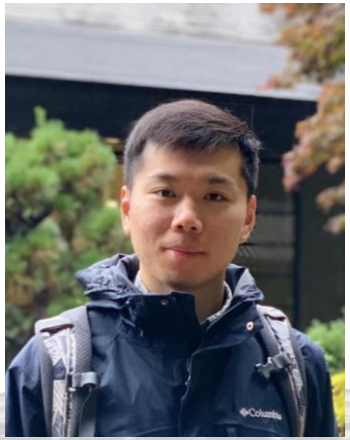
JuryGCN: Quantifying Jackknife Uncertainty on Graph Convolutional Networks

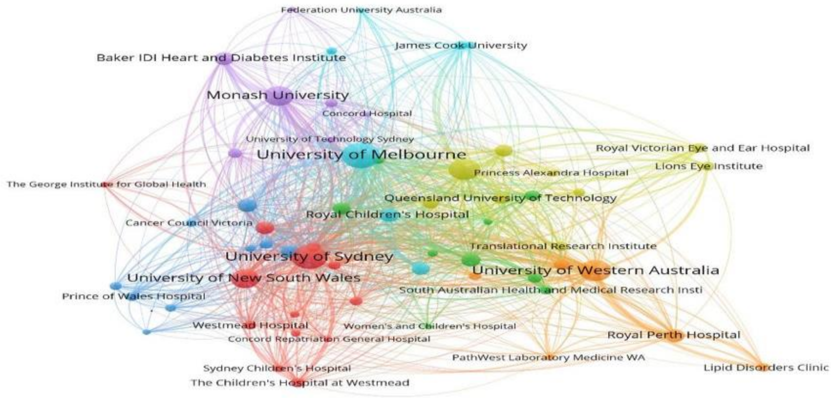




Jian Kang* Qinghai Zhou* Hanghang Tong

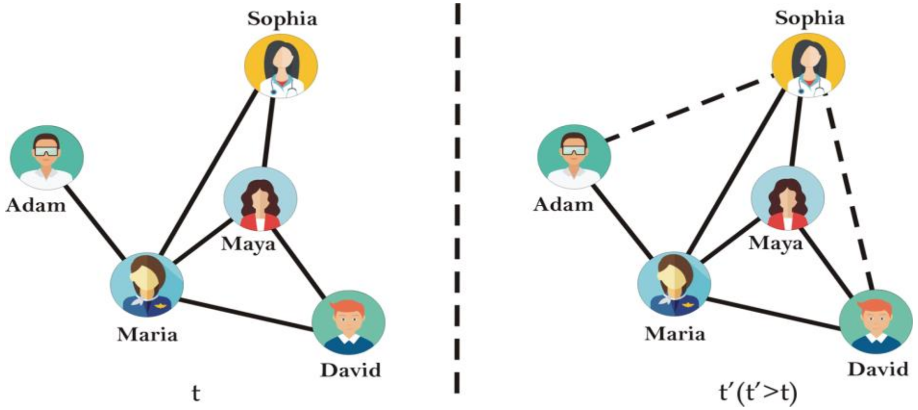
ILLINOIS

*: equal contribution

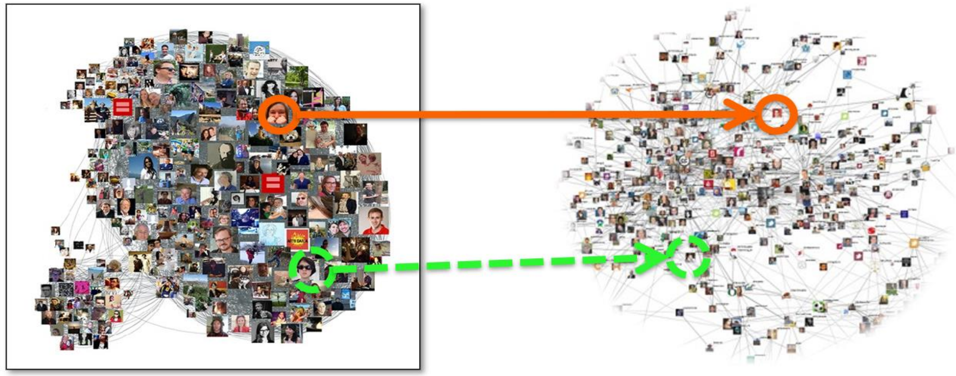
Applications of Graph Neural Networks



Node classification [1]



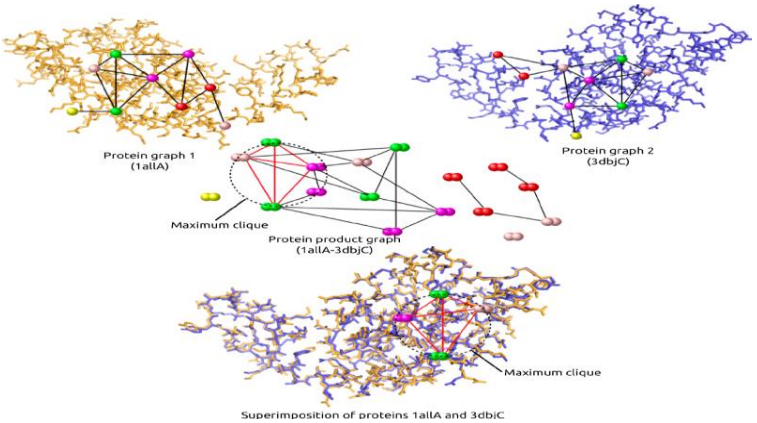
Link prediction [2]



Network A: Facebook

Network B: Twitter

Network alignment [3]

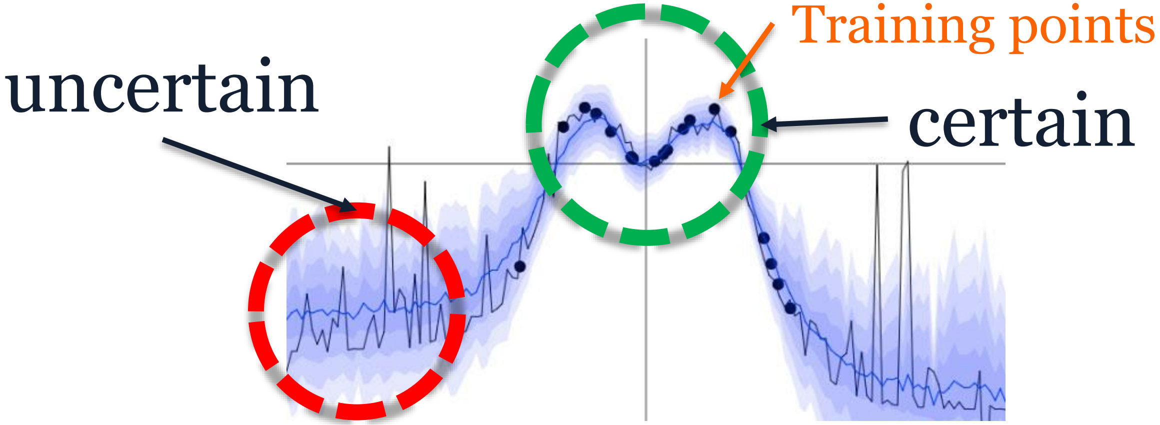


Graph classification [4]

[1] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. In *arXiv 2016*.
[2] Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. In NeurIPS 2018.
[3] Zhang, S., Tong, H., Xia, Y., Xiong, L., & Xu, J. (2020, August). Nettrans: Neural cross-network transformation. In KDD 2020.
[4] Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. In arXiv 2019.

Uncertainty in Model Prediction

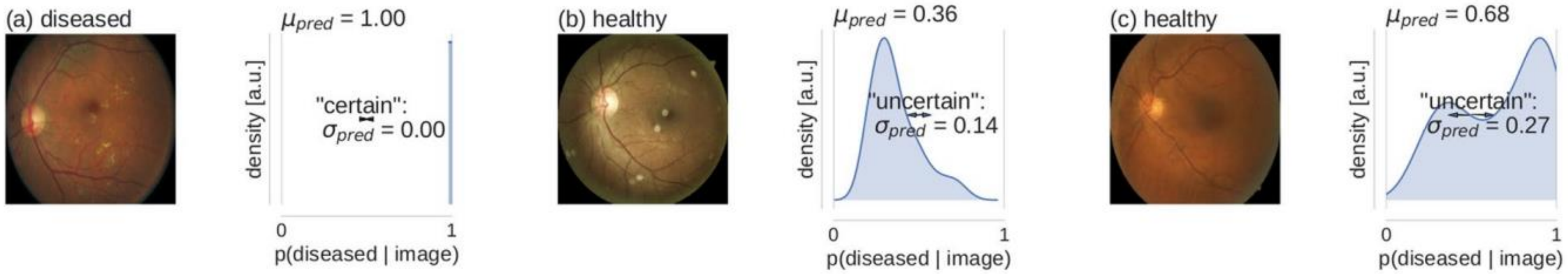
- Examples



Regression

Classification

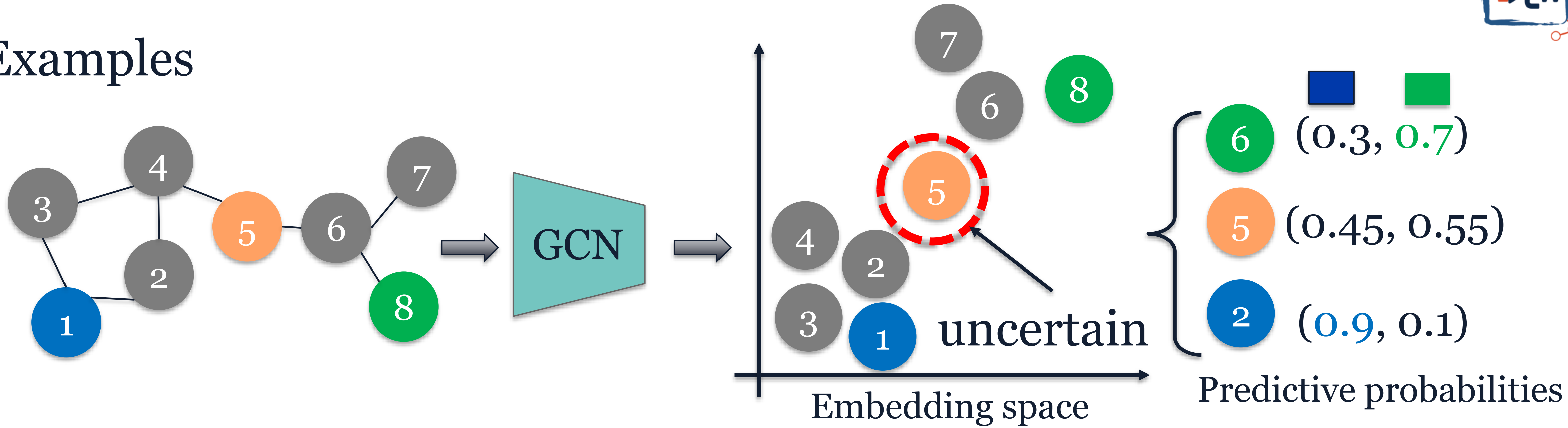
- Quantifying the uncertainty is important in high-risk applications
  - E.g., medical



[1] Peterson, J. C., Battleday, R. M., Griffiths, T. L., & Russakovsky, O. (2019). Human uncertainty makes classification more robust. In ICCV 2019.
[2] Xiao, Y., & Wang, W. Y. (2019, July). Quantifying uncertainties in natural language processing tasks. In AAAI 2019.

Uncertainty in Graph Learning

- Examples



- Questions:
  - Q1: How uncertain is a GCN in its own predictions?
    - ➔Uncertainty quantification (UQ)
  - Q2: How to improve GCN predictions by leveraging uncertainty?
    - ➔ Application of UQ

[1] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

Existing Solutions: Bayesian-based

- Motivation: address over-smoothing/fitting

- Key idea:
  - adaptively drop edges
  - Monte Carlo estimation for posterior uncertainty [1].

- Limitations: not explicitly quantify the uncertainty on model prediction (ad-hoc)

[1] Hasanzadeh, A. et al. Bayesian graph neural networks with adaptive connection sampling. In ICML 2020.
[2] Zhao, X., Chen, F., Hu, S., & Cho, J. H. (2020). Uncertainty aware semi-supervised learning on graph data. In NeurIPS 2020.

Existing Solutions: Deterministic Quantification-based

- Motivation: estimate multi-source uncertainty for GNNs

- Key idea: a graph-based Dirichlet distribution to reduce errors in quantifying uncertainties [2].

- Limitations: changing the training procedure, e.g., additional parameters (e.g., Dirichlet distribution) or architectures (e.g., teacher network)

[1] Hasanzadeh, A. et al. Bayesian graph neural networks with adaptive connection sampling. In ICML 2020.
[2] Zhao, X., Chen, F., Hu, S., & Cho, J. H. (2020). Uncertainty aware semi-supervised learning on graph data. In NeurIPS 2020.

Roadmap

- Background & Motivation ✓
- JuryGCN Formulation
- JuryGCN Algorithms
- JuryGCN Applications
- Experimental Results
- Conclusion

Problem Definition

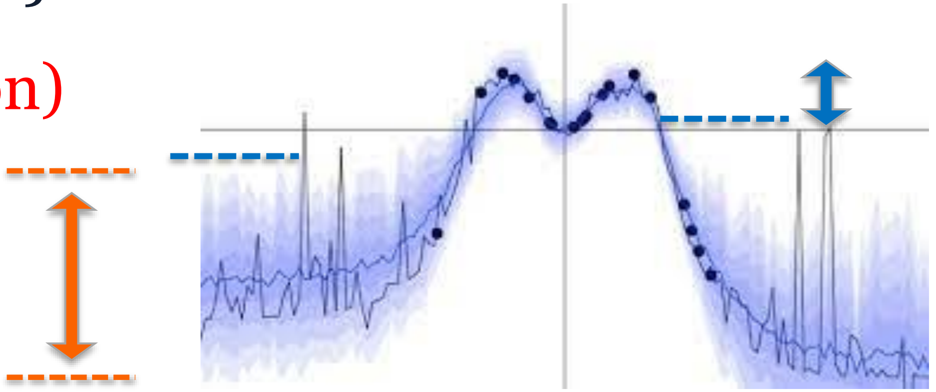
- Given:
  (1) an undirected graph $G = \{V, \mathbf{A}, \mathbf{X}\}$;
  (2) an L-layer GCN with parameter Θ;
  (3) a task-specific objective $R(G, Y, \Theta)$ (Y: ground-truth)
- Find:
  An uncertainty score $\mathrm{U}_{\Theta}(u)$ for any node u in graph G w.r.t. parameters Θ and objective $R(G, Y, \Theta)$.

Preliminaries: Jackknife+ Resampling

- Key idea: leaving out an observation ➔ evaluating prediction error (LOO)
- Given: training data: $D = \{(x_i, y_i) | i = 1, \dots, n\}$; a test point (x^*, y^*); a trained model $f_\theta()$; target coverage α;
- Confidence interval: $[C^-(x^*), C^+(x^*)]$
  - $C^+(x^*) = Q_{1-\alpha}(P^+), C^-(x^*) = Q_\alpha(P^-)$
  - $P^+ = \{f_{\theta_{-i}}(x^*) + |y_i - f_{\theta_{-i}}(x_i)| | i = 1, \dots, n\}$

    $P^- = \{\underbrace{f_{\theta_{-i}}(x^*)}_{\text{LOO prediction}} - \underbrace{|y_i - f_{\theta_{-i}}(x_i)|}_{\text{Error residual (generalization)}} | i = 1, \dots, n\}$

Larger interval ➔ less confident



[1] Barber, Rina Foygel, et al. "Predictive inference with the jackknife+." The Annals of Statistics 49.1 (2021): 486-507.

Jackknife+ Resampling: A Numerical Example

- Regression task: training set, $\{(x_1, y_1), \ldots, (x_5, y_5)\}$, a test point, (x^*, y^*) where $y^* = 10$, coverage, $\alpha = 0.2$

$x_1, x_2, \ldots, x_5$

Train

f_θ

Testing

$f_\theta(x^*) = 9.8$

$x_1, x_2, \ldots, x_5$

Leave-one-out, re-train $f()$

Repeat 5 times

$\{f_{\theta_{-1}}, \ldots, f_{\theta_{-5}}\}$

Testing

$f_{\theta_{-i}}(x^*) = \{9.9, 9.7, 9.6, 10.1, 10.2\}$

$$P^+ = \{f_{\theta_{-i}}(x^*) + |y_i - f_{\theta_{-i}}(x_i)| \mid i\}$$
$$P^- = \{f_{\theta_{-i}}(x^*) - |y_i - f_{\theta_{-i}}(x_i)| \mid i\}$$

$$P^+ = \{9.9, 10.1, 10.2, 9.9, 10\}$$
$$P^- = \{9.7, 9.5, 9.4, 9.7, 9.6\}$$

Applying quantile: $Q_{1-\alpha}(P^+), Q_\alpha(P^-)$

$C^+(x^*) = 10.1, C^-(x^*) = 9.4$

Confidence interval width = 0.7

[1] Barber, Rina Foygel, et al. "Predictive inference with the jackknife+." The Annals of Statistics 49.1 (2021): 486-507.

Challenges

- C1: How to formally define the Jackknife uncertainty for GNNs?
  - Non-IID graph data
- C2: How to efficiently compute the node uncertainty?
  - Avoid re-training

w.l.o.g, considering a node-level tasks (e.g., node classification)

$$\Theta^* = \mathrm{argmin}_\Theta\, R(G, \boxed{Y_{\text{train}}}, \Theta) = \mathrm{argmin}_\Theta \frac{1}{\boxed{|V_{\text{train}}|}} \sum_v \boxed{r(v, y_v, \Theta)}$$

Training labels

Training set

Node-specific loss (cross-entropy)

$$r(v, y_v, \Theta) = -\sum_{i=1}^{c} y_v[i] \log(GCN(v, \Theta)[i])$$

Jackknife Uncertainty: Definition

- Confidence interval: $\mathrm{U}_{\Theta}(u) = C_{\Theta}^{+}(u) - C_{\Theta}^{-}(u)$

Error residual:

$$err_i = \left|\left|y_i - GCN(i, \Theta_{\epsilon,i}^{*})\right|\right|_2$$

- Compute C^{+}, C^{-}

$$C_{\Theta^*}^{-}(u) = Q_{\alpha}(\{\left|\left|\mathrm{GCN}(u, \Theta_{\epsilon,i}^{*})\right|\right|_2 - err_i | \forall i \in V\_\mathrm{train}\})$$

$$C_{\Theta^*}^{+}(u) = Q_{1-\alpha}(\{\left|\left|\mathrm{GCN}(u, \Theta_{\epsilon,i}^{*})\right|\right|_2 + err_i | \forall i \in V\_\mathrm{train}\})$$

- Why Jackknife+: stable coverage

Upweighting the loss of node i:

$$\Theta_{\epsilon,\mathrm{i}}^{*} = \mathrm{argmin}_{\Theta} \epsilon r(i, y_i, \Theta) \frac{1}{|V_{\mathrm{train}}|} \sum_{v} r(v, y_v, \Theta)$$

Question: How to obtain $\Theta_{\epsilon,\mathrm{i}}^{*}$ without re-training?

[1] Barber, Rina Foygel, et al. "Predictive inference with the jackknife+." The Annals of Statistics 49.1 (2021): 486-507.

Roadmap

- Background & Motivation
- JuryGCN Formulation
- JuryGCN Algorithms
- JuryGCN Applications
- Experimental Results
- Conclusion

Jackknife Uncertainty: Efficient Computation

- Key idea: efficiently estimate $\Theta^*_{\epsilon,\text{i}}$ with influence function [1]
- Taylor expansion over parameters

$$\Theta^*_{\epsilon,\text{i}} \approx \Theta^* + \epsilon \mathbf{I}_{\Theta^*}(i) \;\; (1) \;\; \text{where } \mathbf{I}_{\Theta^*}(i) = \frac{d\Theta^*_{\epsilon,\text{i}}}{d\epsilon}\Big|_{\epsilon \to 0}$$

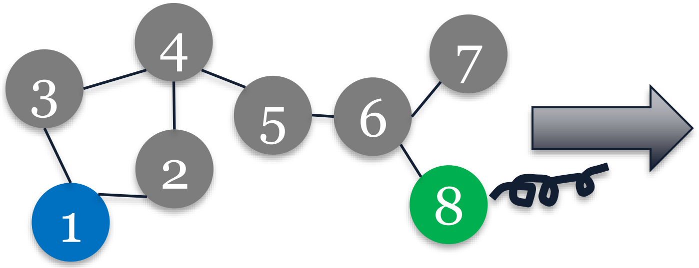
- The influence function can be further computed as [2],

$$\mathbf{I}_{\Theta^*}(i) = \mathbf{H}_{\Theta^*}^{-1} \nabla_\Theta r(i, y_i, \Theta^*) \;\; (2)$$

Hessian matrix w.r.t. model parameters

$$\mathbf{H}_{\Theta^*} = \frac{1}{|V_{\text{train}}|} \nabla^2_\Theta R(G, Y_{\text{train}}, \Theta^*)$$

By setting $\epsilon = -\frac{1}{|V_{\text{train}}|}$, the leave-one-out parameters, $\Theta^*_{\epsilon,\text{i}}$ (Eq. (1)) can be computed efficiently.



Adjusting the weights by ϵ

[1] Pang Wei Koh and Percy Liang. 2017. Understanding Black-Box Predictions via Influence Functions. In ICML 2017.
[2] Cook, R. D., & Weisberg, S. (1982). Residuals and influence in regression. New York: Chapman and Hall.

Jackknife Uncertainty: Efficient Computation (Cont.)

- Proposition: First-order derivative of GCN [1] w.r.t. the parameters in the l-th layer, i.e., $\mathbf{W}^{(l)} \leftarrow \nabla_{\mathbf{W}^{(l)}} \mathrm{r}(\mathrm{i}, \mathrm{y_i}, \Theta)$

$$\boxed{\mathbf{I}_{\Theta^*}(i) = \mathbf{H}_{\mathbf{\Theta}^*}^{-1} \nabla_{\Theta} r(i, y_i, \Theta^*)}$$

  - Key idea: apply chain rule on layer parameters.

$$\nabla_{\mathbf{W}^{(l)}} \mathrm{r}(\mathrm{i}, \mathrm{y_i}, \Theta) = \left(\hat{\mathbf{A}}\mathbf{E}^{l-1}\right)^T \left(\frac{\partial \mathrm{r}(\mathrm{i}, \mathrm{y_i}, \Theta)}{\partial \mathbf{E}^{(l)}} \odot \sigma'\left(\hat{\mathbf{A}}\mathbf{E}^{(l-1)}\mathbf{W}^{(l)}\right)\right)$$

Hidden representations

$$\mathbf{E}^{(l)} = \sigma(\hat{\mathbf{A}}\mathbf{E}^{(l-1)}\mathbf{W}^{(l)})$$

Normalized graph Laplacian

[1] Kang, J. et al. RawlsGCN: Towards Rawlsian Difference Principle on Graph Convolutional Network. In WWW 2022.
[2] Alaa, A. et al. Discriminative Jackknife: Quantifying Uncertainty in Deep Learning via Higher-Order Influence Functions. In ICML 2020.

Jackknife Uncertainty: Efficient Computation (Cont.)

$$\boxed{\mathbf{I}_{\Theta^*}(i) = \mathbf{H}_{\Theta^*}^{-1} \nabla_{\Theta} r(i, y_i, \Theta^*)}$$

- Theorem: Computing the Hessian tensor of GCN

  (the i-th and l-th layer) → $S_{l,i} = \dfrac{\partial^2 R}{\partial \mathbf{W}^{(l)} \partial \mathbf{W}^{(i)}}$

  4-D tensor

  - vectorize the first-order
  - compute the element-wise second-order

  - Flattened Hessian matrix
  - Applying Hessian-vector product [2] using power iteration

THEOREM 1. *(The Hessian tensor of GCN) Following the settings of Proposition 2, denoting the overall loss $R(\mathcal{G}, \mathcal{Y}_{train}, \Theta)$ as R and σ_l' as $\sigma'(\hat{\mathbf{A}}\mathbf{E}^{(l-1)}\mathbf{W}^{(l)})$, the Hessian tensor $\mathfrak{H}_{l,i} = \frac{\partial^2 R}{\partial \mathbf{W}^{(l)} \partial \mathbf{W}^{(i)}}$ of R with respect to $\mathbf{W}^{(l)}$ and $\mathbf{W}^{(i)}$ has the following forms.*

Case 1. $i = l$, $\mathfrak{H}_{l,i} = 0$

Case 2. $i = l - 1$

$$\mathfrak{H}_{l,i}[:,:,c,d] = \left(\hat{\mathbf{A}} \frac{\partial \mathbf{E}^{(l-1)}}{\partial \mathbf{W}^{(i)}[c,d]}\right)^T \left(\frac{\partial R}{\partial \mathbf{E}^{(l)}} \circ \sigma_l'\right) \tag{11}$$

where $\frac{\partial \mathbf{E}^{(l-1)}}{\partial \mathbf{W}^{(i)}[c,d]}$ is the matrix whose entry at the a-th row and the b-th column is

$$\frac{\partial \mathbf{E}^{(l-1)}[a,b]}{\partial \mathbf{W}^{(l-1)}[c,d]} = \sigma_{l-1}'[a,b]\left(\hat{\mathbf{A}}\mathbf{E}^{(l-2)}\right)[a,c]\mathbf{I}[b,d] \tag{12}$$

Case 3. $i < l - 1$

- *Apply Eq.* (12) *for the i-th hidden layer.*
- *Forward to the* $(l-1)$*-th layer iteratively with*

$$\frac{\partial \mathbf{E}^{(l-1)}}{\partial \mathbf{W}^{(i)}[c,d]} = \sigma_{l-1}' \circ \left(\hat{\mathbf{A}} \frac{\partial \mathbf{E}^{(l-2)}}{\partial \mathbf{W}^{(i)}[c,d]} \mathbf{W}^{(l-1)}\right) \tag{13}$$

- *Apply Eq.* (11).

Case 4. $i = l + 1$

$$\mathfrak{H}_{l,i}[:,:,c,d] = (\hat{\mathbf{A}}\mathbf{E}^{(l-1)})^T \left(\frac{\partial^2 R}{\partial \mathbf{E}^{(l)} \partial \mathbf{W}^{(i)}[c,d]} \circ \sigma_l'\right) \tag{14}$$

where $\frac{\partial^2 R}{\partial \mathbf{E}^{(l)}[a,b] \partial \mathbf{W}^{(l+1)}[c,d]} = \mathbf{I}[b,c]\left[\hat{\mathbf{A}}^T\left(\frac{\partial R}{\partial \mathbf{E}^{(l+1)}} \circ \sigma_{l+1}'\right)\right][a,d]$.

Case 5. $i > l + 1$

- *Compute* $\frac{\partial^2 R}{\partial \mathbf{E}^{(i-1)} \partial \mathbf{W}^{(i)}[c,d]}$ *whose* (a,b)*-th entry has the form* $\frac{\partial^2 R}{\partial \mathbf{E}^{(i-1)}[a,b] \partial \mathbf{W}^{(i)}[c,d]} = \mathbf{I}[b,c]\left(\hat{\mathbf{A}}^T\left(\frac{\partial R}{\partial \mathbf{E}^{(i)}} \circ \sigma_i'\right)\right)[a,d]$
- *Backward to* $(l+1)$*-th layer iteratively with*

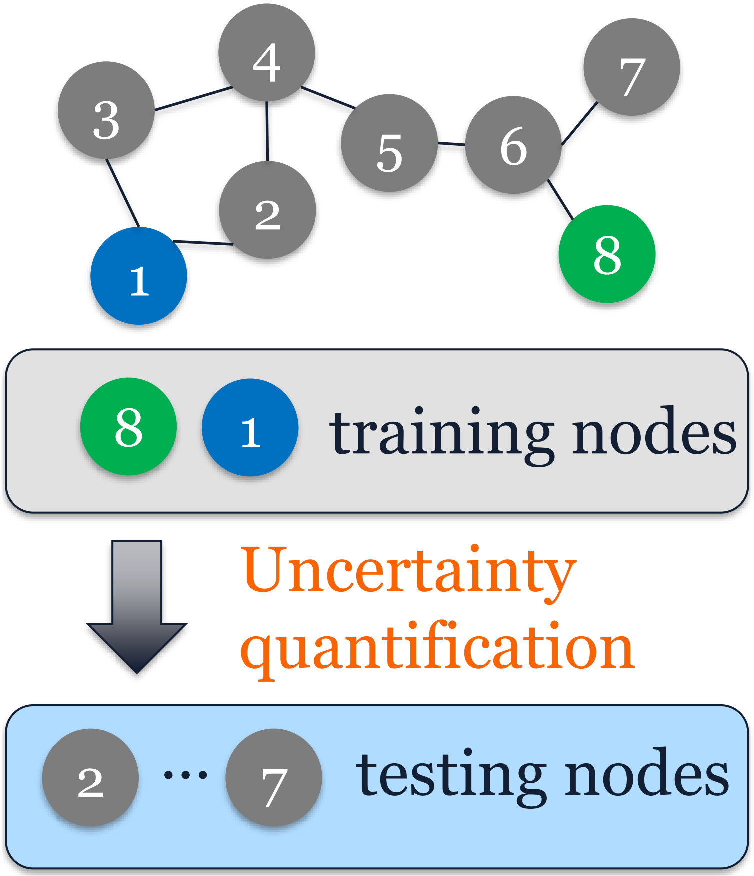
$$\frac{\partial^2 R}{\partial \mathbf{E}^{(l)} \partial \mathbf{W}^{(i)}[c,d]} = \hat{\mathbf{A}}^T \left(\frac{\partial^2 R}{\partial \mathbf{E}^{(l+1)} \partial \mathbf{W}^{(i)}[c,d]} \circ \sigma_{l+1}'\right) (\mathbf{W}^{(l+1)})^T \tag{15}$$

- *Apply Eq.* (14).

[1] Kang, J. et al. RawlsGCN: Towards Rawlsian Difference Principle on Graph Convolutional Network. In WWW 2022.
[2] Alaa, A. et al. Discriminative Jackknife: Quantifying Uncertainty in Deep Learning via Higher-Order Influence Functions. In ICML 2020.

Algorithm: JuryGCN

- Goal: to estimate uncertainty $\mathrm{U}_{\Theta}(u)$ of node u.
- Initialize: $\epsilon = -\frac{1}{|V_{\text{train}}|}$, a GCN with parameter Θ
- Key steps (for each training node):
  - Compute node-wise loss $r_{i,\Theta}$ and derivative $\nabla_{\Theta} r_{i,\Theta}$
  - Evaluate the influence w.r.t. training node
  - Compute LOO parameters/predictions/errors
  - Compute lower and upper bound
- Return: confidence interval of node u.

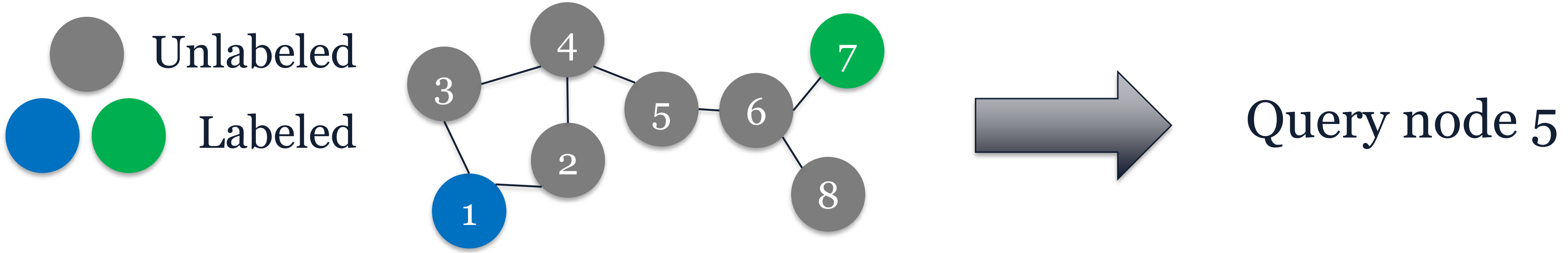


Roadmap

- Background & Motivation
- JuryGCN Formulation
- JuryGCN Algorithms
- JuryGCN Applications
- Experimental Results
- Conclusion

Applications: Active Learning on Node Classification

- Task: query the nodes for true labels ➔ node classifier
- General idea: select the most informative nodes



- Our idea: iteratively query the nodes with the largest uncertainty

$$Acq(V_{\text{train}}) = \text{argmax}_{u \in V_{\text{train}}} \mathrm{U}_{\Theta}(u)$$

Applications: Semi-supervised Node Classification

- Existing objective: mean of loss from all training nodes

$$R = \frac{1}{|V_{\text{train}}|} \sum_{i \in V_{\text{train}}} r(i, y_i, \Theta)$$

- Uncertainty-aware node-specific objective

$$r_u = -\beta_u^{\tau} \log\left(p_u^{(i)}\right) \qquad \beta_u = \frac{|\mathrm{U}_{\Theta}(u)|}{\sqrt{\sum_{i \in V_{\text{train}}} |\mathrm{U}_{\Theta}(u)|^2}}$$

normalizing over all training nodes

i-th class predictive probability

(1) u is misclassified & $p_u^{(i)}$ is small.

(2) u is well classified & $p_u^{(i)}$ is large.

[1] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In ICCV 2017.

Roadmap

- Background & Motivation
- JuryGCN Formulation
- JuryGCN Algorithms
- JuryGCN Applications
- Experimental Results
- Conclusion

Experiment Settings

Datasets	Cora	Citeseer	PubMed	Reddit
# nodes	2, 708	3, 327	19, 717	232, 965
# edges	5, 429	4, 732	44, 338	114, 615, 892
# features	1, 433	3, 703	500	602
# classes	7	6	3	41

- Datasets: 4 widely-adopted datasets
- Evaluation metric: micro-F1
- Comparison methods
  - Active learning-based: AGE[1], ANRMAB[2], Coreset[3], SOPT-GCN[4], Centrality, Random
  - Semi-supervised: S-GNN[5], GPN[6], GCN[7], GAT[8]
- Parameters
  - Active node classification (Cora, Citeseer, Pubmed and Reddit)
    - Query budget: 100, 100, 50, 250, step size: 20, 20, 10, 50
  - Semi-supervised node classification
    - hyperparameter: $\tau = 2$, coverage: $\alpha = 0.025$

[1] Cai, H. et al. Active Learning for Graph Embedding. In arXiv 2017.
[2] Gao, L. et al. Active Discriminative Network Representation Learning. In IJCAI 2018.
[3] Sener, O. et al. Active Learning for Convolutional Neural Networks: A Core-set Approach. In arXiv 2017.
[4] Ng, Y. et al. Bayesian Semi-Supervised Learning with Graph Gaussian Processes. In NeurIPS 2018.
[5] Zhao, X. et al. Uncertainty Aware Semi-Supervised Learning on Graph Data. In NeurIPS 2020.
[6] Stadler, M. et al. Graph Posterior Network: Bayesian Predictive Uncertainty for Node Classification. In NeurIPS 2021.
[7] Kipf, T. et al. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR 2016.
[8] Veličković, P. et al. Graph Attention Networks. In ICLR 2017.

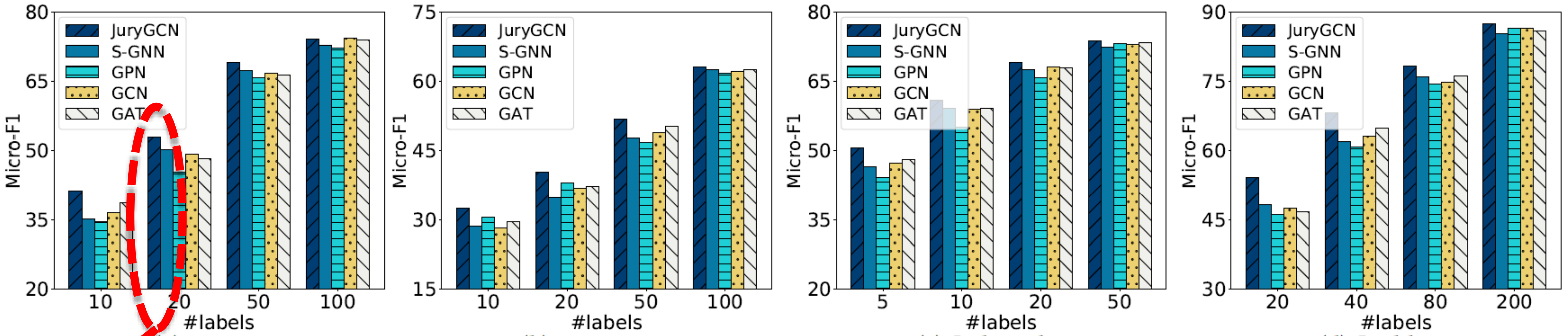
Experimental Results:
Active Learning on Node Classification

proposed method

Data	Query size	JuryGCN (Ours)	ANRMAB	AGE	Coreset	Centrality	Degree	Random	SOPT-GCN
Cora	20	**51.1 ± 1.2**	46.8 ± 0.5	49.4 ± 1.0	43.8 ± 0.8	41.9 ± 0.6	38.5 ± 0.7	40.5 ± 1.6	48.8 ± 0.7
	40	**64.7 ± 0.8**	61.2 ± 0.8	58.2 ± 0.7	55.4 ± 0.5	57.3 ± 0.7	48.4 ± 0.3	56.8 ± 1.3	62.6 ± 0.8
	60	**69.9 ± 0.9**	67.8 ± 0.7	65.7 ± 0.8	62.2 ± 0.6	63.1 ± 0.5	58.8 ± 0.6	64.5 ± 1.5	67.9 ± 0.6
	80	**74.2 ± 0.7**	73.3 ± 0.6	72.5 ± 0.4	70.2 ± 0.5	69.1 ± 0.4	67.6 ± 0.4	69.7 ± 1.6	73.6 ± 0.5
	100	**75.5 ± 0.6**	74.9 ± 0.4	74.2 ± 0.3	73.8 ± 0.4	74.1 ± 0.3	73.0 ± 0.2	74.2 ± 1.2	**75.5 ± 0.7**
Citeseer	20	**38.4 ± 1.5**	35.9 ± 1.0	33.1 ± 0.9	30.2 ± 1.2	35.6 ± 1.1	31.5 ± 0.9	30.3 ± 2.3	36.1 ± 0.7
	40	**51.1 ± 0.9**	46.7 ± 1.3	49.5 ± 0.6	42.1 ± 0.8	49.8 ± 1.3	39.8 ± 0.7	41.1 ± 1.8	49.2 ± 0.5
	60	**58.2 ± 0.8**	55.2 ± 0.9	56.1 ± 0.5	52.1 ± 0.9	57.1 ± 0.7	50.1 ± 1.1	49.8 ± 1.3	56.4 ± 0.5
	80	**63.8 ± 1.1**	63.2 ± 0.7	61.5 ± 0.8	59.9 ± 0.6	63.3 ± 1.0	58.8 ± 0.6	58.1 ± 1.1	63.2 ± 0.8
	100	**64.3 ± 1.2**	64.1 ± 0.5	63.2 ± 0.7	62.8 ± 0.4	63.9 ± 0.6	61.8 ± 0.5	62.9 ± 0.8	63.8 ± 0.6
Pubmed	10	**61.8 ± 0.9**	60.5 ± 1.3	58.9 ± 1.1	53.1 ± 0.7	55.8 ± 1.2	56.4 ± 1.5	52.4 ± 1.7	59.5 ± 0.6
	20	**70.2 ± 0.6**	66.8 ± 1.1	68.7 ± 0.7	62.8 ± 0.5	67.2 ± 1.4	64.3 ± 1.0	60.5 ± 1.4	67.9 ± 0.9
	30	**73.9 ± 0.3**	71.6 ± 0.8	72.8 ± 1.0	68.9 ± 0.3	73.5 ± 0.9	70.1 ± 0.7	68.9 ± 1.1	72.3 ± 0.8
	40	74.6 ± 0.4	73.2 ± 0.6	**74.7 ± 0.8**	72.8 ± 0.8	74.1 ± 0.7	72.0 ± 0.8	71.8 ± 1.2	73.8 ± 0.7
	50	**75.4 ± 0.5**	74.7 ± 0.4	75.1 ± 0.5	73.5 ± 0.6	74.2 ± 0.6	72.9 ± 0.5	73.1 ± 1.0	75.2 ± 0.5
Reddit	50	**69.7 ± 1.7**	67.8 ± 0.9	64.2 ± 1.1	62.1 ± 0.6	65.5 ± 1.2	62.5 ± 1.4	63.7 ± 2.4	68.1 ± 1.2
	100	**82.9 ± 1.5**	81.3 ± 1.0	79.5 ± 0.8	81.2 ± 1.0	78.2 ± 0.9	81.1 ± 1.2	80.5 ± 1.6	80.4 ± 1.3
	150	**86.0 ± 1.4**	84.3 ± 0.7	83.2 ± 0.4	84.8 ± 0.9	84.1 ± 1.1	82.5 ± 1.2	81.5 ± 1.4	85.0 ± 1.5
	200	**88.1 ± 0.9**	86.1 ± 0.8	85.8 ± 0.5	85.5 ± 0.8	87.5 ± 0.8	85.4 ± 0.7	83.1 ± 1.8	87.2 ± 0.9
	250	**89.2 ± 0.8**	87.6 ± 0.7	87.1 ± 0.4	86.6 ± 1.1	88.7 ± 0.6	86.1 ± 1.0	87.3 ± 1.5	87.8 ± 1.1

Observation: JuryGCN achieves the best query performance

Experimental Results: Semi-supervised Node Classification



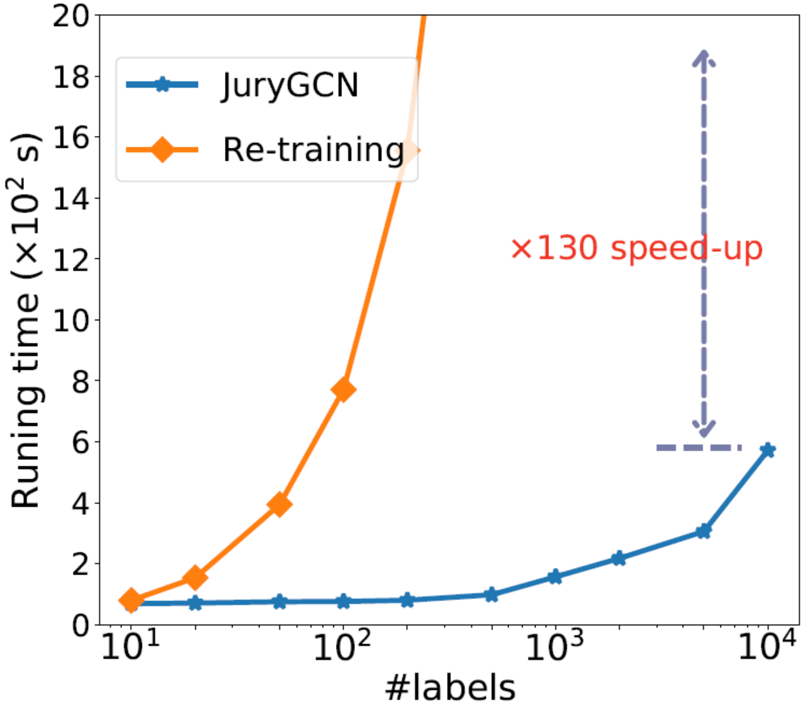
(a) Cora (b) Citeseer (c) Pubmed (d) Reddit

proposed method

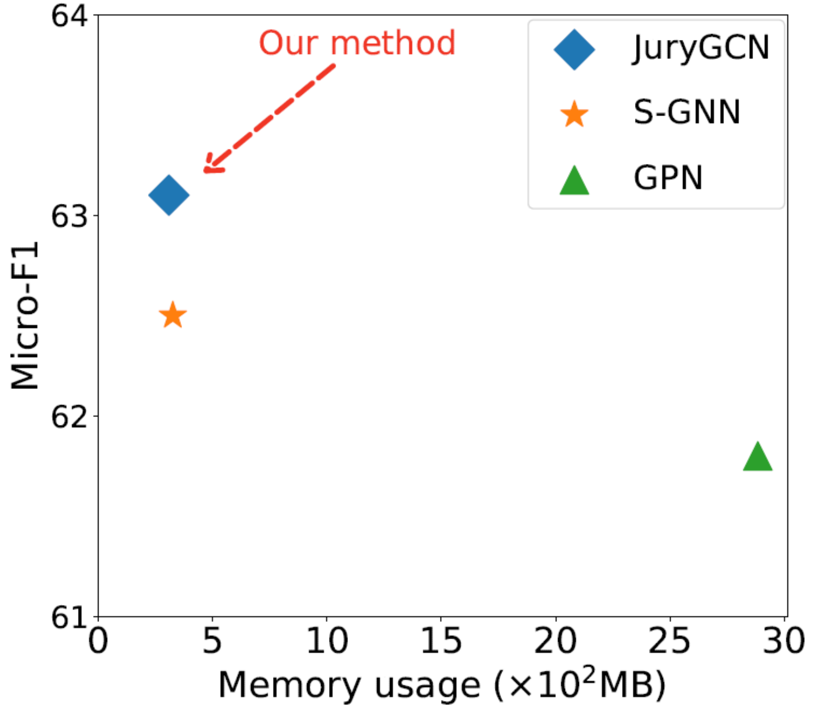
Observation: achieving better performance when #labels is smaller

Experimental Results: Efficiency

- Metrics: running time, memory usage



Running time vs. #training labels

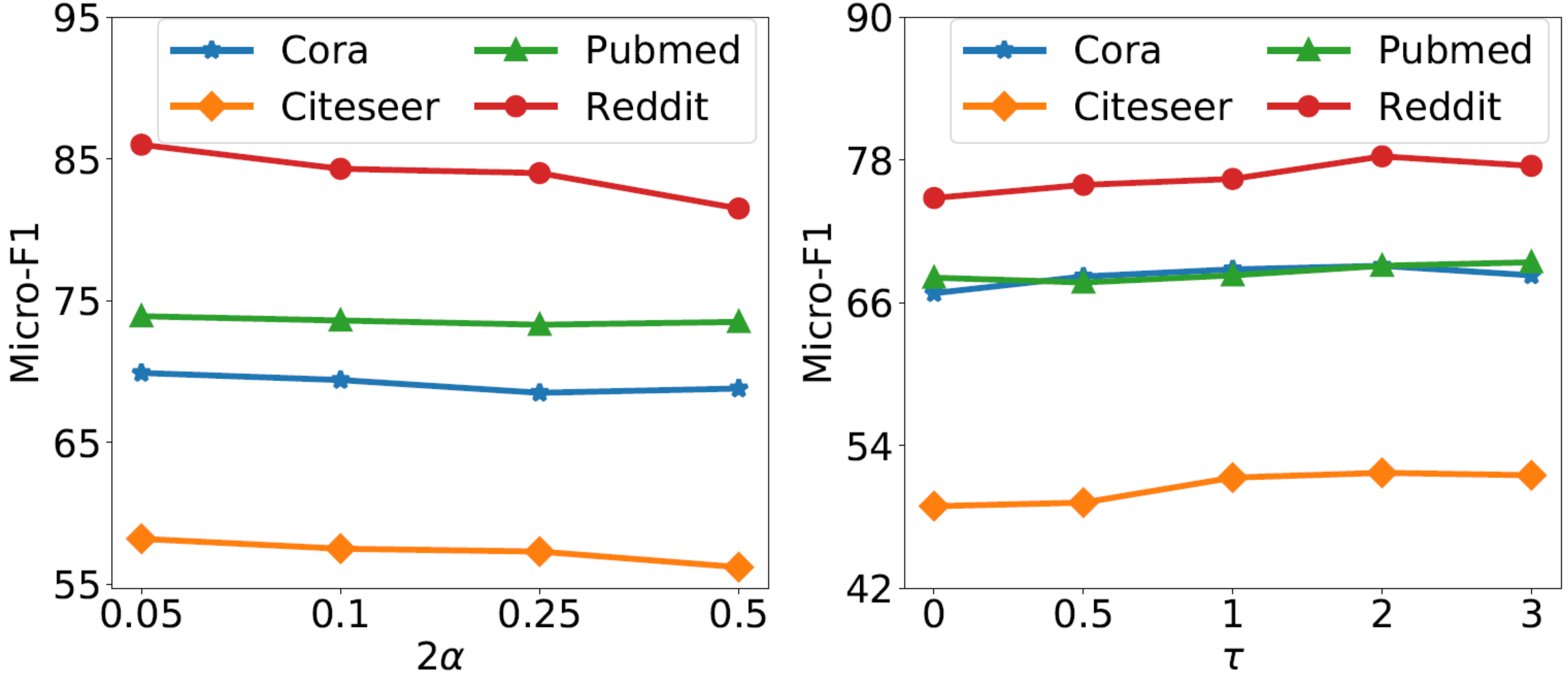


Micro-F1 vs. memory usage

Observation: JuryGCN can achieve the best efficiency performance.

Experimental Results: Parameter Study

- coverage, α; hyperparameter, τ



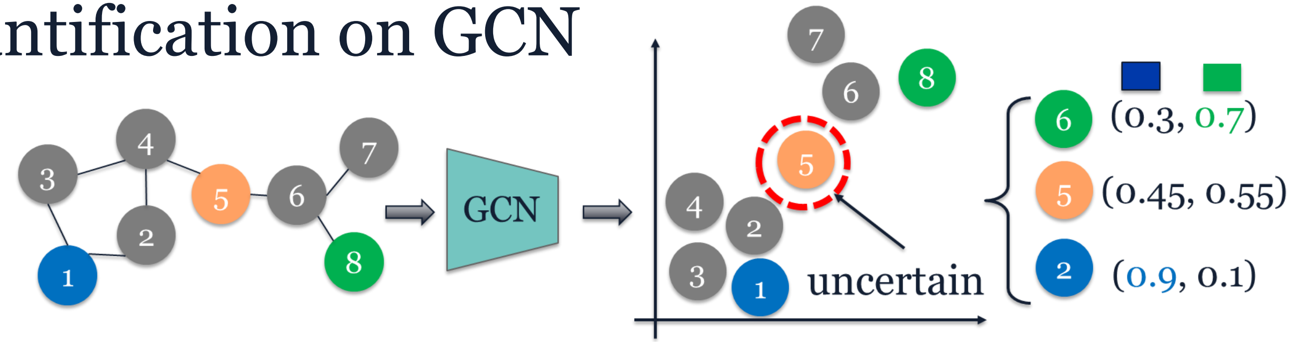
Observation: constantly achieving good performance.

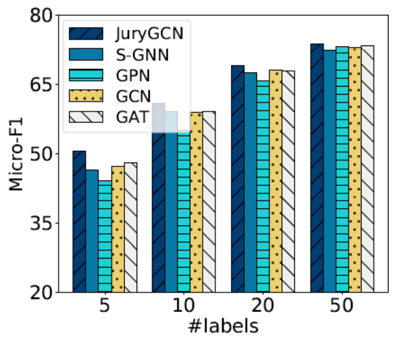
Roadmap

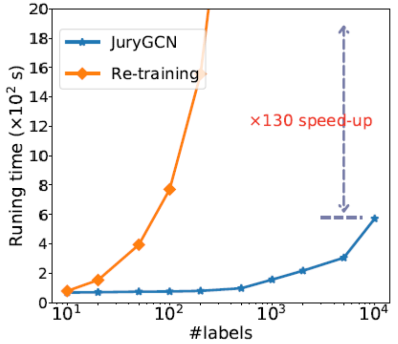
- Background & Motivation ✓
- JuryGCN Formulation ✓
- JuryGCN Algorithms ✓
- JuryGCN Applications ✓
- Experimental Results ✓
- Conclusion

Conclusion

- ❑Problem: Jackknife Uncertainty Quantification on GCN
- ❑Solution:
  - Jackknife+ estimation
  - Influence-based approach
- ❑Applications:
  - Active learning on node classification
  - Semi-supervised node classification
- ❑Results: outperforming other comparison method
  - Improve node classification accuracy
  - Select the most informative nodes
  - Efficient computation compared to re-training







Authors: Jian Kang, Qinghai Zhou, Hanghang Tong
Email: jiank2@illinois.edu, qinghai2@illinois.edu