南京大学
NANJING UNIVERSITY

PASA Lab

# GraphSAGE论文阅读

报告人：王云攀

时间：2019.12.6

# 汇报内容

PASA Lab

Abstract

1 Introduction

2 Related work

3 Proposed method: GraphSAGE[1]

4 Experiments

1. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *NIPS*, *2017-Decem*(Nips), 1025–1035. Retrieved from http://arxiv.org/abs/1706.02216

# Abstract

PASA Lab

## Content

- Background
  - Low-dimensional embeddings  of nodes in large graphs have proved useful
- Problem:
  - Most existing approaches require that all nodes in the graph are present during training of the embeddings: these approaches are transductive and do not naturally generalize to unseen nodes
- Solution
  - Present GraphSAGE, a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data.
  - Learn a function that generates embeddings by sampling and aggregating features from a node’ s local neighborhood.
- Result
  - Outperforms strong baselines on three inductive node-classification benchmarks.
  - Dataset:  citation and Reddit post data, a multi-graph dataset of protein-protein interactions

# 1 Introduction

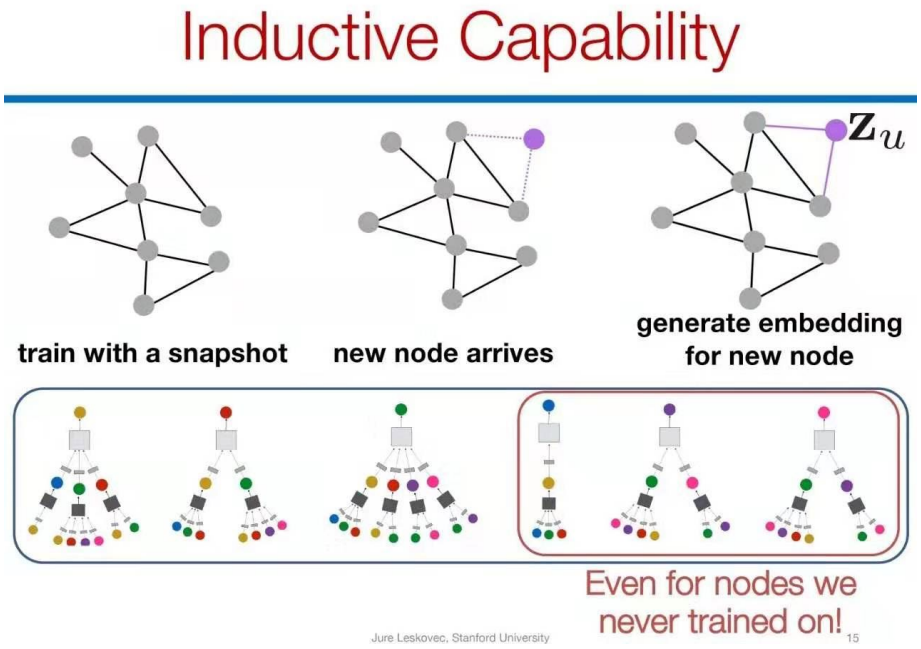
PASA Lab

## 1.1 Previous work about node-embedding: [5, 11, 28, 35, 36]

- Ideas
  - use dimensionality reduction techniques and distill the high-dimensional information about a node’ s graph neighborhood into a dense vector embedding.
- Problem
  - focused on embedding nodes from a single fixed graph. And many real-world applications require embedding to be quickly generated for unseen nodes, or entirely new (sub)graphs.
- Solution
  - increase the inductive capability.

PASA Lab

## 1.2 Inductive Approaches




- Inductive
  - Advantages:
    - high-throughput, production machine learning systems, which operate on evolving graphs and constantly encounter unseen nodes.
  - Usage:
    - facilitate generalization across graphs with the same form of features.
  - Why difficult?
    - Generalizing to unseen nodes requires "aligning" newly observed subgraphs to the node embedding that the algorithm has already optimized on.
- Transductive [5, 11, 23, 28, 35, 36, 37, 39]
  - Majority of them using matrix-factorization-based objectives.
  - How to be Inductive?
    - They can be modified to operate in an inductive setting, but computationally expensive, requiring additional rounds of gradient descent before new predictions can be made.

# 1 Introduction

PASA Lab

## 1.3 Present work

- GraphSAGE(SAmple and aggreGatE), a general framework.
- Incorporate node features, the topological structure of each node' s neighborhood as well as the distribution of node features in the neighborhood.

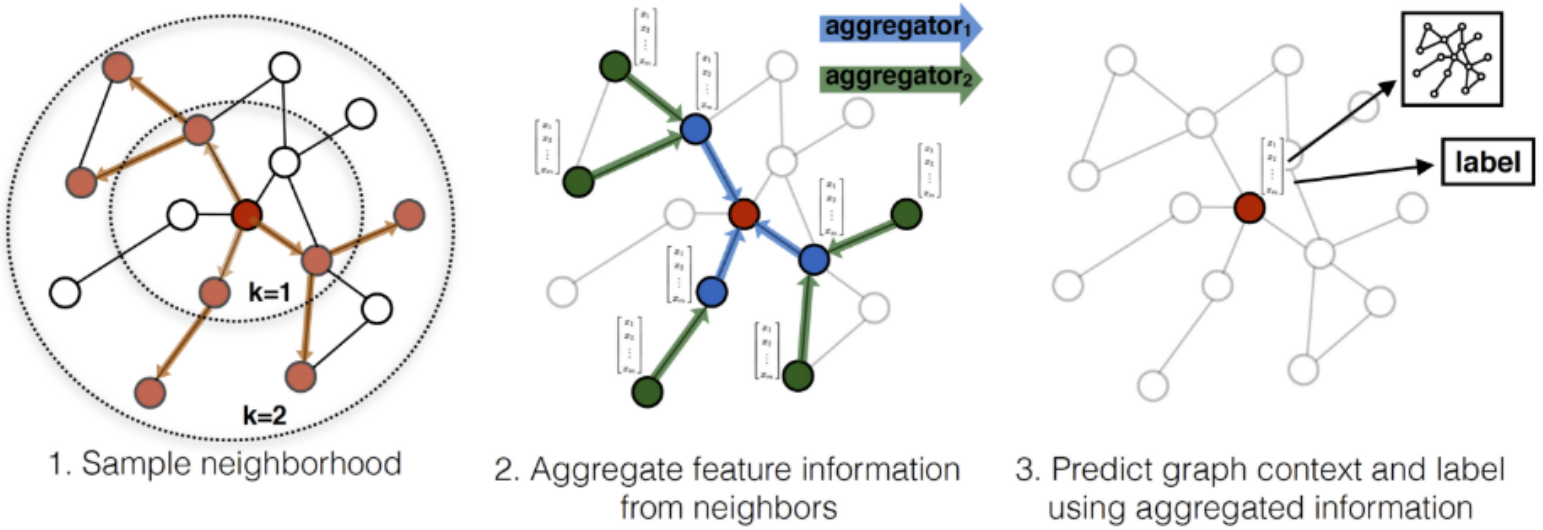



Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

# 1 Introduction

PASA Lab

## 1.3 Present work

- Creative:
  - Instead of training a distinct embedding vector for each node, we train a set of aggregator functions that learn to aggregate feature information from a node’ s local neighborhood
  - Each aggregator function aggregates information from a different number of hops, or search depth, away from a given node.
  - At test, use trained system to generate embedding for entirely unseen nodes by applying the learned aggregation functions.
- Benchmarks
  - Three node-classification benchmarks
  - Two evolving document graphs (citation data and Reddit post data) and a multi-graph generalization experiment based on a dataset of protein-protein interactions(predicting protein functions.)
- Results
  - Improves classification F1-scores by an average of 51% compared to using node features alone.
  - Compared to transductive baseline, taking ~100x faster to run on unseen nodes.

# 2 Related work

PASA Lab

## Content

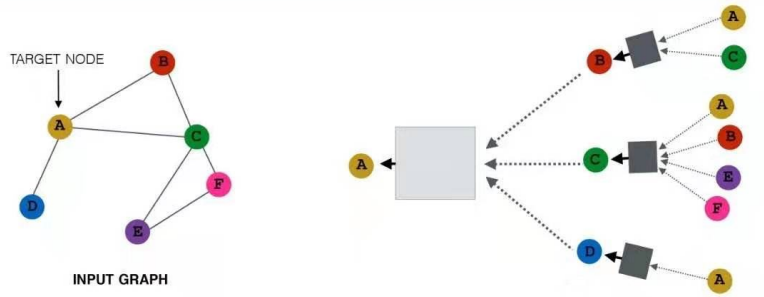
- Node-embedding approaches: Factorization-based embedding approaches
  - Random walk statistics and matrix factorization-based [5, 11, 28, 35, 36]
- General supervised approaches to learning over graphs
  - Kernel-based approaches [32]
  - Supervised learning over graph structures [7, 10, 21, 31]
- Recent advancements in applying convolutional neural networks to graph-structured data
  - Several convolutional neural network architectures [4, 9, 8, 17, 24]
  - Our approach is closely related to the graph convolutional network, GCN, semi-supervised learning in a transductive setting.

# 3 Proposed method: GraphSAGE

PASA Lab

## Key idea

### Graph Neural Networks



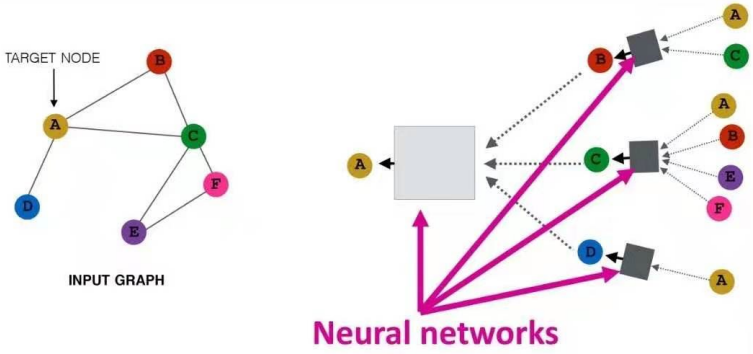

Each node defines a computation graph

- Each edge in this graph is a transformation/aggregation function

Scarselli et al. 2005. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*.



### Graph Neural Networks




**Intuition:** Nodes aggregate information from their neighbors using neural networks

Inductive Representation Learning on Large Graphs. W. Hamilton, R. Ying, J. Leskovec. NIPS, 2017.

# 3 Proposed method: GraphSAGE

PASA Lab

- 3.1 Embedding generation (i.e., forward propagation) algorithm
- 3.2 Learning the parameters of GraphSAGE
- 3.3 Aggregator Architectures

# 3 Proposed method: GraphSAGE

PASA Lab

## 3.1 Embedding generation (i.e forward propagation) algorithm

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

- **Relation to the Weisfeilier-Lehman Isomorphism Test (Test graph isomorphism)**
  - Set K = |V|
  - Set the weight matrices as the identity
  - Use an appropriate hash function as an aggregator (with no non-linearity)
- **Neighborhood definition**
  - Uniformly sample a fixed-size set of neighbors
  - Draw different uniform samples at each iteration, k.
  - In the k iteration, the fix-sized of neighborhood is Sk.
  - K, S1, S2,… Sk, user-specified constants.
  - Our approach: K =2, S1 * S2 <= 500

# 3 Proposed method: GraphSAGE

## 3.1 Embedding generation (i.e forward propagation) algorithm

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

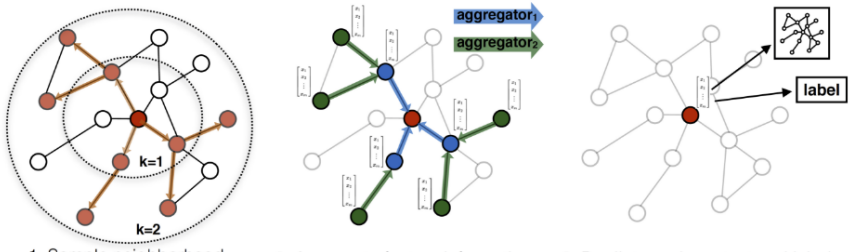



Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

# 3 Proposed method: GraphSAGE

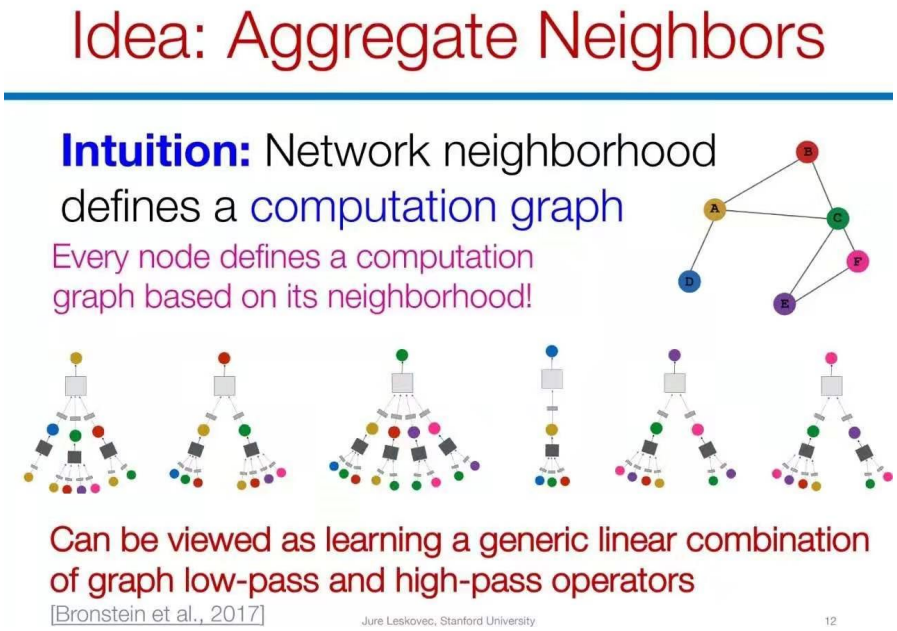
## 3.1 Embedding generation (i.e forward propagation) algorithm

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$



# 3 Proposed method: GraphSAGE

PASA Lab

## 3.1 Embedding generation (i.e forward propagation) algorithm

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

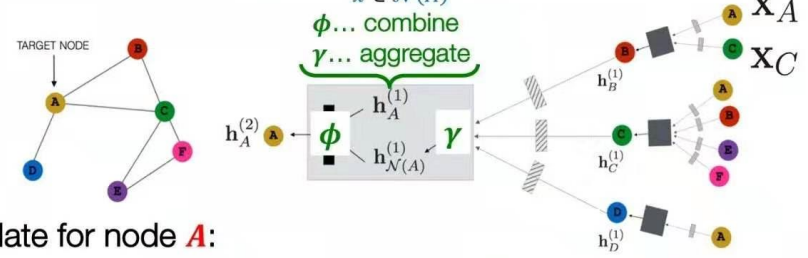
$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

[NIPS '17]

## Our Approach: GraphSAGE

$$\phi(\gamma_{x \in \mathcal{N}(A)}(h_x))$$

$\phi$... combine
$\gamma$... aggregate




Update for node $A$:

$$h_A^{(k+1)} = \sigma\left(W^{(k)} h_A^{(k)}, \gamma_{x \in \mathcal{N}(A)}\left(\sigma(Q^{(k)} h_x^{(k)})\right)\right)$$

- $k+1^{st}$ level embedding of node $A$
- Transform $A$'s own embedding from level $k$
- Transform and aggregate embeddings of neighbors $n$

- $h_A^{(0)}$ = attributes $X_A$ of node $A$, $\sigma(\cdot)$ is a sigmoid activation function

# 3 Proposed method: GraphSAGE

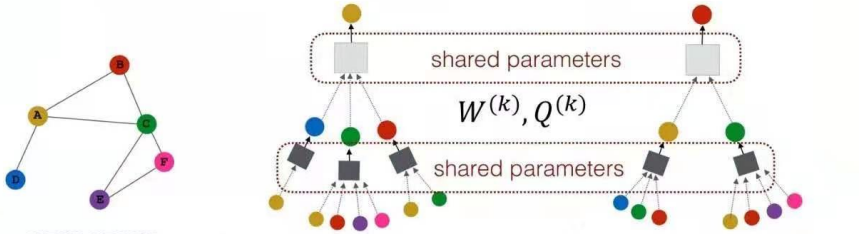
PASA Lab

## 3.2 Learning the parameters of GraphSAGE

[NIPS '17]

### GraphSAGE: Training

- Aggregation parameters are shared for all nodes
- Number of model parameters is independent of |V|
- Can use different loss functions:
  - Classification/Regression: $\mathcal{L}(h_A) = ||y_A - f(h_A)||^2$
  - Pairwise Loss: $\mathcal{L}(h_A, h_B) = \max(0, 1 - dist(h_A, h_B))$




Jure Leskovec, Stanford University 14

- Symbol Definition
  - $z_v$, the output representations, u in V。
  - Zv, a node that co-occurs near u on fixed-length random walk
  - Pn(v): a negative sampling distribution
  - Q: the number of negative samples
- Content
  - Encourage nearby nodes to have similar representations
  - Enforce that the representations of disparate nodes are highly distinct

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right)$$

# 3 Proposed method: GraphSAGE

PASA Lab

## 3.3 Aggregator Architectures

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 **for** $k = 1...K$ **do**
3  **for** $v \in \mathcal{V}$ **do**
4  $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
5  $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
6  **end**
7  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
8 **end**
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

- **Mean aggregator**
  - Content
    - Take the elementwise mean of the vectors
  - Contrast
    - Between old convolutional aggregator and out is the concatenation operation in line 5 of Algorithm1

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$$

# 3 Proposed method: GraphSAGE

PASA Lab

## 3.3 Aggregator Architectures

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

- **LSTM aggregator**
  - Content
    - Based on LSTM architecture
  - Contrast
    - Not inherently symmetric (i.e. they are not permutation invariant.

# 3 Proposed method: GraphSAGE

PASA Lab

## 3.3 Aggregator Architectures

**Algorithm 1:** GraphSAGE embedding generation (i.e., forward propagation) algorithm

**Input :** Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Output :** Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
**for** $k = 1...K$ **do**
  **for** $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  **end**
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
**end**
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

- **Pooling aggregator**
  - Content
    - an elementwise max-pooling operation
  - Contrast
    - Symmetric and trainable

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\}),$$

# 4 Experiment

PASA Lab

## 4.1 Inductive benchmarks

- Four baselines
  - A random classifer
  - A logistic regression feature-based classifier(that ignores graph structure)
  - The Deep Walk algorithm
  - A concatenation of the raw features and DeepWalk embedding.

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

| | Citation | | Reddit | | PPI | |
|---|---|---|---|---|---|---|
| Name | Unsup. F1 | Sup. F1 | Unsup. F1 | Sup. F1 | Unsup. F1 | Sup. F1 |
| Random | 0.206 | 0.206 | 0.043 | 0.042 | 0.396 | 0.396 |
| Raw features | 0.575 | 0.575 | 0.585 | 0.585 | 0.422 | 0.422 |
| DeepWalk | 0.565 | 0.565 | 0.324 | 0.324 | — | — |
| DeepWalk + features | 0.701 | 0.701 | 0.691 | 0.691 | — | — |
| GraphSAGE-GCN | 0.742 | 0.772 | **0.908** | 0.930 | 0.465 | 0.500 |
| GraphSAGE-mean | 0.778 | 0.820 | 0.897 | 0.950 | 0.486 | 0.598 |
| GraphSAGE-LSTM | 0.788 | 0.832 | **0.907** | **0.954** | 0.482 | **0.612** |
| GraphSAGE-pool | **0.798** | **0.839** | 0.892 | 0.948 | **0.502** | 0.600 |
| % gain over feat. | 39% | 46% | 55% | 63% | 19% | 45% |

# 4 Experiment

PASA Lab

## 4.2 Runtime and parameter sensitivity

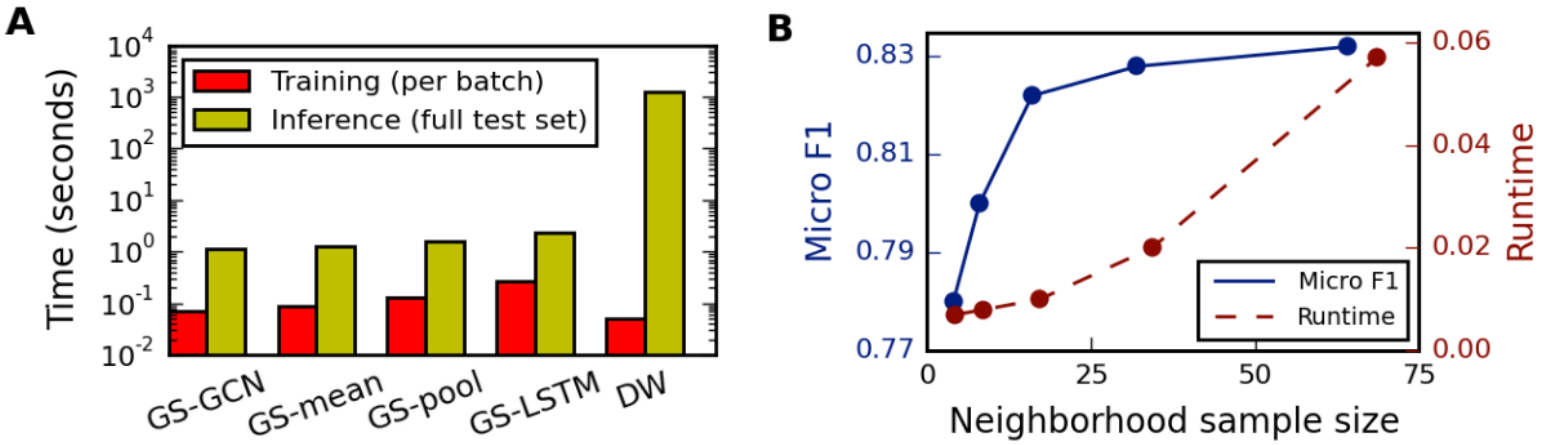



Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each depth for $K = 2$ with $S_1 = S_2$ (on the citation data using GraphSAGE-mean).

# 谢 谢