SIGN: Scalable Inception Graph Neural Networks

Fabrizio Frasca [*1] **Emanuele Rossi** [*1] **Davide Eynard** [1] **Benjamin Chamberlain** [1] **Michael Bronstein** [1 2] **Federico Monti** [1]

Abstract

The popularity of graph neural networks has sparked interest, both in academia and in industry, in developing methods that scale to very large graphs such as Facebook or Twitter social networks. In most of these approaches, the computational cost is alleviated by a sampling strategy retaining a subset of node neighbors or subgraphs at training time. In this paper we propose a new, efficient and scalable graph deep learning architecture, which sidesteps the need for graph sampling by using graph convolutional filters of different size that are amenable to efficient precomputation, allowing extremely fast training and inference. Our architecture allows using different local graph operators (e.g. motif-induced adjacency matrices or Personalized Page Rank diffusion matrix) to best suit the task at hand. We conduct extensive experimental evaluation on various open benchmarks and show that our approach is competitive with other state-of-the-art architectures, while requiring a fraction of training and inference time.

1. Introduction

Deep learning on graphs, also known as *geometric deep learning* (GDL) (Bronstein et al., 2017) or *graph representation learning* (GRL) (Hamilton et al., 2017b; Battaglia et al., 2018; Zhang et al., 2018), has emerged in a matter of just a few years from a niche topic to one of the most prominent fields in machine learning. Graph deep learning models have recently scored successes in various applications relying on modeling relational data, see e.g. (Zhang & Chen, 2018; Qi et al., 2018; Monti et al., 2016; Choma et al., 2018; Duvenaud et al., 2015; Gilmer et al., 2017; Parisot et al., 2018; Zitnik et al., 2018; Veselkov et al., 2019; Gainza et al., 2019; Rossi et al., 2019; Monti et al., 2019). Graph neural networks (GNNs) seek to generalize classical convolutional architectures (CNNs) to graph-structured data, with a wide variety of convolution-like operations available in the literature (Scarselli et al., 2008; Defferrard et al., 2016; Atwood & Towsley, 2016; Niepert et al., 2016; Simonovsky & Komodakis, 2017; Monti et al., 2016; Kipf & Welling, 2017; Wu et al., 2019; Velickovic et al., 2018; Hamilton et al., 2017a).

Until recently, most of the research in the field has focused on small-scale datasets, and relatively little effort has been devoted to scaling these methods to web-scale graphs. Scaling is a major challenge precluding the wide application of graph deep learning methods in industrial settings. Compared to Euclidean neural networks where the training loss can be decomposed into individual samples and computed independently, graph convolutional networks diffuse information between nodes along the edges of the graph, making the loss computation interdependent for different nodes. Furthermore, in typical graphs the number of nodes grows exponentially with the increase of the filter receptive field, incurring significant computational and memory complexity. So far, various *graph sampling* approaches (Hamilton et al., 2017a; Ying et al., 2018; Chen et al., 2018; Huang et al., 2018; Chen & Zhu, 2018; Chiang et al., 2019; Zeng et al., 2019; Zou et al., 2019) have been proposed as a way to alleviate the cost of training graph neural networks by selecting a small number of neighbors that reduce the computational and memory complexity. Such methods can potentially scale to web-size graphs (Ying et al., 2018).

In this paper, we take a different approach for scalable deep learning on graphs. We propose SIGN, a simple scalable graph neural network architecture, inspired by the inception module (Szegedy et al., 2015; Kazi et al., 2019), which generalizes several previous methods such as GCN (Kipf & Welling, 2017), S-GCN (Wu et al., 2019), ChebNet (Defferrard et al., 2016), and MotifNet (Monti et al., 2018). SIGN combines graph convolutional filters of different types and sizes that are amenable to efficient precomputation, allowing extremely fast training and inference with complexity independent on the graph structure.

The most important observation of our paper is that employing SIGN with only one graph convolutional layer, we are

[*]Equal contribution [1]Twitter, London [2]Imperial College London. Correspondence to: Fabrizio Frasca <ffrasca@twitter.com>.

Graph Representation Learning and Beyond (GRL+) Workshop at the 37th International Conference on Machine Learning, ICML 2020 , Vienna, Austria, Copyright 2020 by the author(s).

Table 1. Theoretical time complexity, where $L_{\mathrm{c}}, L_{\mathrm{ff}}$ is the number of graph convolition and MLP layers, r is the filter size, N the number of nodes (in training or inference, respectively), $|\mathcal{E}|$ the number of edges, and d the feature dimensionality (assumed fixed for all layers). For GraphSAGE, k is the number of sampled neighbors per node. Forward pass complexity corresponds to an entire epoch where all nodes are seen.

	Preproc.	*Forward Pass*
GraphSAGE	$\mathcal{O}(k^{L_{\mathrm{c}}}N)$	$\mathcal{O}(k^{L_{\mathrm{c}}}Nd^2)$
ClusterGCN	$\mathcal{O}(\|\mathcal{E}\|)$	$\mathcal{O}(L_{\mathrm{c}}\|\mathcal{E}\|d + L_{\mathrm{ff}}Nd^2)$
GraphSAINT	$\mathcal{O}(kN)$	$\mathcal{O}(L_{\mathrm{c}}\|\mathcal{E}\|d + L_{\mathrm{ff}}Nd^2)$
SIGN-r	$\mathcal{O}(r\|\mathcal{E}\|d)$	$\mathcal{O}(rL_{\mathrm{ff}}Nd^2)$

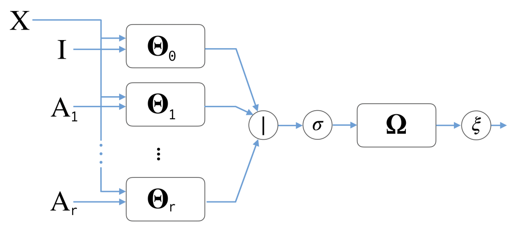

Figure 1. The SIGN architecture for r generic graph filtering operators. $\mathbf{\Theta}_k$ represents the k-th dense layer transforming node-wise features downstream the application of operator k, | is the concatenation operation and $\mathbf{\Omega}$ refers to the dense layer used to compute final predictions.

able to achieve results on par with the state-of-the-art, while being faster in training and, especially, inference (even one order of magnitude speedup). We provide extensive experimental validation of this claim on large-scale graph learning datasets. This result raises the important question on when deep graph neural network architectures are useful, especially when scalability is required. Significant effort has recently been devoted to methods allowing to design deep graph neural networks with many graph convolutional layers (Xu et al., 2018; Gong et al., 2020; Li et al., 2019; Zhao & Akoglu, 2020; Rong et al., 2020), which otherwise appear difficult to train (Li et al., 2018; Klicpera et al., 2018; Wu et al., 2020). We conjecture, on the contrary, that deep graph learning architectures are not useful for general irregular graphs and argue that future research in the field should focus on designing local more expressive operators (Barbarossa & Sardellitti, 2019; Monti et al., 2018; Flam-Shepherd et al., 2020) rather than going deeper.

2. Scalable Inception Graph Neural Networks

In this work we propose SIGN, an alternative method to scale graph neural networks to very large graphs. The key building block of our architecture is a set of linear diffusion operators represented as $n \times n$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_r$, whose application to the node-wise features can be pre-computed.

Let $\mathcal{G} = (\mathcal{V} = \{1, \ldots, n\}, \mathcal{E}, \mathbf{W})$ be an undirected weighted graph, represented by the symmetric $n \times n$ *adjacency matrix* $\mathbf{W}$, where $w_{ij} > 0$ if $(i,j) \in \mathcal{E}$ and zero otherwise. Let also $\mathbf{D} = \mathrm{diag}(\sum_{j=1}^n w_{1j}, \ldots, \sum_{j=1}^n w_{nj})$ be the *degree matrix*. We denote by $\mathbf{A} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ the normalized adjacency matrix. We further assume that each node is endowed with a d-dimensional feature vector and arrange all the node features as rows of the $n \times d$ matrix $\mathbf{X}$. For node-wise classification tasks, our architecture has the form (Figure 1):

$$\begin{aligned} \mathbf{Z} &= \sigma\left([\mathbf{X}\mathbf{\Theta}_0, \mathbf{A}_1\mathbf{X}\mathbf{\Theta}_1, \ldots, \mathbf{A}_r\mathbf{X}\mathbf{\Theta}_r]\right) \\ \mathbf{Y} &= \xi\left(\mathbf{Z}\mathbf{\Omega}\right), \end{aligned} \tag{1}$$

where $\mathbf{\Theta}_0, \ldots, \mathbf{\Theta}_r$ and $\mathbf{\Omega}$ are learnable matrices respectively of dimensions $d \times d'$ and $d'(r+1) \times c$ for c classes, and σ, ξ are non-linearities, the second one computing class probabilities. We denote a model with r operators by *SIGN-r*.

A key observation is that matrix products $\mathbf{A}_1\mathbf{X}, \ldots, \mathbf{A}_r\mathbf{X}$, in equation (1) *do not depend* on the learnable model parameters and can be easily precomputed. For large graphs, distributed computing infrastructures such as Apache Spark can speed up computation. This effectively reduces the computational complexity of the overall model to that of a multi-layer perceptron (MLP), i.e. $\mathcal{O}(rL_{\mathrm{ff}}Nd^2)$, where d is the number of features, N the number of nodes in the training/testing graph and L_{ff} is the overall number of feed-forward layers in the model. Table 1 compares the complexity of our SIGN model to other scalable architectures GraphSAGE, ClusterGCN, and GraphSAINT. Importantly, the forward and backward pass complexity of our model does not depend on the graph structure, in contrast to graph sampling-based methods such as ClusterGCN and GraphSAINT that can potentially be significantly slowed down by 'unfriendly' graphs. Unlike the aforementioned methods, SIGN is not based on sampling nodes or subgraphs, operations potentially introducing bias in training.

Choice of the operators. Generally speaking, the choice of the diffusion operators jointly depends on the task, graph structure, and the features. In social networks, operators induced by triangles or cliques might help distinguishing edges representing weak or strong ties (Granovetter, 1982). In graphs with noisy connectivity, it was shown diffusion operators based on personalized PageRank (PPR) or heat kernel can boost performance (Klicpera et al., 2019). In our experiments, we choose three specific types of operators: simple (normalized) adjacency, personalized PageRank-based adjacency, and triangle-based adjacency matrices, as well as their powers. We denote by *SIGN(p,s,t)* with $r = p + s + t$ the configuration using p, s, and t powers of simple, PPR-based, and triangle-based adjacency matrices, respectively.

Table 2. Micro-averaged F1 score average and standard deviation over 10 runs with the same train/val/test split but different random model initialization. For SIGN, we show the best performing configurations. The top three performance scores are highlighted as: **First**, **Second**, **Third**. † Requires a GPU with 33GB of memory.

					OGBN-Products		
	Reddit	*Flickr*	*PPI*	*Yelp*	*Train*	*Val*	*Test*
GCN	0.933±0.000	0.492±0.003	0.515±0.006	0.378±0.001	0.929±0.001	0.917±0.001	0.757±0.002†
FastGCN	0.924±0.001	**0.504±0.001**	0.513±0.032	0.265±0.053	—	—	—
Stoch.-GCN	**0.964±0.001**	0.482±0.003	**0.963±0.010**	**0.640±0.002**	—	—	—
AS-GCN	0.958±0.001	**0.504±0.002**	0.687±0.012	—	—	—	—
GraphSAGE	0.953±0.001	0.501±0.013	0.637±0.006	0.634±0.006	0.930±0.002	0.916±0.001	0.780±0.002†
ClusterGCN	0.954±0.001	0.481±0.005	0.875±0.004	0.609±0.005	**0.928±0.004**	**0.904±0.003**	**0.752±0.004**
GraphSAINT	**0.966±0.001**	**0.511±0.001**	**0.981±0.004**	**0.653±0.003**	**0.933±0.001**	**0.918±0.001**	**0.773±0.002**
SIGN	**0.968±0.000**	**0.514±0.001**	**0.970±0.003**	0.625±0.003	**0.970±0.005**	**0.931±0.001**	**0.776±0.001**
(p,s,t)	$(4,2,0)$	$(4,0,1)$	$(2,0,1)$	$(2,0,1)$		$(5,3,0)$	

Inception-like module. In the configuration *SIGN(r,0,0)*, we use the GCN-normalized adjacency matrix $\mathbf{B} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}$ and define $\mathbf{A}_k = \mathbf{B}^k$ for $k = 1, \ldots, r$. This model is analogous to the popular *Inception module* (Szegedy et al., 2015) for classic CNN architectures: it consists of convolutional filters of different sizes determined by the parameter r, where $r = 0$ corresponds to 1×1 convolutions in the inception module (amounting to linear transformations of the features in each node without diffusion across nodes). Owing to this analogy, we refer to our model as the Scalable Inception Graph Network (SIGN). It is also easy to observe that various graph convolutional layers can be obtained as particular settings of (1). In particular, by setting the σ non-linearity to PReLU (He et al., 2015), ChebNet, GCN, and S-GCN can be automatically learnt if suitable diffusion operator $\mathbf{B}$ and activation ξ are used (see Supplementary Materials).

3. Experiments

Datasets. We evaluated SIGN on node-wise classification tasks, both in transductive and inductive settings. Inductive experiments were performed on four datasets: *Reddit* (Hamilton et al., 2017a), *Flickr*, *Yelp* (Zeng et al., 2019), and *PPI* (Zitnik & Leskovec, 2017). To date, these are the largest graph learning inductive node classification benchmarks available in the public domain. Transductive experiments were performed on the new *OGBN-Products* dataset (Hu et al., 2020), the largest semi-supervised node classification instance from the Open Graph Benchmark at the time of writing. Furthermore, we tested the scalability of our method on *Wikipedia* links (Kon, 2017), a large-scale network of links between articles in the English version of Wikipedia. Statistics for all the datasets are reported in Supplementary Materials.

Setup. We experimented with several *SIGN*(p, s, t) configurations, with p the maximum power of the GCN-normalized adjacency matrix, s that of a random-walk normalized PPR diffusion operator (Klicpera et al., 2019), and t that of a row-normalized triangle-induced adjacency matrix (Monti et al., 2018), with weights proportional to edge occurrences in closed triads. PPR-based operators are computed from a symmetrically normalized adjacency transition matrix in an approximated form, with a restart probability of $\alpha = 0.01$ for inductive datasets and $\alpha = 0.05$ in the transductive case. The Wikipedia dataset, due to the lack of node attributes and labels, is only used to assess scalability: we randomly generate 100-dimensional node feature vectors and scalar targets and consider the whole network for both training and inference. We refer the reader to the Supplementary Materials for specifics on the hyperparameter tuning procedure and further implementation details.

Baselines. On the inductive datasets, we compare our method to *GCN* (Kipf & Welling, 2017), *FastGCN* (Chen et al., 2018), *Stochastic-GCN* (Chen & Zhu, 2018), *AS-GCN* (Huang et al., 2018), *GraphSAGE* (Hamilton et al., 2017a), *ClusterGCN* (Chiang et al., 2019), and *GraphSAINT* (Zeng et al., 2019), which constitute the current state-of-the-art. On OGBN-Products we compare our performance against the scalable *ClusterGCN* (Chiang et al., 2019), and *GraphSAINT* (Zeng et al., 2019). We additionally report the results attained by *GCN* (Kipf & Welling, 2017) and *GraphSAGE* (Hamilton et al., 2017a), although they exhibit an intractable memory footprint on this dataset (Hu et al., 2020).

Performance. Table 2 presents the results on all the inductive datasets, as well as the transductive *OGBN-Products*. We report the best performing SIGN configuration; results for other configurations and further ablations are in Supplementary Materials. For the inductive datasets we report, in line with (Zeng et al., 2019), the micro-averaged F1 score means and standard deviations computed over 10 runs. SIGN outperforms other methods on Reddit and Flickr, and performs competitively to state-of-the-art on PPI. Our performance on Yelp is worse than in the other datasets; we hypothesize that a more tailored operators choice is required to better suit the characteristics of this dataset. As

Table 3. Mean and standard deviation of preprocessing, training (one epoch) and inference times, in seconds, on *Wikipedia*, computed over 10 runs. *SIGN-r* denotes architecture with r precomputed operators. Preprocessing and training times for ClusterGCN are not reported due to the clustering algorithm failing to complete.

	OGBN-Products			*Wikipedia*		
	Preprocessing	*Training*	*Inference*	*Preprocessing*	*Training*	*Inference*
ClusterGCN	36.93 ± 0.52	13.34 ± 0.16	93.00 ± 0.68	—	—	183.76 ± 3.01
GraphSAINT	52.06 ± 0.54	2.89 ± 0.05	94.76 ± 0.81	123.60 ± 1.60	135.73 ± 0.06	209.86 ± 4.73
SIGN-2	88.21 ± 1.33	1.04 ± 0.10	2.86 ± 0.10	192.88 ± 0.12	62.37 ± 0.17	13.40 ± 0.15
SIGN-4	160.16 ± 1.20	1.54 ± 0.04	3.79 ± 0.08	326.21 ± 1.14	93.84 ± 0.08	18.15 ± 0.05
SIGN-6	226.48 ± 1.43	2.05 ± 0.00	4.84 ± 0.08	459.24 ± 0.14	125.24 ± 0.03	22.94 ± 0.02
SIGN-8	297.92 ± 2.92	2.53 ± 0.04	5.88 ± 0.09	598.67 ± 0.82	154.73 ± 0.12	27.69 ± 0.11

for *OGBN-Products*, in accordance to (Hu et al., 2020), we report performance on training, validation, and test sets. Our proposed model attains state-of-the-art results using the $(5, 3, 0)$ configuration by outperforming other methods on all these sets, demonstrating the ability of SIGN to generalize over the out-of-distribution evaluation sets proposed in OGB (Hu et al., 2020).

Operator combination. We notice that best performance is obtained on each benchmark by a specific combination of operators, remarking the fact that each dataset features particular topological and content characteristics requiring suitable filters. Interestingly, we also observe that, while the PPR operators do not bring significant improvements in the inductive setting, they are beneficial on the transductive OGBN-Products. This is in accordance with (Klicpera et al., 2019), where the effectiveness of PPR diffusion operators in transductive settings has been extensively studied.

Runtime. While performing on par or better than state-of-the-art methods on most benchmarks in terms of accuracy, our method has the advantage of being significantly faster than other methods for large graphs. Average training, inference, and preprocessing times for our largest datasets, i.e. *Wikipedia* and *OGBN-Products*, are reported in Table 3. Our model is of comparable speed w.r.t. GraphSAINT in training[1], while being by far the fastest approach in inference: all SIGN architectures are always at least one order of magnitude faster than other methods. SIGN's preprocessing is slightly longer than other methods, but we notice that most of the calculations can be cast as sparse matrix multiplications and are easily parallelized with frameworks for distributed computing. We envision to engineer faster and even more scalable SIGN preprocessing implementations in future developments of this work.

4. Conclusion and Future Work

Our architecture achieves a good trade off between simplicity, allowing efficient and scalable applications to very large graphs and very fast training and inference, and expressiveness, with competitive performances on a variety of applications on common graph learning benchmarks. For this reason, SIGN is well suited for industrial large-scale systems. Remarkably, we use only one graph convolutional layer and hence a shallow architecture.

In light of our results, the most important question is when (or whether at all) one should apply deep architectures to graphs, where by 'depth' we refer to the number of stacked graph convolutional layers. We conjecture that going deep with graph neural networks on irregular graphs is of little use and believe that a promising direction for future research is, rather than 'going deep', to 'go wide', in the sense of exploring more expressive local operators. In our experiments, triangle-based operators showed promising results. Possible extensions can employ operators that account for higher-order structures such as simplicial complexes (Barbarossa & Sardellitti, 2019), paths (Flam-Shepherd et al., 2020), or motifs (Monti et al., 2018) that can be tailored to the specific problem. Furthermore, temporal information can be integrated e.g. in the form of temporal motifs (Paranjape et al., 2017).

[1]Training time is measured as forward-backward time to complete one epoch.

References

Wikipedia links, english network dataset – KONECT, April 2017. URL `http://konect.uni-koblenz.de/networks/wikipedia_link_en`.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1993–2001, 2016.

Barbarossa, S. and Sardellitti, S. Topological signal processing over simplicial complexes. *arXiv:1907.11577*, 2019.

Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and

Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2546–2554. Curran Associates, Inc., 2011.

Bhatia, K., Dahiya, K., Jain, H., Mittal, A., Prabhu, Y., and Varma, M. The extreme classification repository: Multi-label datasets and code, 2016. URL http://manikvarma.org/downloads/XC/XMLRepository.html.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Proc. Magazine*, 34(4): 18–42, July 2017. ISSN 1053-5888.

Chen, J. and Zhu, J. Stochastic training of graph convolutional networks, 2018.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*, 2019.

Choma, N., Monti, F., Gerhardt, L., Palczewski, T., Ronaghi, Z., Prabhat, P., Bhimji, W., Bronstein, M., Klein, S., and Bruna, J. Graph neural networks for icecube signal classification. In *ICMLA*, 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*. 2015.

Flam-Shepherd, D., Wu, T., Friederich, P., and Aspuru-Guzik, A. Neural message passing on high order paths. *arXiv:2002.10413*, 2020.

Gainza, P. et al. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17:184–192, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Gong, S., Bahri, M., Bronstein, M. M., and Zafeiriou, S. Geometrically principled connections in graph neural networks. In *Proc. CVPR*, 2020.

Granovetter, M. The strength of weak ties: A network theory revisited. In *Sociological Theory*, pp. 105–130, 1982.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017b.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, abs/2005.00687, 2020. URL https://arxiv.org/abs/2005.00687.

Huang, W., Zhang, T., Rong, Y., and Huang, J. Adaptive sampling towards fast graph representation learning. In *NIPS*, 2018.

Kazi, A. et al. Inceptiongcn: Receptive field aware graph convolutional network for disease prediction. In *Information Processing in Medical Imaging*, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv:1810.05997*, 2018.

Klicpera, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Li, G., Muller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *Proc. ICCV*, 2019.

Li, R., Wang, S., Zhu, F., and Huang, J. Adaptive graph convolutional neural networks. In *Proc. AAAI*, 2018.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2016.

Monti, F., Otness, K., and Bronstein, M. M. Motifnet: a motif-based graph convolutional network for directed graphs. In *DSW*, 2018.

Monti, F., Frasca, F., Eynard, D., Mannion, D., and Bronstein, M. M. Fake news detection on social media using geometric deep learning. *CoRR*, abs/1902.06673, 2019. URL http://arxiv.org/abs/1902.06673.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In Balcan, M. and Weinberger, K. Q. (eds.), *International Conference on*

Machine Learning, ICML, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2014–2023. JMLR.org, 2016.

Paranjape, A., Benson, A. R., and Leskovec, J. Motifs in temporal networks. In *Proc. Web Search and Data Mining*, 2017.

Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., and Rueckert, D. Disease prediction using graph convolutional networks: Application to autism spectrum disorder and alzheimer's disease. *Med Image Anal*, 48:117–130, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alche-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Qi, S., Wang, W., Jia, B., Shen, J., and Zhu, S.-C. Learning human-object interactions by graph parsing neural networks. In *ECCV*, pp. 401–417, 2018.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *Proc. ICLR*, 2020.

Rossi, E., Monti, F., Bronstein, M., and Liò, P. ncrna classification with graph convolutional networks. In *KDD Workshop on Deep Learning on Graphs*, 2019.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *Trans. Neural Networks*, 20(1):61–80, 2008.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 29–38. IEEE Computer Society, 2017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Veselkov, K. et al. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific Reports*, 9(1):1–12, 2019.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *ICML*, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks and Learning Systems*, 2020.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. 12 2018.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. Graphsaint: Graph sampling based inductive learning method. *arXiv:1907.04931*, 2019.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *NIPS*, 2018.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *arXiv:1812.04202*, 2018.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *Proc. ICLR*, 2020.

Zitnik, M. and Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33 (14):i190–i198, Jul 2017. ISSN 1460-2059. doi: 10.1093/bioinformatics/btx252. URL http://dx.doi.org/10.1093/bioinformatics/btx252.

Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems*, 2019.

Supplementary Materials

Generalization of other graph convolutional layers

By setting the σ non-linearity to PReLU (He et al., 2015), SIGN can automatically learn ChebNet, GCN, and S-GCN if suitable diffusion operator $\mathbf{B}$ and activation ξ are used (see Table 4).

Table 4. SIGN can replicate some popular graph convolutional layers (α represents the learnable parameter of a PReLU activation).

	$\mathbf{B}_1, \dots, \mathbf{B}_r$	α	r	$\mathbf{\Theta}_0, \dots, \mathbf{\Theta}_r$	$\mathbf{\Omega}$
ChebNet	$\mathbf{\Delta}, \dots, \mathbf{\Delta}^r$	1	r	$\mathbf{\Theta}_0, \dots, \mathbf{\Theta}_r$	$[\mathbf{I}, \dots, \mathbf{I}]^\top$
GCN	$r = 1,\ \tilde{\mathbf{A}}$	1	1	$\mathbf{0}, \mathbf{\Theta}$	$[\mathbf{0}, \mathbf{I}]^\top$
S-GCN	$r = 1,\ \tilde{\mathbf{A}}$	1	L	$\mathbf{0}, \dots, \mathbf{0}, \mathbf{\Theta}$	$[\mathbf{0}, \dots, \mathbf{0}, \mathbf{I}]^\top$

Datasets

Reddit and *Flickr* are multiclass classification problems, *Yelp* and *PPI* are multilabel classification instances. In *Reddit*, the task is to predict communities of online posts based on user comments. In *Flickr* the task is image categorization based on the description and common properties of online images. In *Yelp* the objective is to predict business attributes based on customer reviews; the task of *PPI* consists in predicting protein functions from the interactions of human tissue proteins. *OGBN-Products* represents an Amazon product co-purchasing network (Bhatia et al., 2016) where the task is to predict the category of a product in a multi-class classification setup. *Wikipedia* links is a large-scale directed network of links between articles in the English version of Wikipedia; for the sake of our experiments, edge directions have been discarded. Statistics for all datasets are reported in Table 5.

Table 5. Summary of (s)ingle and (m)ulti-label dataset statistics. Wikipedia is used, with random features, for timing purposes only.

	n	$\lvert\mathcal{E}\rvert$	*Avg. Deg.*	d	Classes	Train / Val / Test
Wikipedia	12,150,976	378,142,420	62	100	2	100% / — / 100%
OGBN-Products	2,449,029	61,859,140	51	100	47	10% / 2% / 88%
Reddit	232,965	11,606,919	50	602	41(s)	66% / 10% / 24%
Yelp	716,847	6,977,410	10	300	100(m)	75% / 10% / 15%
Flickr	89,250	899,756	10	500	7(s)	50% / 25% / 25%
PPI	14,755	225,270	15	50	121(m)	66% / 12% / 22%

Triangle-based Operators

The triangle operator encodes the concept of *homophily* with a stronger acceptation with respect to the adjacency matrix: two nodes are connected by an edge only if they are both part of the same closed triad, i.e. if they are connected together and are both connected to the same node. Edge weights are proportional to the amount of triangles an edge belongs to, and they are normalised row-wise so to represent, for each node in a neighbourhood, its relative importance with respect to all the other neighbors.

This brings us to two considerations: first of all, the triangle operator is not carrying information related to nodes which were not already in the neighborhood. Secondly, it emphasizes the connections with those neighbors which are more related to our source node in virtue of the relationship described above. We can thus envision this operator being more useful in those graphs where this kind of relationship can be more discriminative within a neigborhood.

In Figure 2 we plot the normalized frequency distribution of intra-neighborhood standard deviation for the weights of triangle operators. It is interesting to notice the significantly different trends characterizing *Flickr* and *Reddit*, the two datasets where we experimentally observed triangle operators to bring, respectively, the largest and the smallest performance improvement.[2] *Flickr* tends to exhibit larger weight variations than other datasets, while, on the contrary, *Reddit* is the dataset where the smallest intra-neighborhood variation is observed. This suggests how, in *Flickr*, the triangle operator is

[2] *Flickr* – best *without* triangles is $(4, 0, 0)$: 0.508 ± 0.001, best *with* triangles is $(4, 0, 1)$: 0.514 ± 0.001. *Reddit* – best *without* triangles is $(4, 2, 0)$: 0.968 ± 0.000, best *with* triangles is $(4, 0, 1)$: 0.967 ± 0.000.

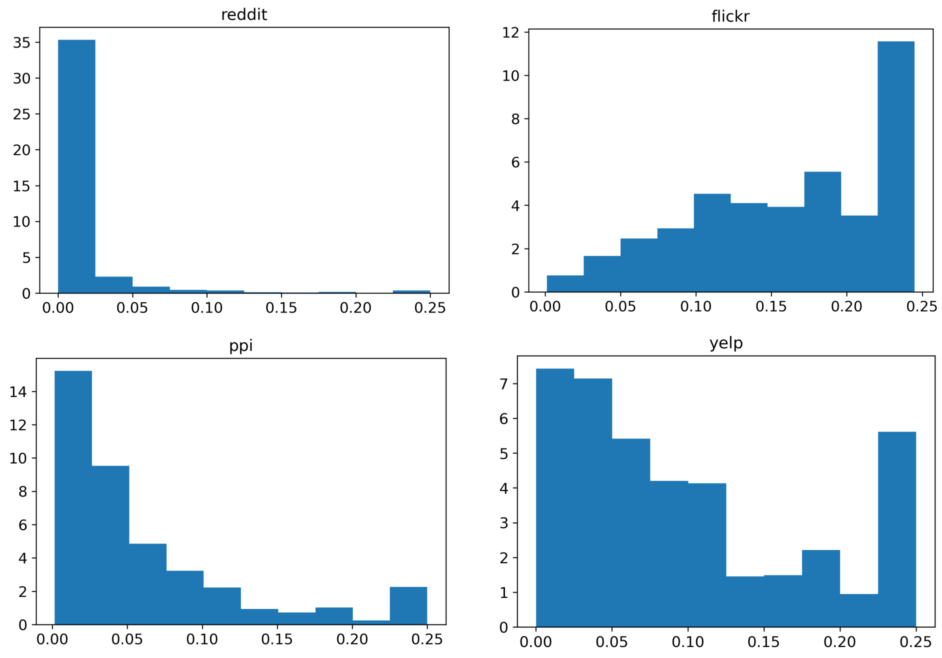

Figure 2. Normalized frequency distributions for row-wise variations on the diffusion weights of triangle operators over inductive datasets. Variations are measured as the standard deviation on the weight value over original neighborhoods from the test graph.

able to restrict feature aggregation to a subset of the original neighbors –those co-occurring in the larger number of triangles– while in *Reddit* it mostly boils down to uniform averaging, making this operator not much more expressive than a simple adjacency matrix.

For replicability we report that, in the computation of triangle operators for *PPI* and *Yelp*, we retained the self-loops already present in the original datasets. Investigations on how the presence of these edges affects the expressiveness of the triangle operator are left for future work.

Model Selection and Hyperparameter Tuning

Architectural and optimization hyperparameters (weight decay, dropout rate, batch size, learning rate, number of feedforward layers and units in inception and classification modules) were estimated using Bayesian optimization with a tree Parzen estimator surrogate function (Bergstra et al., 2011) over all inductive datasets. For each experiment we chose the set of hyperparameters matching the best average validation loss calculated over 5 runs. For the the transductive setting, we employed standard exhaustive search on a predefined hyperparameter grid, keeping the set of hyperparameters with minimum validation loss over a single run. The hyperparameter search space for the inductive setting and grid for the transductive one are described in Table 6. The estimated hyperparameters for each best SIGN configuration are reported in Table 7 for the inductive datasets and Table 8 for *OGBN-Products*.

Model convergence

In Figure 3 we plot the validation performance on OGBN-Products from the start of the training as a function of run time for ClusterGCN, GraphSaint and several SIGN configurations. SIGN converges to a better accuracy than other methods. It also exhibits much faster convergence than ClusterGCN and comparable speed than to GraphSAINT.

Implementation

All experiments, including timings, were run on an AWS p2.8xlarge instance, with 8 NVIDIA K80 GPUs, 32 vCPUs, a processor Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz and 488GiB of RAM. SIGN is implemented using Pytorch (Paszke et al., 2019).

Table 6. Hyperparameter search space/grid. Ranges in the form [*low*, *high*] and sampling distributions. *Inception Layers* and *Classification Layers* are the number of feedforward layers in, respectively, the representation part of the model (replacing Θ) and the classification part of the model (replacing Ω). The only exception is represented by *Yelp*: the Ω module was kept shallow (no hidden layers) to allow for lighter training and the left bound on the dropout interval was lowered to 0.0 as even mild values for this hyperparameter showed to be particularly detrimental in early experiments.

	TRANSDUCTIVE	INDUCTIVE	
HYPERPARAMETER	VALUES	SPACE	DISTRIBUTION
Learning Rate	0.0001, 0.001	[0.0001, 0.0025]	UNIFORM
Batch Size	4096, 8192, 16384	[128, 2048]	QUANTIZED UNIFORM
Dropout	0.5	[0.2, 0.8]	UNIFORM
Weight Decay	0.0, 0.00001	[0, 0.0001]	UNIFORM
Inception Layers	1	1, 2	—
Inception Units	256, 512	[128, 512]	QUANTIZED UNIFORM
Classification Layers	1	1, 2	—
Classification Units	256, 512	[512, 1024]	QUANTIZED UNIFORM
Activation	PRELU	RELU, PRELU	—

Table 7. Hyperparameters chosen for the best configuration of SIGN on inductive datasets.

HYPERPARAMETER	REDDIT	FLICKR	PPI	YELP
Learning Rate	0.00012278578238312588	0.0017230142114465549	0.0014386686616183625	0.0007758652074111595
Dropout	0.707328910934901	0.7608352140584778	0.3085607444207686	0.01
Weight Decay	9.176773905054599E-05	9.419820474221673E-05	3.2571631135664696E-06	4.452466189193362E-07
Batch Size	830	330	210	90
Inception Layers	1	2	2	2
Inception Units	460	465	315	320
Classification Layers	1	1	2	0
Classification Units	675	925	870	—
Activation	RELU	PRELU	RELU	RELU

Table 8. Hyperparameters chosen for the best configuration of SIGN on *OGBN-Product* dataset.

HYPERPARAMETER	OGBN-PRODUCTS
Learning Rate	0.0001
Dropout	0.5
Weight Decay	0.0001
Batch Size	4096
Inception Layers	1
Inception Units	512
Classification Layers	1
Classification Units	512
Activation	PRELU

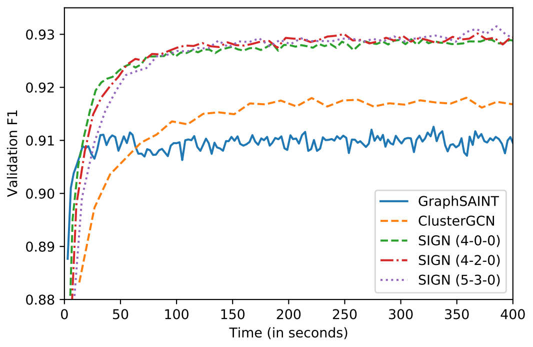

Figure 3. Convergence of different methods on *OGBN-Products*.