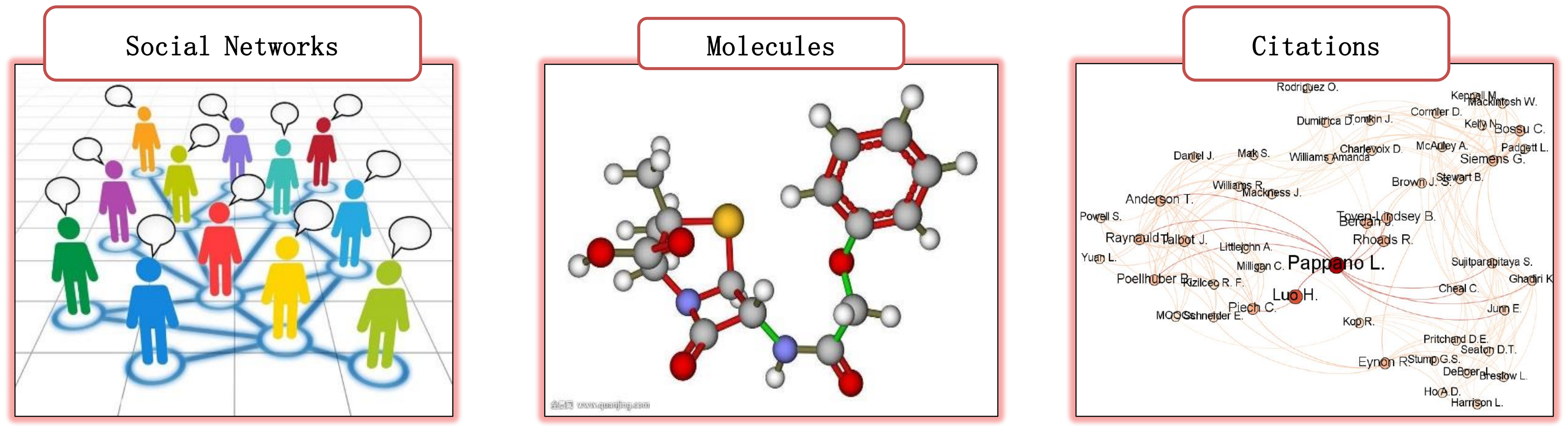
Adaptive Sampling Towards Fast Graph Representation Learning

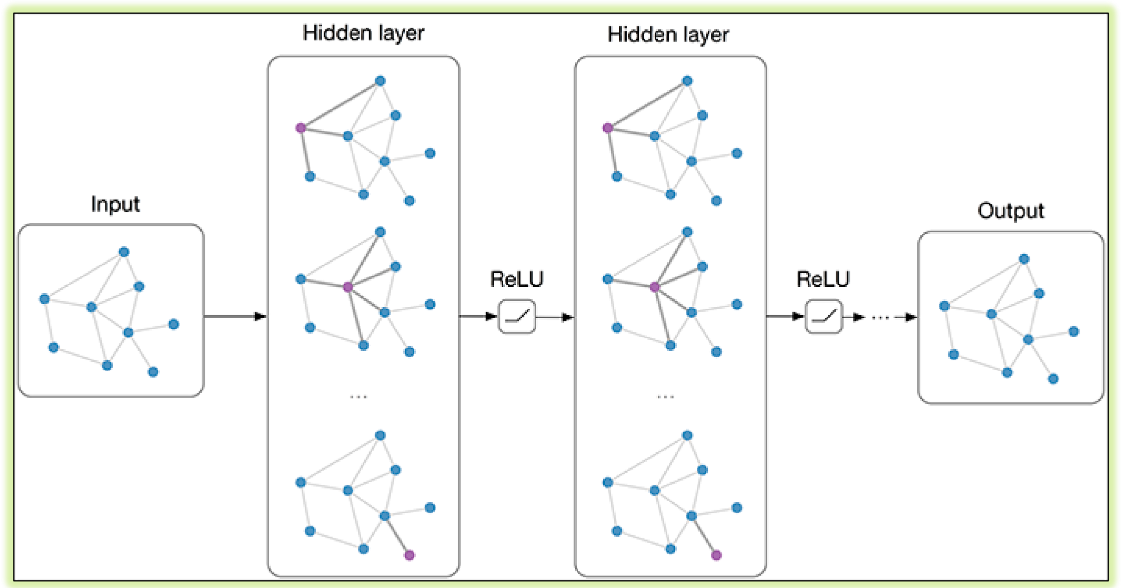
Wenbing Huang[1], Tong Zhang[2], Yu Rong[1], Junzhou Huang[1]
[1]Tencent AI Lab, [2]Australian National University

Graph Convolution Networks (GCNs)



- In many cases, data form kinds of **Graphical Structures.**
- To perform classification on nodes, certain machinery should be developed to use their connections. Thus, we have **GCNs.**

Kipf, T. N., & Welling, M. ICLR-2017



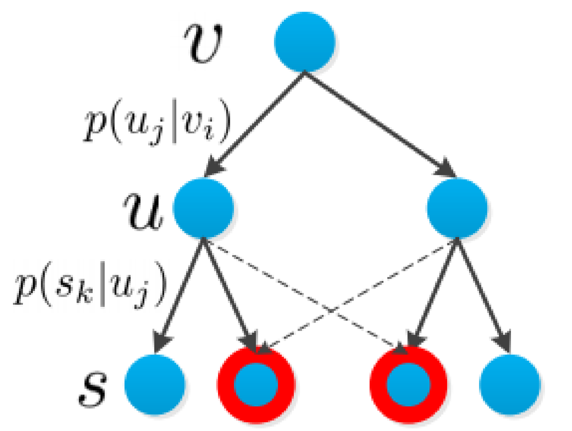
$$h^{(l+1)}(v_i) = \sigma\left(\sum_{j=1}^{N} \hat{a}(v_i, u_j) h^{(l)}(u_j) W^{(l)}\right)$$

Relu; Hidden features; Normalized adjacent Matrix; Parameters

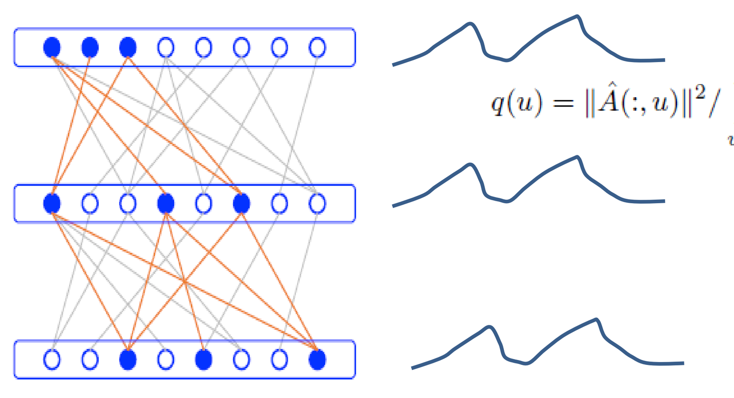
- **Issue:** GCNs incur heavy cost both in computation and memory due to the uncontrollable neighborhood expansion across layers.

Accelerating GCNs: GraphSAGE and FastGCN

GraphSAGE (node-wise)



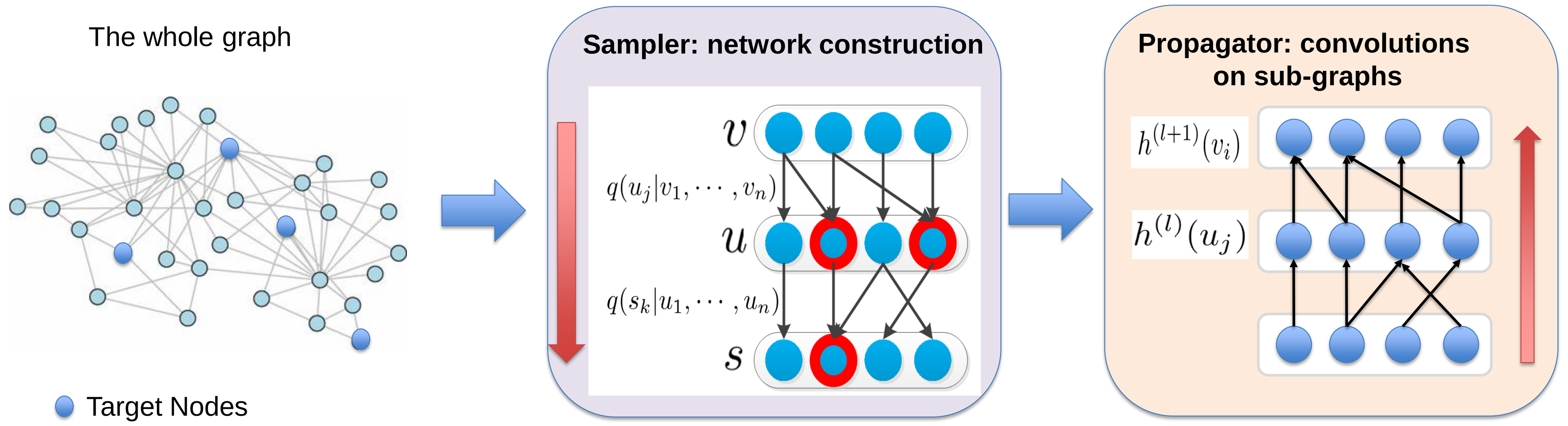
FastGCN (layer-independent)



$$q(u) = \|\hat{A}(:,u)\|^2 / \sum_{u' \in V} \|\hat{A}(:,u')\|^2, \quad u \in V$$

Our Solution: AS-GCN

- Our basic idea: 1) **Sampler** to construct the neural network of subgraphs by top-down layer-dependent sampling; 2) **Propagator** to perform the forward pass on the constructed network.



- Why does this idea make sense?

GCN: $h^{(l+1)}(v_i) = \sigma\left(\sum_{j=1}^{N} \hat{a}(v_i, u_j) h^{(l)}(u_j) W^{(l)}\right)$

Rewrite the sum as the expectation

GCN: $h^{(l+1)}(v_i) = \sigma_{W^{(l)}}(N(v_i)\mathbb{E}_{p(u_j|v_i)}[h^{(l)}(u_j)])$

Estimate the expectation by sampling

$$h^{(l+1)}(v_i) = \sigma_{W^{(l)}}(N(v_i)\hat{\mu}_p(v_i))$$

with $\hat{\mu}_p(v_i) = \frac{1}{n}\sum_{j=1}^{n} h^{(l)}(\hat{u}_j), \quad \hat{u}_j \sim p(u_j|v_i).$

Estimate the expectation by importance sampling

$$h^{(l+1)}(v_i) = \sigma_{W^{(l)}}(N(v_i)\mathbb{E}_{q(u_j|v_1,\cdots,v_n)}[\frac{p(u_j|v_i)}{q(u_j|v_1,\cdots,v_n)} h^{(l)}(u_j)]).$$

$$h^{(l+1)}(v_i) = \sigma_{W^{(l)}}(N(v_i)\hat{\mu}_p(v_i))$$

with $\hat{\mu}_q(v_i) = \frac{1}{n}\sum_{j=1}^{n} \frac{p(\hat{u}_j|v_i)}{q(\hat{u}_j|v_1,\cdots,v_n)} h^{(l)}(\hat{u}_j), \quad \hat{u}_j \sim q(\hat{u}_j|v_1,\cdots,v_n).$

- Formulating the sampler by variance reduction

A good estimator should reduce the variance caused by the sampling process

Variance: $\mathrm{Var}_q(\hat{\mu}_q(v_i)) = \frac{1}{n}\mathbb{E}_{q(u_j)}[\frac{(p(u_j|v_i)|h^{(l)}(u_j)| - \mu_q(v_i)q(u_j))^2}{q^2(u_j)}].$

Optimal Sampler

$$q^*(u_j) = \frac{p(u_j|v_i)|h^{(l)}(u_j)|}{\sum_{j=1}^{N} p(u_j|v_i)|h^{(l)}(u_j)|}.$$

Unfortunately, The sampler is computed based on the hidden features that are obtained only after the previous layers are sampled

Replace hidden features with self-dependent functions, and explicitly reduce the variance by adding it to the loss

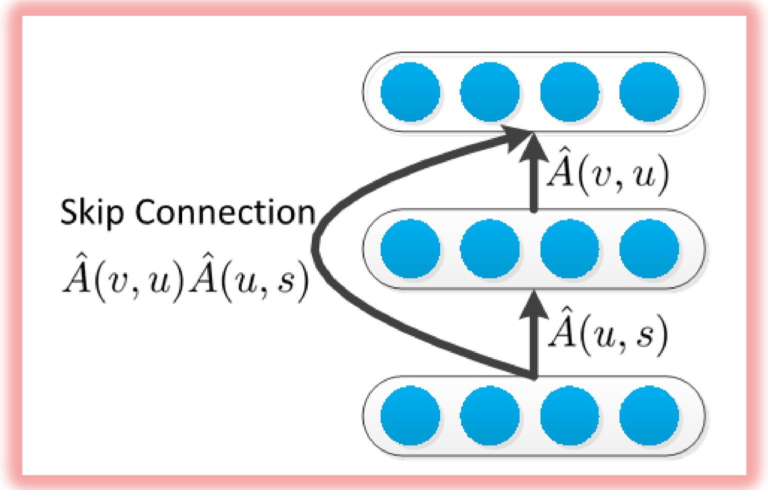
Surrogate Sampler

$$q^*(u_j) = \frac{p(u_j|v_i)|g(x(u_j))|}{\sum_{j=1}^{N} p(u_j|v_i)|g(x(u_j))|},$$

with $g(x(u_j)) = W_g x(u_j)$

- Two more things: skip connections and new attentions

I. Skip Connections



II. New Attentions

$$h^{(l+1)}(v_i) = \sigma(\sum_{j=1}^{N} a((h^{(l)}(v_i), (h^{(l)}(u_j)) h^{(l)}(v_j) W^{(l)})$$

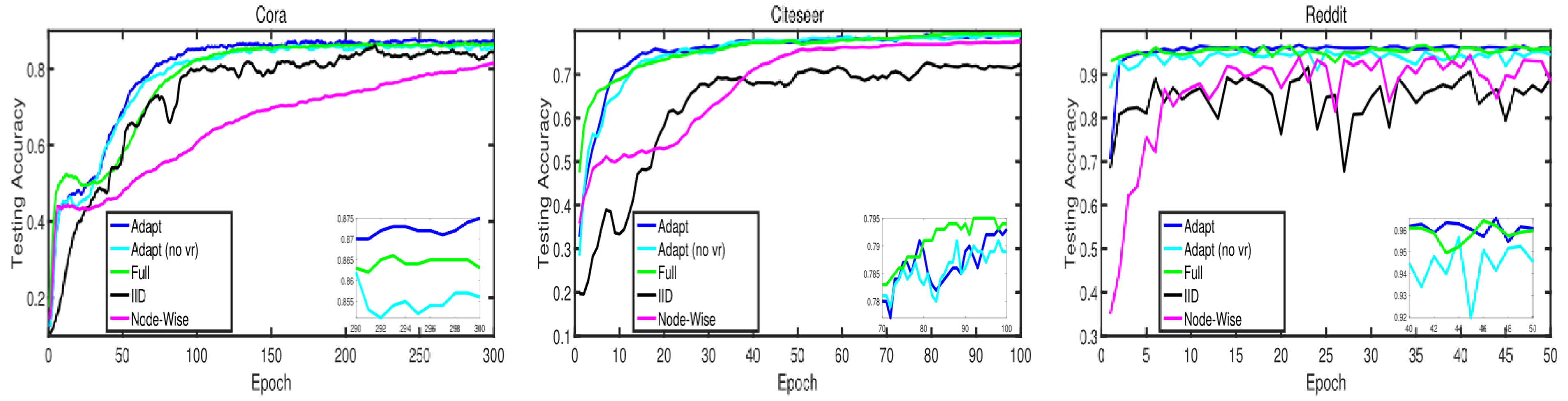
Replace hidden features with self-dependent functions,

$$a(x(v_i), x(u_j)) = \frac{1}{n}\mathrm{ReLu}(W_1 g(x(v_i)) + W_2 g(x(u_j)))$$

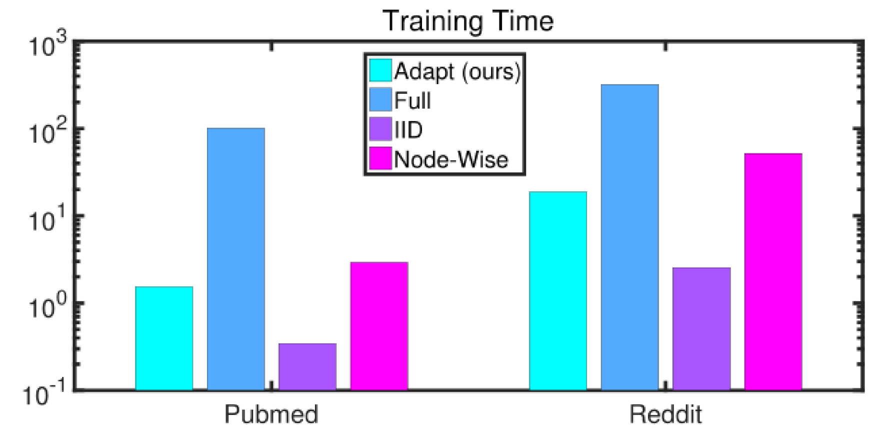
Experiments

- Classification accuracies on Cora, Citeseer, Pubmed and Reddit. We use all training samples for training.

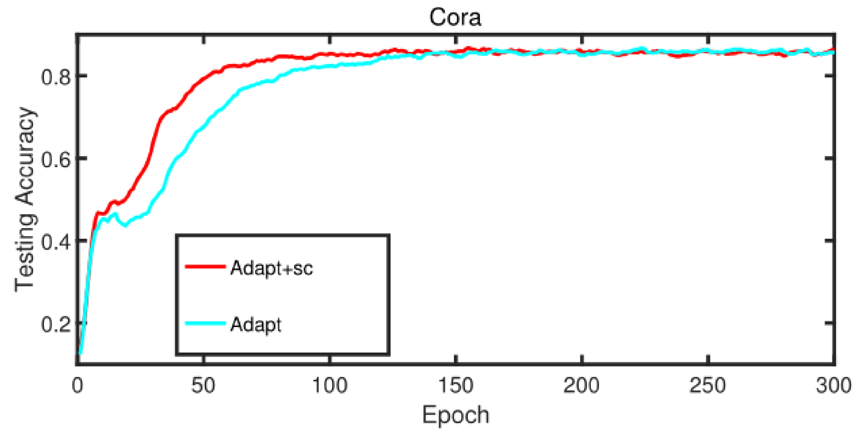
Methods	Cora	Citeseer	Pubmed	Reddit
KLED [25]	0.8229	-	0.8228	-
2-hop DCNN [18]	0.8677	-	0.8976	-
FastGCN [21]	0.8500	0.7760	0.8800	0.9370
GraphSAGE[3]	0.8220	0.7140	0.8710	0.9432
Full	0.8664 ± 0.0011	0.7934 ± 0.0026	0.9022 ± 0.0008	0.9568 ± 0.0069
IID	0.8506 ± 0.0048	0.7387 ± 0.0078	0.8200 ± 0.0114	0.8611 ± 0.0437
Node-Wise	0.8202 ± 0.0133	0.7734 ± 0.0081	0.9002 ± 0.0017	0.9449 ± 0.0026
Adapt (no vr)	0.8588 ± 0.0062	0.7942 ± 0.0022	0.9060 ± 0.0024	0.9501 ± 0.0047
Adapt	$\mathbf{0.8744 \pm 0.0034}$	$\mathbf{0.7966 \pm 0.0018}$	$\mathbf{0.9060 \pm 0.0016}$	$\mathbf{0.9627 \pm 0.0032}$



- Time comparisons and the impact of skip connections



(a)



(b)