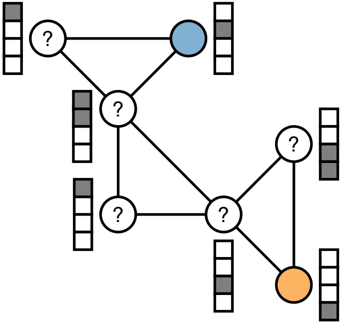
PushNet: Efficient and Adaptive Neural Message Passing

Julian Busch [1,2], Jiaxing Pi [1] and Thomas Seidl [2]

[1] Siemens Corporate Technology, Princeton, NJ, USA
jiaxing.pi@siemens.com
[2] Ludwig-Maximilians-Universität München, Munich, Germany
{busch, seidl}@dbs.ifi.lmu.de

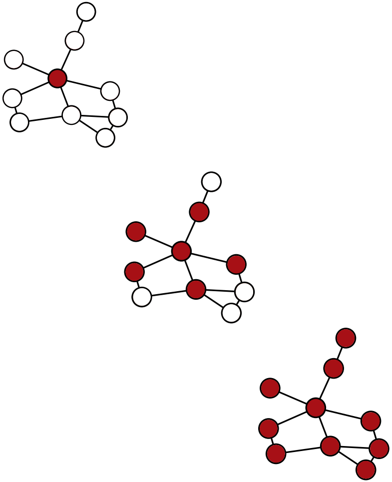
Introduction

- Semi-supervised node classification
  - Given
    - Graph $G = (V, E)$
    - Feature matrix $X \in \mathbb{R}^{n \times d}$
    - Incomplete label matrix $Y \in \mathbb{R}^{n \times c}$
  - Goal
    - Predict labels of unlabeled nodes
- Applications
  - Document classification in citation networks
  - User recommendations in social networks
  - Function prediction in protein interaction networks
  - And many more!



Neural Message Passing [1–15]

- Nodes repeatedly pull features from their neighbors:
  1. *Send messages* to all neighbors
  2. *Aggregate* incoming messages
  3. *Update* own feature vector
- k rounds of message passing gathers features from k-hop neighborhood
- A node's label is predicted from its updated feature vector



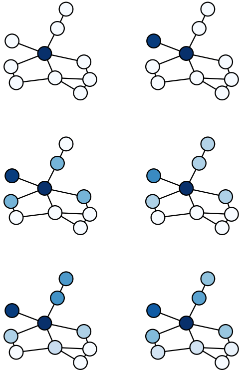
Motivation

- These methods are very successful
- However, they share several severe issues:
  - Not adaptive: In each iteration, all nodes send messages to all neighbors
    - Over-smoothing effect for multi-layer message passing [16, 17]
    - Inefficient: Only a few long-range dependencies are actually relevant
  - Restriction to k-hop neighborhoods
    - Include irrelevant, miss relevant neighbors
    - Need to specify number of message passing rounds k

Idea: Push information on demand instead of just pulling it from all neighbors!

Push-based Asynchronous Message Passing

- Main idea
  - Targeted propagation by updating nodes sequentially
- General procedure
  - Each node aggregates incoming messages until it is chosen to be updated
  - The chosen node updates its own state and pushes messages to all its neighbors
  - A node is only considered for updating if it aggregated enough new information

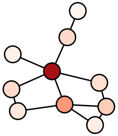


Challenge: Can we do this and still be efficient?

PushNet

- *Yes!* PushNet can be formulated as a single synchronous message passing step:

$$
\begin{aligned}
H^{(0)} &= f\left(X; \theta_f\right) && \in \mathbb{R}^{n \times h_1} \\
H^{(1)} &= P H^{(0)} && \in \mathbb{R}^{n \times h_1} \\
H^{(2)} &= g\left(H^{(1)}; \theta_g\right) && \in \mathbb{R}^{n \times h_2}
\end{aligned}
$$



  - The weights p_{ij} are given by *Approximate Personalized PageRank (APPR)* [18–20]
  - g and f are MLPs and act as learnable feature transformations/predictor

Combines the best of both worlds of synchronous/asynchronous message passing!

Extensions

- Multi-scale neighborhood aggregation
  - Locality hyper-parameter α in *APPR* controls the effective neighborhood size
  - Idea: Propagate over multiple scales instead of just one and aggregate the results
  - Benefits:
    - *Simplified hyper-parameter search*: Just combine a set of potential values
    - We empirically observe *improved classification accuracy and robustness*
- Model variants
  - Learnable feature transformations could be applied before and after propagation
  - General model: *PushNet*
  - No transformations before propagation
    - *Fast, propagated features can be cached!*
    - *PushNet-PTP*, *PushNet-PP*
  - Propagation of predicted class labels
    - *PushNet-TPP*

Experimental Setup

- Evaluation on 5 document classification benchmark datasets
- Comparison with 7 SOTA neural message passing algorithms
- Rigorous evaluation setup [21, 22]
  - Grid search to determine best hyper-parameters for each model
  - Results from 100 independent runs

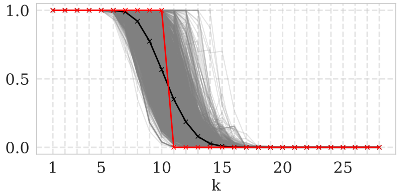
	$\|V\|$	$\|E\|$	d	c	avgSP	maxSP
CiteSeer	2120	3679	3703	6	9.33	28
Cora	2485	5069	1433	7	6.31	19
PubMed	19717	44324	500	3	6.34	18
Coauthor CS	18333	81894	6805	15	5.43	24
Coauthor Physics	34493	247962	8415	5	5.16	17

Node Classification Accuracy

	CiteSeer	Cora	PubMed	Coauthor CS	Coauthor Physics
GCN [4]	72.82 ± 1.48	81.07 ± 1.43	78.29 ± 1.48	91.64 ± 0.62	93.42 ± 0.63
GAT [7]	73.82 ± 1.35	82.12 ± 1.41	78.21 ± 1.60	90.20 ± 0.75	93.43 ± 0.50
JK-GCN [17]	71.09 ± 1.66	79.57 ± 1.63	77.23 ± 2.01	91.60 ± 0.54	93.49 ± 0.56
JK-GAT [17]	71.76 ± 1.27	80.10 ± 1.52	77.59 ± 2.25	92.20 ± 0.43	*o.o.m.*
SGC [15]	73.91 ± 1.30	80.13 ± 2.15	77.00 ± 1.78	91.27 ± 0.58	*o.o.m.*
GIN [13]	70.81 ± 1.61	80.24 ± 1.54	77.19 ± 1.75	91.46 ± 0.54	93.79 ± 0.49
APPNP [22]	74.36 ± 1.44	83.58 ± 1.03	79.61 ± 2.98	91.10 ± 1.12	93.96 ± 0.45
PushNet	75.08 ± 0.99	84.12 ± 1.08	79.80 ± 1.39	92.40 ± 0.52	94.01 ± 0.53
PushNet-PTP	$\mathbf{75.19 \pm 1.15}$	83.41 ± 1.24	$\mathbf{80.22 \pm 1.27}$	92.37 ± 0.40	93.97 ± 0.48
PushNet-PP	75.17 ± 1.32	81.52 ± 1.40	77.52 ± 2.05	91.04 ± 0.76	93.67 ± 0.55
PushNet-TPP	75.01 ± 1.11	$\mathbf{84.23 \pm 1.26}$	80.10 ± 1.33	$\mathbf{92.54 \pm 0.34}$	$\mathbf{94.09 \pm 0.47}$

Node Classification Accuracy

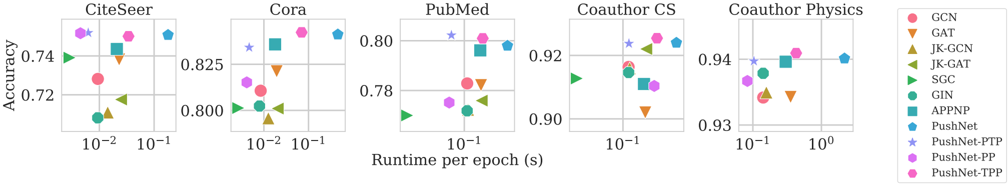
- *PushNet* already outperforms all competitors on all datasets
  - *PushNet-TPP* performs best overall
  - *PushNet-PTP* improves performance on *CiteSeer* and *PubMed*
  - *PushNet-PP* is remarkably competitive, even though it performs no learnable feature transformation at all!
- All improvements are statistically significant
- Improvements due to *individually adapted neighborhoods for each node*



Fraction of k-neighbors included in *APPR*-neighborhood for each node

- Solid: Average over all nodes
- Red: 10-neighborhood

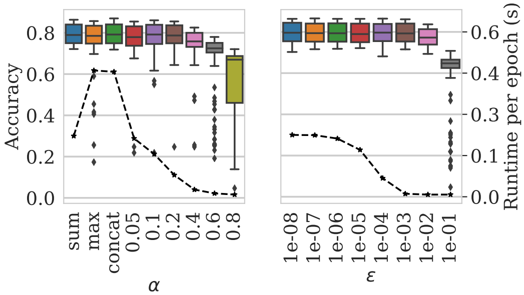
Runtime



- *SGC* [15] is fastest but less accurate than most competitors
- *PushNet-PP* is second fastest, followed by *PushNet-PTP* which offers a good tradeoff between runtime and accuracy
- *PushNet* and *PushNet-TPP* are slower but provide superior accuracy

Ablation Study

- Best single scales: $\alpha \in \{0.05, 0.1, 0.2\}$
- Best aggregation: *sum*
  - Efficient: Runtime determined by smallest single α considered
  - Improved results over best single α
- Sparsity
  - Parameter ε in *APPR* to exclude small propagation weights
  - Accuracy remains very stable
  - Intuitive to set, just set large enough for available GPU memory!



Conclusion

- Novel asynchronous approach to neural message passing
- Efficient aggregation over adaptive node neighborhoods
  - No restriction to k-neighborhoods
  - Effective handling of long-range dependencies
- Multi-scale feature aggregation
  - Improved performance
  - Simplified hyper-parameter selection
- Different variants of our base model
  - Allow for varying trade-offs between efficiency and accuracy
- Hyper-parameters relatively easy to set

Thank you! Questions?

References I

[1] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *NeurIPS*, 2015.

[2] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *ICLR*, 2016.

[3] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 2016.

[4] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 2017.

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *ICML*, 2017.

References II

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[8] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *ICLR*, 2018.

[9] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *TKDD*, 2019.

[10] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *AAAI*, 2019.

[11] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *ESWC*, 2018.

References III

[12] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

[13] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019.

[14] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[15] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *ICML*, 2019.

[16] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI*, 2018.

[17] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *ICML*, 2018.

References IV

[18] Glen Jeh and Jennifer Widom. Scaling personalized web search. *WWW*, 2003.

[19] Pavel Berkhin. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Mathematics*, 2006.

[20] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S Mirrokni, and Shang-Hua Teng. Local computation of pagerank contributions. *International Workshop on Algorithms and Models for the Web-Graph*, 2007.

[21] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *NeurIPS Relational Representation Learning Workshop*, 2018.

[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *ICLR*, 2019.