure 2: Graph Laplacian (left) v.s. Estimated receptive paths (right) with respect to the black node on PPI dataset: we retain all the paths to the black node in 2-step that involved in the propagation, with edge thickness denotes the importance $w_{i,j} = \alpha(h_i, h_j)$ of edge (i, j) estimated in the first adaptive path layer. We also discretize the importance of each edge into 3 levels: Green ($w_{i,j} < 0.1$), Blue ($0.1 \leq w_{i,j} < 0.2$), and Red ($w_{i,j} \geq 0.2$).

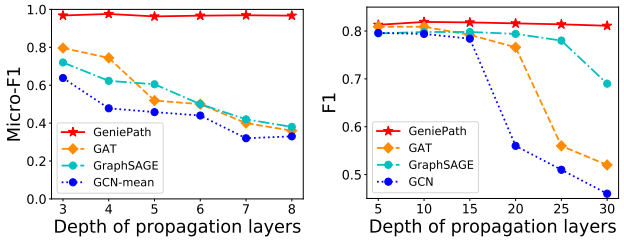

Figure 3: The classification measures with respect to the depths of propagation layers: PPI (left), Alipay (right).

To further study the usefulness of residual architectures in various approaches, we compare GAT, GeniePath, and the variant GeniePath-lazy with additional residual architecture ("skip connections") in Table 4. The results of PPI show that GeniePath is less sensitive to the additional "skip connections". The significant improvement of GAT seems rely on the residual structure.

4.5 Depths of Propagation

We show our results on classification measures with respect to the depths of propagation layers in Figure 3. As we stack more graph convolutional layers, i.e. with deeper and broader receptive fields, GCN, GAT and even GraphSAGE with residual architectures can hardly maintain consistently resonable results. Interestingly, GeniePath with adaptive path layers can adaptively learn the receptive paths and achieve consistent results, which is remarkable.

4.6 Qualitative Analysis

We show a qualitative analysis about the receptive paths with respect to a sampled node learned by GeniePath in Figure 2. It can be seen, the graph Laplacian assigns importance of neighbors at nearly the same scale, i.e. results into very dense paths. However, the GeniePath can help select significantly important paths to propagate while ignoring the rest, i.e. results into much sparser paths. Such "neighbor selection" processes essentially lead the direction of the receptive paths and improve the effectiveness of propagation.

5 CONCLUSION

In this paper, we studied the problems of prior graph convolutional networks on identifying meaningful receptive paths. We propose adaptive path layers with adaptive breadth and depth functions to essentially guide the receptive paths. Experiments on large graphs show that our approaches are quite competitive, and are less sensitive to the depths of the stacked layers , as we showed in the node classification tasks.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. 2016. TensorFlow: a system for large-scale machine learning. (2016).
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[3] Fan RK Chung. 1997. *Spectral graph theory.* Number 92. American Mathematical Soc.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances*

in Neural Information Processing Systems. 3844–3852.
[5] Paul A Gagniuc. 2017. *Markov Chains: From Theory to Implementation and Experimentation*. John Wiley & Sons.
[6] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344* (2016).
[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *arXiv preprint arXiv:1706.02216* (2017).
[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *arXiv preprint arXiv:1709.05584* (2017).
[10] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 630–645.
[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[14] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[16] Ziqi Liu, Chaochao Chen, Jun Zhou, Xiaolong Li, Feng Xu, Tao Chen, and Le Song. 2017. POSTER: Neural Network-based Graph Embedding for Malicious Accounts Detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 2543–2545. https://doi.org/10.1145/3133956.3138827
[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[18] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.
[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *arXiv preprint arXiv:1710.10903* (2017).
[20] Reza Zafarani and Huan Liu. 2009. Social computing data repository at ASU.
[21] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33, 14 (2017), i190–i198.