SIGN: Scalable Inception Graph Neural Networks

Fabrizio Frasca et al.

February 28, 2025

Twitter, Imperial College London

Presented by: George Giapitzakis

Graph Neural Networks: A Brief Overview

- ➔ Graph Neural Networks (GNNs) extend deep learning methods to graph-structured data.
- ➔ Applications: social networks, chemistry, physics, recommender systems, etc.
- ➔ Standard GNNs often face scalability challenges when dealing with large graphs.

Scaling Challenges in GNNs

- ➔ Many GNNs rely on neighbor sampling or multi-layer message passing.
- ➔ Deep architectures can suffer from over-smoothing and high computational cost.
- ➔ Sampling may introduce bias and inefficiency in both training and inference.

Motivation: How can we design a GNN that scales to web-scale graphs without these drawbacks?

Motivation for a Sampling-Free Approach

- ➔ **Scalability:** Web-scale graphs (e.g., billions of edges) require methods whose cost is independent of the graph structure during training/inference.
- ➔ **Expressiveness:** Retaining performance without sacrificing the ability to capture local graph structure.
- ➔ **Efficiency:** Avoiding iterative neighborhood sampling can greatly reduce both runtime and memory usage.

Inspiration from Inception Modules

- ➔ In CNNs, Inception modules combine filters of various sizes to capture multi-scale features.
- ➔ SIGN adapts this idea for graphs: using multiple precomputed diffusion operators as "filters" to capture different scales of neighborhood information.
- ➔ This approach bypasses the need for deep stacking of convolutional layers.

The SIGN Architecture

- SIGN stands for **S**calable **IN**ception **G**raph **N**eural network.
- Core idea: precompute multiple graph diffusion operators and combine their outputs.
- The model is defined as:

$$\boldsymbol{Z} = \sigma\Big([\boldsymbol{X}\boldsymbol{\Theta}_0,\ \boldsymbol{A}_1\boldsymbol{X}\boldsymbol{\Theta}_1,\ \ldots,\ \boldsymbol{A}_r\boldsymbol{X}\boldsymbol{\Theta}_r]\Big)$$

$$\boldsymbol{Y} = \xi(\boldsymbol{Z}\boldsymbol{\Omega})$$

- $\boldsymbol{X}$: Input node features; $\boldsymbol{A}_i$: Diffusion operators; $\boldsymbol{\Theta}_i$, $\boldsymbol{\Omega}$: Learnable parameters.

Precomputation of Diffusion Operators

➔ Diffusion operators $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_r$ can be based on:

- Normalized adjacency (GCN-style)
- Personalized PageRank (PPR)
- Triangle-induced (motif-based)

➔ These operators are computed **once** before training.

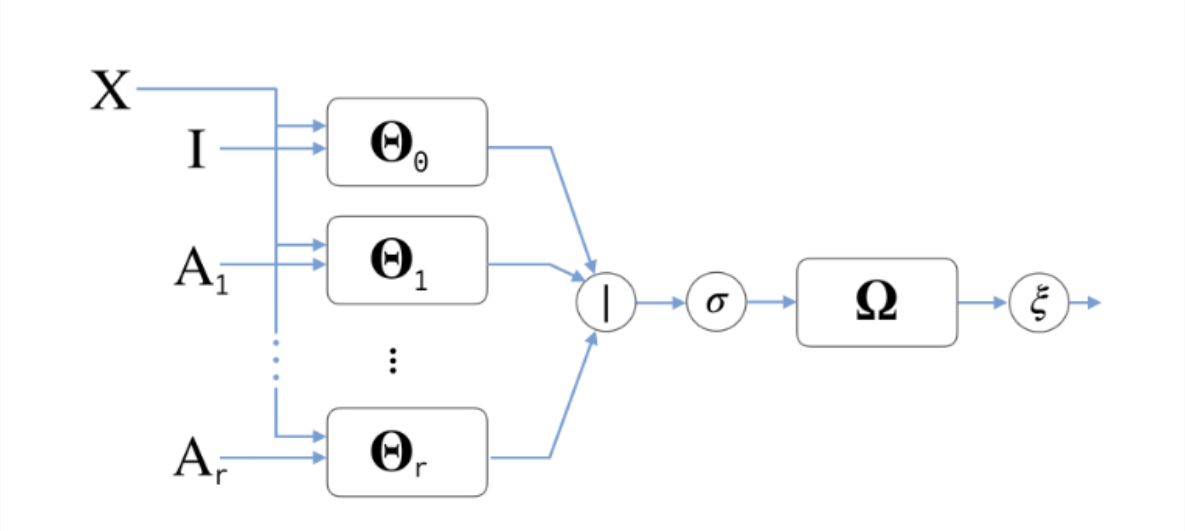

Figure 1: The SIGN architecture for r generic graph filtering operators. $|$ denotes the concatenation operation.

Relation to Inception Modules in CNNs

- ➔ Similar to using multiple kernel sizes in CNNs, SIGN uses different operators to capture various scales.
- ➔ Each operator acts like a convolution filter with a different receptive field.
- ➔ This multi-scale feature extraction is key to handling heterogeneous graph structures.

Computational Complexity

- **Preprocessing:** $O(r|E|d)$, where r is the number of operators, $|E|$ is the number of edges, and d the feature dimension.
- **Training/Inference:** Complexity becomes $O(rL_{ff}Nd^2)$ (similar to an MLP), independent of the graph structure.
- Key advantage: Avoids repeated neighborhood expansion and reduces redundant computations.

Experimental Setup

- **Datasets:**
  - *Inductive:* Reddit, Flickr, Yelp, PPI.
  - *Transductive:* ogbn-products, ogbn-papers100M.
  - *Scalability:* Wikipedia links (with synthetic features).
- **Baselines:** GCN, GraphSAGE, ClusterGCN, GraphSAINT, S-GCN.
- **Metrics:** F1 scores, accuracy, and runtimes (preprocessing, training, and inference).

Key results: Accuracy

- ➔ SIGN achieves state-of-the-art results on ogbn-papers100M (over 110M nodes, 1.6B edges).
- ➔ Outperforms sampling-based methods in both accuracy and speed on several benchmarks.

	Reddit	Flickr	PPI	Yelp
GCN [34]	0.933±0.000	0.492±0.003	0.515±0.006	0.378±0.001
FastGCN [11]	0.924±0.001	**0.504±0.001**	0.513±0.032	0.265±0.053
Stochastic-GCN [12]	**0.964±0.001**	0.482±0.003	**0.963±0.010**	**0.640±0.002**
AS-GCN [30]	0.958±0.001	**0.504±0.002**	0.687±0.012	—
GraphSAGE [24]	0.953±0.001	0.501±0.013	0.637±0.006	**0.634±0.006**
ClusterGCN [13]	0.954±0.001	0.481±0.005	0.875±0.004	0.609±0.005
GraphSAINT [64]	**0.966±0.001**	**0.511±0.001**	**0.981±0.004**	**0.653±0.003**
S-GCN [59]	0.949±0.000	0.502±0.001	0.892±0.015	0.358±0.006
SIGN	**0.968±0.000**	**0.514±0.001**	**0.970±0.003**	0.631±0.003
(p, s, t)	$(4, 2, 0)$	$(4, 0, 1)$	$(2, 0, 1)$	$(2, 0, 1)$

Figure 2: Micro-averaged $F1$ scores. For SIGN, only the best performing configuration is shown. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

Ablation Study: Operator Combinations

- ➔ Evaluated different configurations of simple, PPR-based, and triangle-based operators.
- ➔ Results indicate that dataset-specific operator selection is crucial:

	Reddit	Flickr	PPI	Yelp
SIGN(2,0,0)	0.966±0.003	0.503±0.003	0.965±0.002	0.623±0.005
SIGN(2,0,1)	0.966±0.000	0.510±0.001	**0.970±0.003**	**0.631±0.003**
SIGN(2,2,0)	0.967±0.000	0.495±0.002	0.964±0.003	0.617±0.005
SIGN(4,0,0)	0.967±0.000	0.508±0.001	0.959±0.002	0.623±0.004
SIGN(4,0,1)	0.967±0.000	**0.514±0.001**	0.965±0.003	0.623±0.004
SIGN(4,2,0)	**0.968±0.000**	0.500±0.001	0.930±0.010	0.618±0.004
SIGN(4,2,1)	0.967±0.000	0.508±0.002	0.969±0.001	0.620±0.004

Figure 3: Impact of various operator combinations on inductive datasets. Best results are in bold.

Ablation Study: Operator Combinations

	Training	*Validation*	*Test*
MLP	84.03±0.93	75.54±0.14	61.06±0.08
Node2Vec	93.39±0.10	90.32±0.06	72.49±0.10
S-GCN(L=5)	92.54±0.09	91.38±0.07	74.87±0.25
ClusterGCN	93.75±0.13	92.12±0.09	**78.97±0.33**
GraphSAINT	92.71±0.14	91.62±0.08	**79.08±0.24**
SIGN(3,0,0)	96.21±0.31	**92.99±0.05**	76.52±0.14
SIGN(3,0,1)	**96.46±0.29**	92.93±0.04	75.73±0.20
SIGN(3,3,0)	**96.87±0.23**	**93.02±0.04**	77.13±0.10
SIGN(5,0,0)	95.99±0.69	92.98±0.18	76.83±0.39
SIGN(5,3,0)	**96.92±0.46**	**93.10±0.08**	**77.60±0.13**

Figure 4: Results on `ogbn-papers100M`. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

Runtime Analysis

- ➔ SIGN's preprocessing takes longer than some sampling methods but is executed only once.
- ➔ Training and inference times are drastically reduced.
- ➔ Convergence is faster, leading to overall time savings on large graphs.

	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

Figure 5: Preprocessing, Training, and Inference times.

Conclusion

- ➔ **Scalability:** SIGN scales to web-scale graphs without relying on sampling.
- ➔ **Efficiency:** Precomputing diffusion operators shifts the heavy computation offline.
- ➔ **Performance:** Competitive or state-of-the-art accuracy across a range of benchmarks.
- ➔ **Simplicity vs. Expressiveness:** A single graph convolutional layer with multiple diffusion operators can outperform deeper, sampling-based architectures.

Discussion and Future Directions

- ➔ **Depth vs. Width:** Results suggest that wide (multi-operator) shallow architectures are effective.
- ➔ **Extensions:**
  - Incorporating more expressive local operators (e.g., attention mechanisms).
  - Exploring higher-order network motifs and temporal motifs.
  - Further optimization of precomputation for distributed systems.

Questions?

Thank you!

Any Questions?