

ICML 2022



NAFS: A Simple yet Tough-to-beat Baseline for Graph Representation

Wentao Zhang, Zeang Sheng, Mingyu Yang, Yang Li, Yu Shen, Zhi Yang, Bin Cui

Peking University

Graph Neural Networks

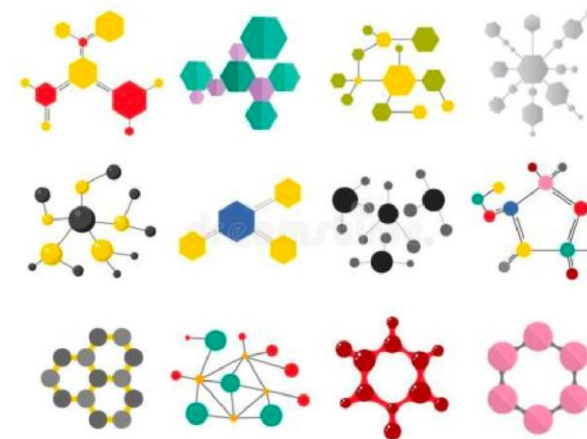
Many real-world data are **graphs**.



Social Network



Knowledge Graph



Drugs and New materials

- Graph representation learning has been extensively applied in various application scenarios.
- GNN-based graph representation learning has attracted intensive interest by combining knowledge from both graph structure and node features

Drawbacks of current methods

1) **Low Scalability**

They can not scale well to large graphs due to the expensive computation cost and high memory usage.

- Repeatedly perform the computationally expensive and recursive feature smoothing
- Introduce high memory usage by storing the dense-form adjacency matrix on GPU.

2) **Shallow Architecture**

The performance decreases as the layers become deeper.

- Information Propagation: Shallow architecture can not involve the full graph information due a few propagations.
- Nonlinear Transformation: The expressive power is low due to a few nonlinear transformation.

Observation



➤ Over-Smoothing Issue

When applying $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{r-1} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-r}$ as adjacency matrix $\hat{\mathbf{A}}$, the stationary state follows

$$\hat{\mathbf{A}}_{i,j}^{\infty} = \frac{(d_i + 1)^r (d_j + 1)^{1-r}}{2m + n}, \quad (3)$$

After infinite times of aggregation, the influence from node v_i to v_j is only determined by the degrees of them.

➤ Over-Smoothing Distance

Definition 1 (Over-smoothing Distance). *The Over-smoothing Distance $D_i(k)$ parameterized by node i and smoothing step k is defined as*

$$D_i(k) = \text{Dis}([\hat{\mathbf{A}}^k \mathbf{X}]_i, [\hat{\mathbf{A}}^{\infty} \mathbf{X}]_i), \quad (4)$$

A smaller value indicates that the node is closer to the stationary state, i.e., closer to over-smoothing.

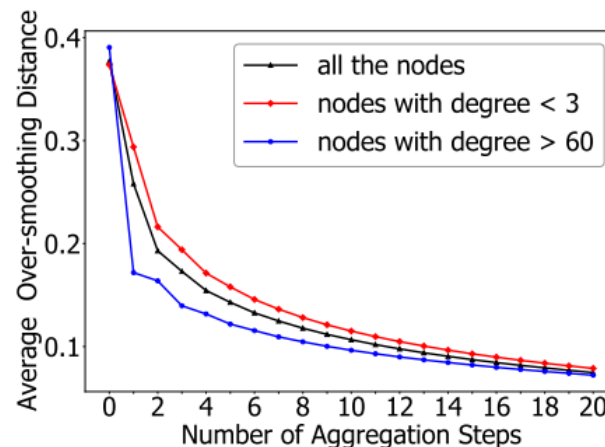


Figure 2. The diverse smoothing speed across nodes with different degrees. Nodes with larger degree have larger smoothing speed.

- Many works (SGC, SIGN, GAMLP) propose to decouple the feature smoothing and feature transformation in each GCN layer.
- Nodes with high degrees should have relatively small smoothing steps than nodes with low degrees. **Node-adaptive feature smoothing** must be adopted to satisfy each node's diverse needs of smoothing level.

Node-Adaptive Feature Smoothing

➤ Feature Smoothing:

We define the **smoothing weight** so that the smoothing operation can be performed in a node-adaptive manner.

Definition 2 (Smoothing Weight). The Smoothing Weight $w_i(k)$ parameterized by node v_i and smoothing step k is defined with the softmax value of $\{D_i(0), D_i(1), \dots, D_i(K)\}$:

$$w_i(k) = e^{D_i(k)} / \sum_{l=0}^K e^{D_i(l)}, \quad (5)$$

where K is the maximal smoothing step.

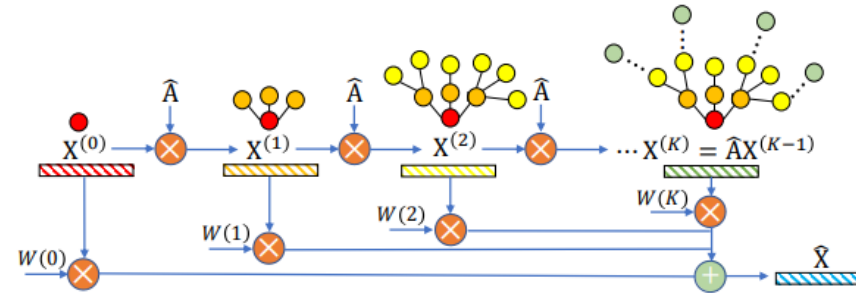


Figure 3. An overview of node-adaptive feature smoothing.

➤ Feature Ensemble:

Different smoothing operators actually act as different knowledge extractors.

we vary the value of r in the normalized adjacency matrix

$$\hat{\mathbf{A}}_r = \tilde{\mathbf{D}}^{r-1} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-r}$$

Firstly perform the feature smoothing operation to generate corresponding smoothed features $\{\hat{\mathbf{X}}^{(1)}, \hat{\mathbf{X}}^{(2)}, \dots, \hat{\mathbf{X}}^{(T)}\}$

Then, we combine them with concatenating, mean pooling, or max pooling, etc.

Comparison with Existing GNNs



➤ **Deep Structural Information**

By assigning each node with personalized smoothing weights, NAFS can gather deep structural information without encountering the over-smoothing issue.

➤ **Efficiency**

Compared with GAE and its variants, our proposed NAFS does not have any trainable parameters, giving it a significant advantage in efficiency.

➤ **Memory Cost**

NAFS only requires to store the sparse adjacency matrix and the smoothed features, and thus the memory cost is $O(m + nf)$, which grows linearly with graph size in typical real-world graphs.

End-to-end Comparison



Table 1. Link prediction performance comparison.

Methods	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6±0.0	88.5±0.0	80.5±0.0	85.0±0.0	84.2±0.0	87.8±0.0
DeepWalk	83.1±0.3	85.0±0.4	80.5±0.5	83.6±0.4	84.4±0.4	84.1±0.5
GAE	91.0±0.5	92.0±0.4	89.5±0.3	89.9±0.4	96.4±0.4	96.5±0.5
VGAE	91.4±0.5	92.6±0.4	90.8±0.4	92.0±0.3	94.4±0.5	94.7±0.4
ARGA	92.4±0.4	93.2±0.3	91.9±0.5	93.0±0.4	96.8±0.3	97.1±0.5
ARVGA	92.4±0.4	92.6±0.4	92.4±0.5	93.0±0.3	96.5±0.5	96.8±0.4
GALA	92.1±0.3	92.2±0.4	94.4±0.5	94.8±0.5	93.5±0.4	94.5±0.4
AGE	95.1±0.5	94.6±0.5	96.3±0.4	96.6±0.4	94.3±0.3	93.5±0.5
NAFS-mean	92.6±0.0	93.9±0.0	94.9±0.0	95.9±0.0	97.4±0.0	97.2±0.0
NAFS-max	93.0±0.0	94.2±0.0	94.8±0.0	96.0±0.0	97.5±0.0	97.1±0.0
NAFS-concat	92.6±0.0	93.8±0.0	93.7±0.0	93.1±0.0	97.6±0.0	97.2±0.0

Table 2. Node clustering performance comparison.

Methods	Cora			Citeseer			PubMed			Wiki		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
GAE	53.3±0.2	40.7±0.3	30.5±0.2	41.3±0.4	18.3±0.3	19.1±0.3	63.1±0.4	24.9±0.3	21.7±0.2	37.9±0.2	34.5±0.3	18.9±0.2
VGAE	56.0±0.3	38.5±0.4	34.7±0.3	44.4±0.2	22.7±0.3	20.6±0.3	65.5±0.2	25.0±0.4	20.3±0.2	45.1±0.4	46.8±0.3	26.3±0.4
MGAE	63.4±0.5	45.6±0.3	43.6±0.4	63.5±0.4	39.7±0.4	42.5±0.5	59.3±0.5	28.2±0.2	24.8±0.4	52.9±0.3	<u>51.0±0.4</u>	37.9±0.5
ARGA	63.9±0.4	45.1±0.3	35.1±0.5	57.3±0.5	35.2±0.3	34.0±0.4	68.0±0.5	27.6±0.4	29.0±0.4	38.1±0.5	34.5±0.3	11.2±0.4
ARVGA	64.0±0.5	44.9±0.4	37.4±0.5	54.4±0.5	25.9±0.5	24.5±0.3	51.3±0.4	11.7±0.3	7.8±0.2	38.7±0.4	33.9±0.4	10.7±0.2
AGC	68.9±0.5	53.7±0.3	48.6±0.3	66.9±0.5	41.1±0.4	41.9±0.5	69.8±0.4	31.6±0.3	31.8±0.4	47.7±0.3	45.3±0.5	34.3±0.4
DAEGC	70.2±0.4	52.6±0.3	49.7±0.4	67.2±0.5	39.7±0.5	41.1±0.4	66.8±0.5	26.6±0.2	27.7±0.3	48.2±0.4	44.8±0.4	33.1±0.3
AGE	72.8±0.5	58.1±0.6	56.3±0.4	70.0±0.3	44.6±0.4	45.4±0.5	69.9±0.5	30.1±0.4	31.4±0.6	51.1±0.6	53.9±0.4	<u>36.4±0.5</u>
NAFS-mean	70.4±0.0	56.6±0.0	48.0±0.0	71.8±0.0	<u>45.1±0.0</u>	47.6±0.0	70.5±0.0	33.9±0.0	33.2±0.0	54.6±0.0	49.4±0.0	27.3±0.0
NAFS-max	70.8±0.0	56.6±0.0	49.0±0.0	70.1±0.0	<u>45.1±0.0</u>	44.7±0.0	70.6±0.0	<u>33.4±0.0</u>	<u>33.1±0.0</u>	51.4±0.0	45.8±0.0	25.5±0.0
NAFS-concat	75.4±0.0	58.6±0.0	<u>53.8±0.0</u>	<u>71.1±0.0</u>	45.8±0.0	<u>46.1±0.0</u>	70.5±0.0	33.9±0.0	33.2±0.0	<u>53.6±0.0</u>	50.5±0.0	26.3±0.0

➤ Link Prediction

- The proposed NAFS consistently achieves the best or the second-best performance compared with all the baseline methods.

➤ Node Clustering

- Among three ensemble strategies, NAFS-concat has the overall best performance across the three datasets, which also consistently outperforms the strongest baseline - AGE.
- Three NAFS variants outperform training-based GAE on all the four datasets, and they outperforms the current SOTA method, AGE, in most cases.

Efficiency & Scalability

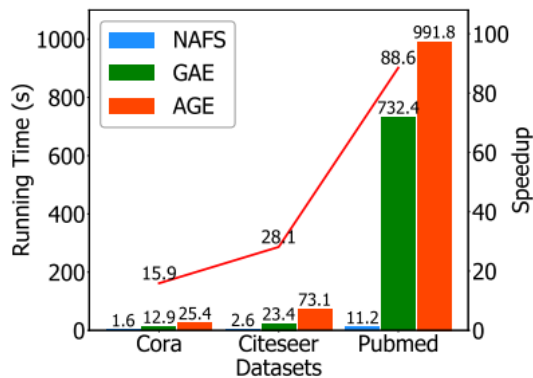


Figure 5. Efficiency comparison on the three citation networks.

Table 3. Scalability comparison with different sample sizes on the ogbn-products dataset.

Methods	10,000	20,000	30,000	50,000	100,000
GAE	10.2s	36.3s	OOM	OOM	OOM
AGE	68.5s	256.2s	727.7s	2315.7s	OOM
NAFS-mean	2.8s	6.5s	10.8s	13.8s	23.7s

➤ Efficiency Analysis

- NAFS is significantly faster than the considered baseline methods on the three datasets

➤ Scalability Analysis

- NAFS can support the larger graphs (i.e., larger than 30,000 nodes) than the compared baselines. Besides, it is significantly faster than the compared baselines, especially for large graph datasets.

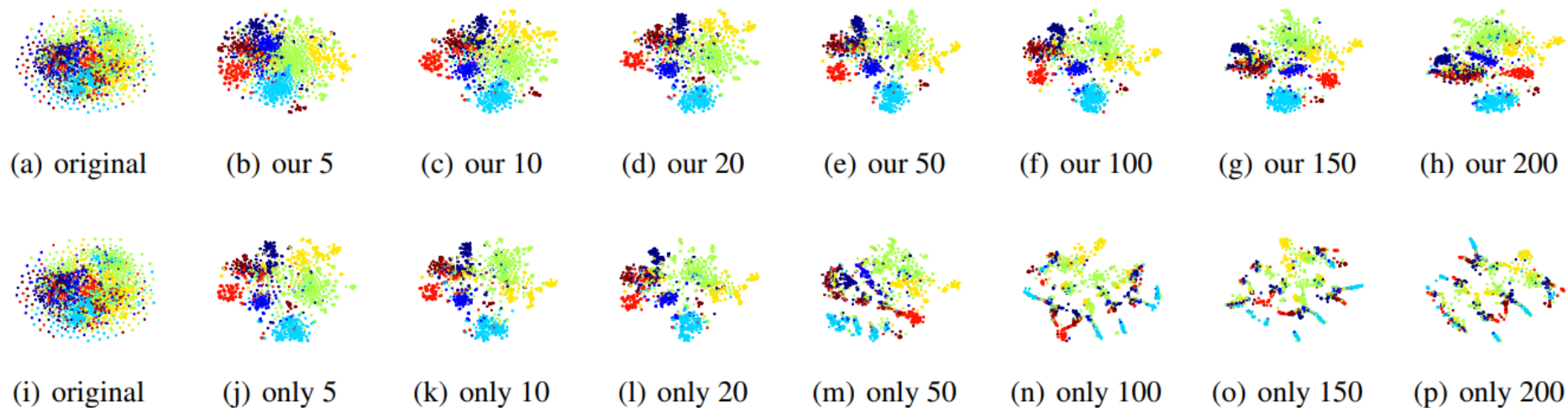


Figure 6. T-SNE visualization of node embedding on the Cora dataset.

➤ Visualization

- At hop 10, the node embedding produced by NAFS and SGC are both distinguishable.
- As the value of maximal smoothing step becomes larger, the node embedding of SGC falls into total disorder, like the situation showed by Figure. 6(n), 6(o) and 6(p).
- NAFS can support large hops while maintain the distinguishable results.

Conclusion



1. To the best of our knowledge, we are the first to explore the possibility that simple feature smoothing without any trainable parameters could even outperform state-of-the-art GNNs; this incredible finding opens up a new direction towards efficient and scalable graph representation learning.
2. We propose NAFS, a node-adaptive feature smoothing approach along with various feature ensemble strategies, to fully exploit knowledge from both the graph structure and node features.
3. Empirical results demonstrate that NAFS performs comparably with or even outperforms the state-of-the-art GNNs, and achieves up to two orders of magnitude speedup.



Thanks