Leveraging Contrastive Learning for Enhanced Node Representations in Tokenized Graph Transformers

Jinsong Chen, Hanpeng Liu, John E. Hopcroft, Kun He*

Hopcroft Center on Computing Science,
School of Computer Science and Technology,
Huazhong University of Science and Technology, China
(*Corresponding author, brooklet60@hust.edu.cn)

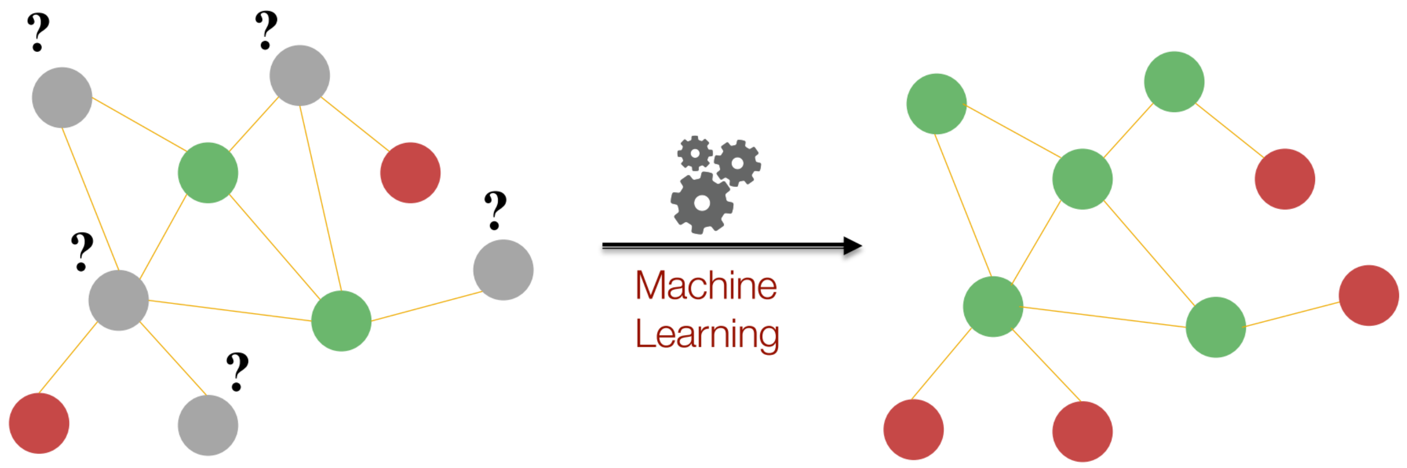
Nov. 2024

Background

- **Node Classification**

An attributed graph $G = (V, E)$, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, and the label matrix $\mathbf{Y} \in \mathbb{R}^{n \times c}$.

Given a labeled node set V_L, predict the labels of other nodes in V - V_L.



Background

- **Transformer**

Transformer layer:

Multi-head self-attention (MSA) + Feed-forward network (FFN)

MSA:

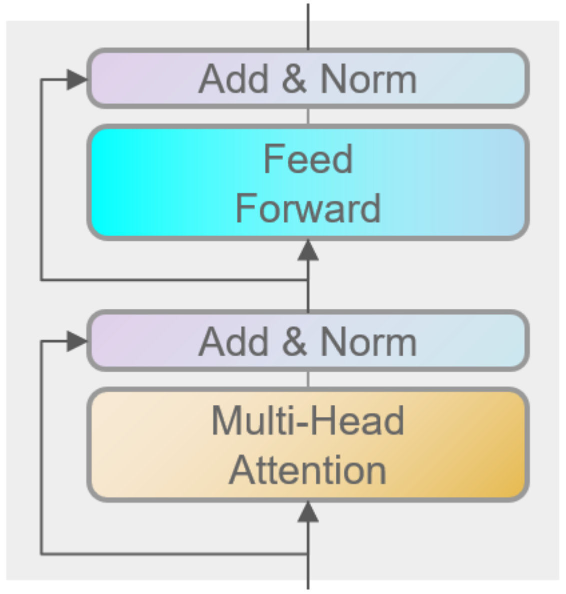
$$\mathrm{MSA}(\mathbf{H}) = \mathrm{Concat}\big(\mathrm{head}_1(\mathbf{H}), \cdots, \mathrm{head}_h(\mathbf{H})\big)\mathbf{W}^O,$$

$$\mathrm{head}_i(\mathbf{H}) = \mathrm{Attention}\big(\mathbf{H}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V\big),$$

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V}.$$

FFN:

$$\mathrm{FFN}(\mathbf{H}) = \mathrm{Linear}\Big(\sigma\big(\mathrm{Linear}(\mathbf{H})\big)\Big).$$



Vaswani A, et al. Attention is all you need. NIPS 2017.

Background

- **Graph Transformers for node classification**

Leveraging the Transformer layer to learn the node representations.

Two main categories of existing GTs:

- **Entire graph-based GTs:**

Requiring the entire graph as the model input. Performing attention calculation on all node pairs.

Involving many irrelevant nodes and introducing high training cost.

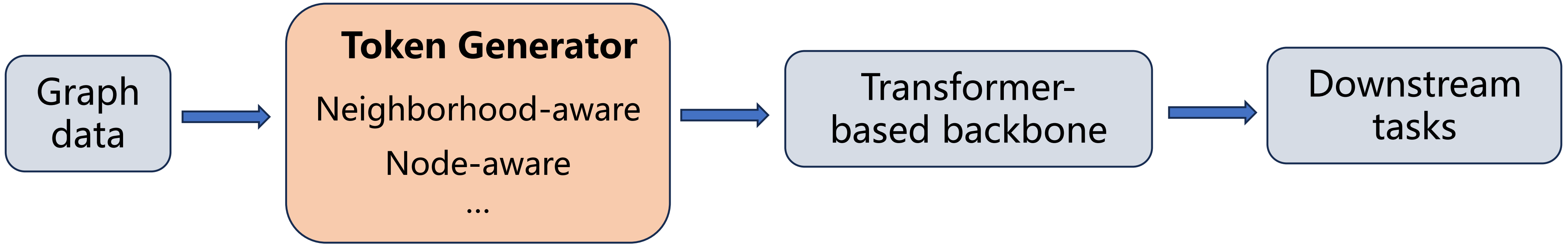
- **Tokenized GTs:**

Transforming the input graph into token sequences for feeding Transformer to learn node representations.

Focusing on necessary graph information carried by tokens and requiring low training cost.

Background

- **Tokenized Graph Transformers**



Neighborhood and node are two important elements in existing token generator.

Compared to neighborhood-aware tokens, node-aware tokens are more flexible to preserve various graph information.

Fu, et al. VCR-Graphormer: A Mini-batch Graph Transformer via Virtual Connections. ICLR 2024.

Background

- **Node-aware token generator**

▪ **Step 1: Measuring the similarity of nodes.**

Develop a function, such as cosine similarity and random walk-based strategies to calculate the similarity of each node pair.

▪ **Step 2: Node sampling**

Apply top-*k* sampling strategy to sampling nodes with high similarity as tokens to construct the token sequence.

Zhang, et al. Hierarchical Graph Transformer with Adaptive Node Sampling. NeurIPS 2022.

Motivation

- **Rethinking Node Tokenized Graph Transformer**

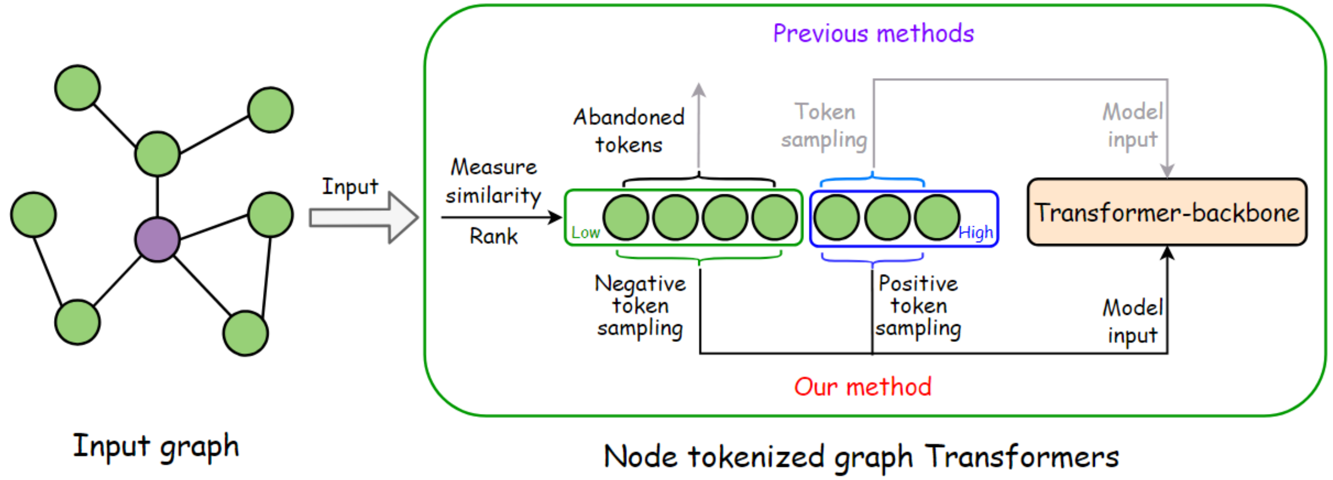


Figure 1: A toy example to illustrate the difference of the token generator between the token generator in our method and that used in the previous node tokenized graph Transformers. Previous methods only sample nodes with high similarity to construct token sequences. In contrast, our method introduces both positive and negative token sampling to preserve information carried by diverse nodes in the graph.

Most nodes are abandoned.

How to fully utilize all nodes in graph to learn node representations?

Chen, et al. Leveraging Contrastive Learning for Enhanced Node Representations in Tokenized Graph Transformers. NeurIPS 2024.

GCFormer

- **Key idea**

  Considering both high- and low-similarity nodes for model training.

- **Main steps**

  - Hybrid Token Generator:

    Generate both positive and negative token sequences.

  - Learning and Aggregation:

    Learn representations from different types of token sequences by Transformer.

  - Auxiliary Loss Function:

    Introduce contrastive learning-based loss function for constraining model training.

GCFormer

- **Hybrid Token Generator**
  - Calculating node similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$

$$\mathbf{S} = \frac{\mathbf{X}^{in} \cdot \mathbf{X}^{in^T}}{|\mathbf{X}^{in}||\mathbf{X}^{in}|}, \mathbf{X}^{in} \in \mathbb{R}^{n \times d}$$ represents the arbitrary node features

$\mathbf{X}^{in} = \mathbf{X}$ for attribute feature view, $\mathbf{X}^{in} = \widehat{\mathbf{A}}^k \mathbf{X}$ for topology feature view

  - Sampling positive token set

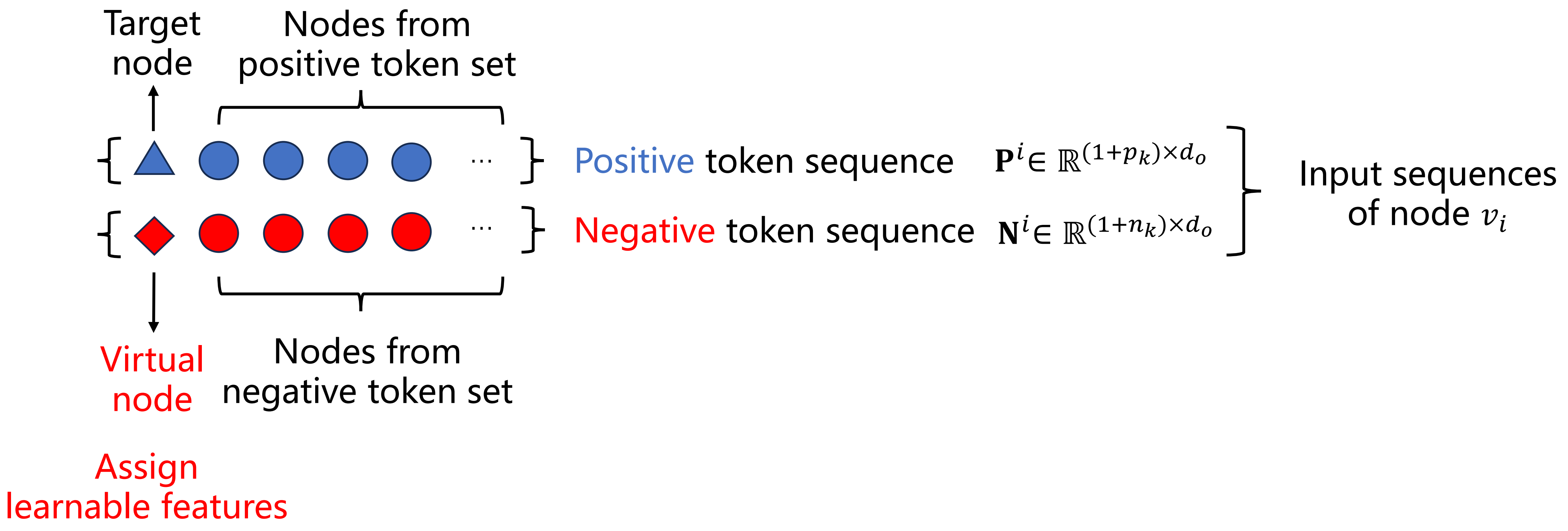
$$V_i^p = \{v_j | v_j \in \mathrm{Top}(\mathbf{S}_i)\}$$

  - Sampling negative token set

$$V_i^n = \left\{v_j \middle| v_j \in \mathrm{Sample}(V_i^r)\right\}, V_i^r = V - V_i^p$$

GCFormer

- **Token sequence construction**



Target node
Nodes from positive token set
Positive token sequence $\mathbf{P}^i \in \mathbb{R}^{(1+p_k)\times d_o}$
Negative token sequence $\mathbf{N}^i \in \mathbb{R}^{(1+n_k)\times d_o}$
Input sequences of node v_i
Virtual node
Nodes from negative token set
Assign learnable features

GCFormer

- **Transformer-based backbone**

  - Learning from Pos. $\mathbf{P}^{i^{(l)}\prime} = \mathrm{MSA}\left(\mathbf{P}^{i^{(l-1)}}\right) + \mathbf{P}^{i^{(l-1)}}, \quad \mathbf{P}^{i^{(l)}} = \mathrm{FFN}\left(\mathbf{P}^{i^{(l)}\prime}\right) + \mathbf{P}^{i^{(l)}\prime}$

  - Learning from Neg. $\mathbf{N}^{i^{(l)}\prime} = \mathrm{MSA}\left(\mathbf{N}^{i^{(l-1)}}\right) + \mathbf{N}^{i^{(l-1)}}, \quad \mathbf{N}^{i^{(l)}} = \mathrm{FFN}\left(\mathbf{N}^{i^{(l)}\prime}\right) + \mathbf{N}^{i^{(l)}\prime}$

  - Signed Aggregation $\mathbf{H}^i = \mathbf{P}_0^i - \mathbf{N}_0^i,$

Final node representation

Learning from positive tokens

Learning from negative tokens

Signed aggregation makes a distinguishable node representation

GCFormer

- **Predicting labels of nodes**

  - Learning from different feature views

$$\mathbf{Z}^{i} = \alpha \cdot \mathbf{H}^{a,i} + (1-\alpha) \cdot \mathbf{H}^{t,i},$$

$\mathbf{H}^{a,i}$ and $\mathbf{H}^{t,i}$ are representations of node v_i learning from attribute and topology feature views.

$\alpha \in [0,1]$ is a hyper-parameter to determine the contributions of different feature views.

  - Loss function

$$\mathcal{L}_{ce} = -\sum_{i \in V_l} \mathbf{Y}_i \ln \widehat{\mathbf{Y}}_i, \widehat{\mathbf{Y}}_i = \mathrm{MLP}\left(\mathbf{Z}^{i}\right),$$

GCFormer

- **Contrastive Learning-based Loss Function**

$$\mathcal{L}_{cl}(v_i) = -\log \frac{\exp\left(\mathbf{P}_0^i \cdot \hat{\mathbf{P}}^{i^{\mathrm{T}}} / \tau\right)}{\sum_{j=1}^{n_k} \exp\left(\mathbf{P}_0^i \cdot \mathbf{N}_j^{i^{\mathrm{T}}} / \tau\right)} \qquad \hat{\mathbf{P}}^i = \frac{1}{p_k} \sum_{j=1}^{p_k} \mathbf{P}_j^i$$

- **Overall Loss**

$$\mathcal{L} = \mathcal{L}_{ce} + \beta \cdot \mathcal{L}_{cl}$$

Experiments

- **Datasets**

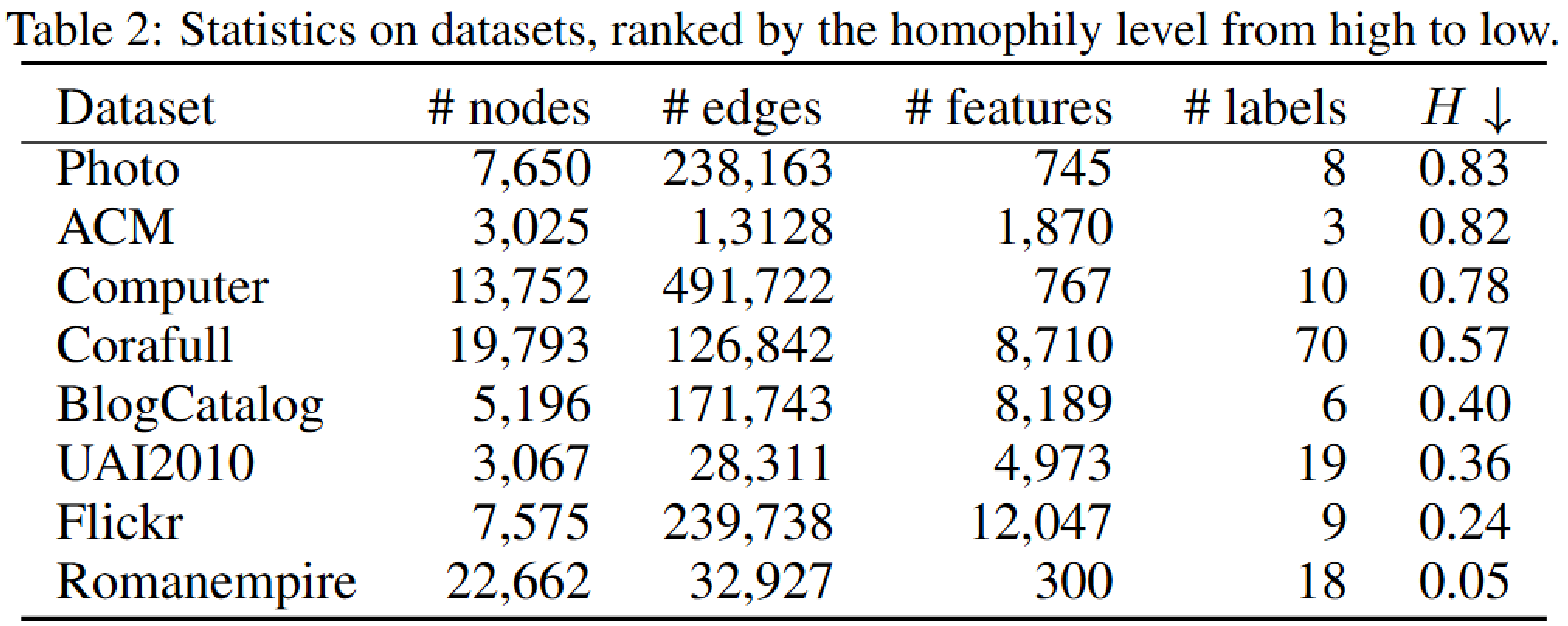


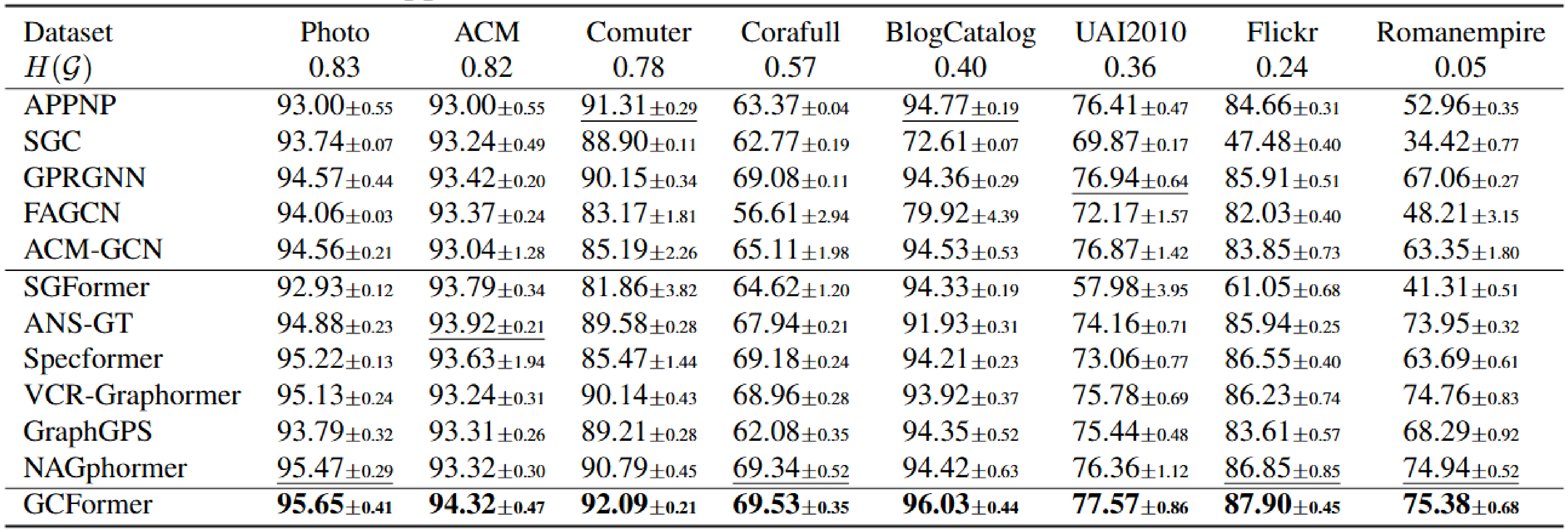
Table 2: Statistics on datasets, ranked by the homophily level from high to low.

Dataset	# nodes	# edges	# features	# labels	$H \downarrow$
Photo	7,650	238,163	745	8	0.83
ACM	3,025	1,3128	1,870	3	0.82
Computer	13,752	491,722	767	10	0.78
Corafull	19,793	126,842	8,710	70	0.57
BlogCatalog	5,196	171,743	8,189	6	0.40
UAI2010	3,067	28,311	4,973	19	0.36
Flickr	7,575	239,738	12,047	9	0.24
Romanempire	22,662	32,927	300	18	0.05

GCFormer

- **Performance comparison**

Table 1: Comparison of all models in terms of mean accuracy ± stdev (%). The best results appear in **bold**. The second results appear in underline.



Dataset	Photo	ACM	Comuter	Corafull	BlogCatalog	UAI2010	Flickr	Romanempire
$H(\mathcal{G})$	0.83	0.82	0.78	0.57	0.40	0.36	0.24	0.05
APPNP	93.00$_{\pm0.55}$	93.00$_{\pm0.55}$	91.31$_{\pm0.29}$	63.37$_{\pm0.04}$	94.77$_{\pm0.19}$	76.41$_{\pm0.47}$	84.66$_{\pm0.31}$	52.96$_{\pm0.35}$
SGC	93.74$_{\pm0.07}$	93.24$_{\pm0.49}$	88.90$_{\pm0.11}$	62.77$_{\pm0.19}$	72.61$_{\pm0.07}$	69.87$_{\pm0.17}$	47.48$_{\pm0.40}$	34.42$_{\pm0.77}$
GPRGNN	94.57$_{\pm0.44}$	93.42$_{\pm0.20}$	90.15$_{\pm0.34}$	69.08$_{\pm0.11}$	94.36$_{\pm0.29}$	76.94$_{\pm0.64}$	85.91$_{\pm0.51}$	67.06$_{\pm0.27}$
FAGCN	94.06$_{\pm0.03}$	93.37$_{\pm0.24}$	83.17$_{\pm1.81}$	56.61$_{\pm2.94}$	79.92$_{\pm4.39}$	72.17$_{\pm1.57}$	82.03$_{\pm0.40}$	48.21$_{\pm3.15}$
ACM-GCN	94.56$_{\pm0.21}$	93.04$_{\pm1.28}$	85.19$_{\pm2.26}$	65.11$_{\pm1.98}$	94.53$_{\pm0.53}$	76.87$_{\pm1.42}$	83.85$_{\pm0.73}$	63.35$_{\pm1.80}$
SGFormer	92.93$_{\pm0.12}$	93.79$_{\pm0.34}$	81.86$_{\pm3.82}$	64.62$_{\pm1.20}$	94.33$_{\pm0.19}$	57.98$_{\pm3.95}$	61.05$_{\pm0.68}$	41.31$_{\pm0.51}$
ANS-GT	94.88$_{\pm0.23}$	93.92$_{\pm0.21}$	89.58$_{\pm0.28}$	67.94$_{\pm0.21}$	91.93$_{\pm0.31}$	74.16$_{\pm0.71}$	85.94$_{\pm0.25}$	73.95$_{\pm0.32}$
Specformer	95.22$_{\pm0.13}$	93.63$_{\pm1.94}$	85.47$_{\pm1.44}$	69.18$_{\pm0.24}$	94.21$_{\pm0.23}$	73.06$_{\pm0.77}$	86.55$_{\pm0.40}$	63.69$_{\pm0.61}$
VCR-Graphormer	95.13$_{\pm0.24}$	93.24$_{\pm0.31}$	90.14$_{\pm0.43}$	68.96$_{\pm0.28}$	93.92$_{\pm0.37}$	75.78$_{\pm0.69}$	86.23$_{\pm0.74}$	74.76$_{\pm0.83}$
GraphGPS	93.79$_{\pm0.32}$	93.31$_{\pm0.26}$	89.21$_{\pm0.28}$	62.08$_{\pm0.35}$	94.35$_{\pm0.52}$	75.44$_{\pm0.48}$	83.61$_{\pm0.57}$	68.29$_{\pm0.92}$
NAGphormer	95.47$_{\pm0.29}$	93.32$_{\pm0.30}$	90.79$_{\pm0.45}$	69.34$_{\pm0.52}$	94.42$_{\pm0.63}$	76.36$_{\pm1.12}$	86.85$_{\pm0.85}$	74.94$_{\pm0.52}$
GCFormer	**95.65**$_{\pm0.41}$	**94.32**$_{\pm0.47}$	**92.09**$_{\pm0.21}$	**69.53**$_{\pm0.35}$	**96.03**$_{\pm0.44}$	**77.57**$_{\pm0.86}$	**87.90**$_{\pm0.45}$	**75.38**$_{\pm0.68}$

GCFormer outperforms advanced GTs as well as representative GNNs on all datasets.

GCFormer

- **Study of token sampling size**

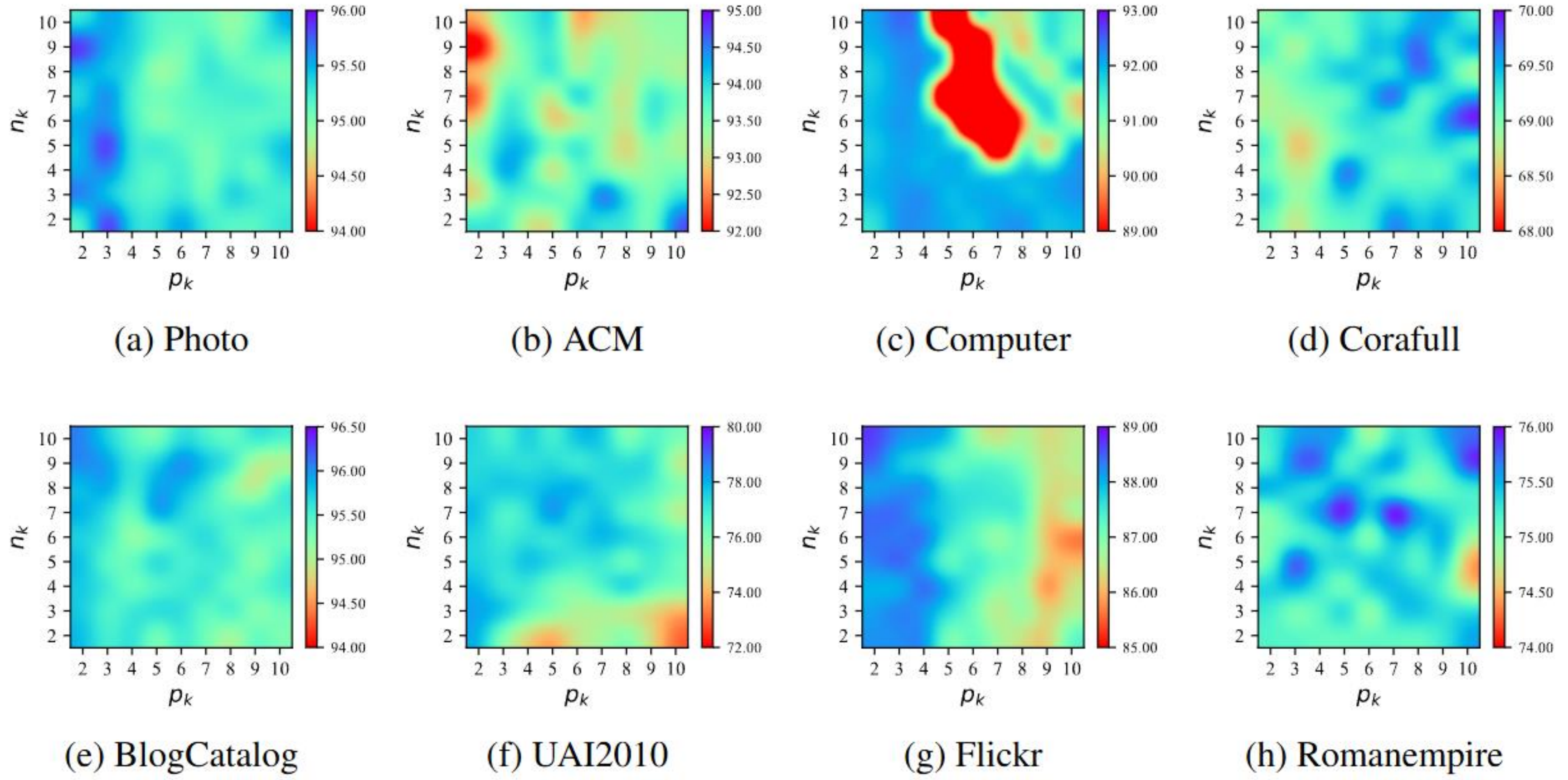


Figure 2: Performance of GCFormer with different sampling sizes on all datasets.

GCFormer

- **Study of aggregation weight**

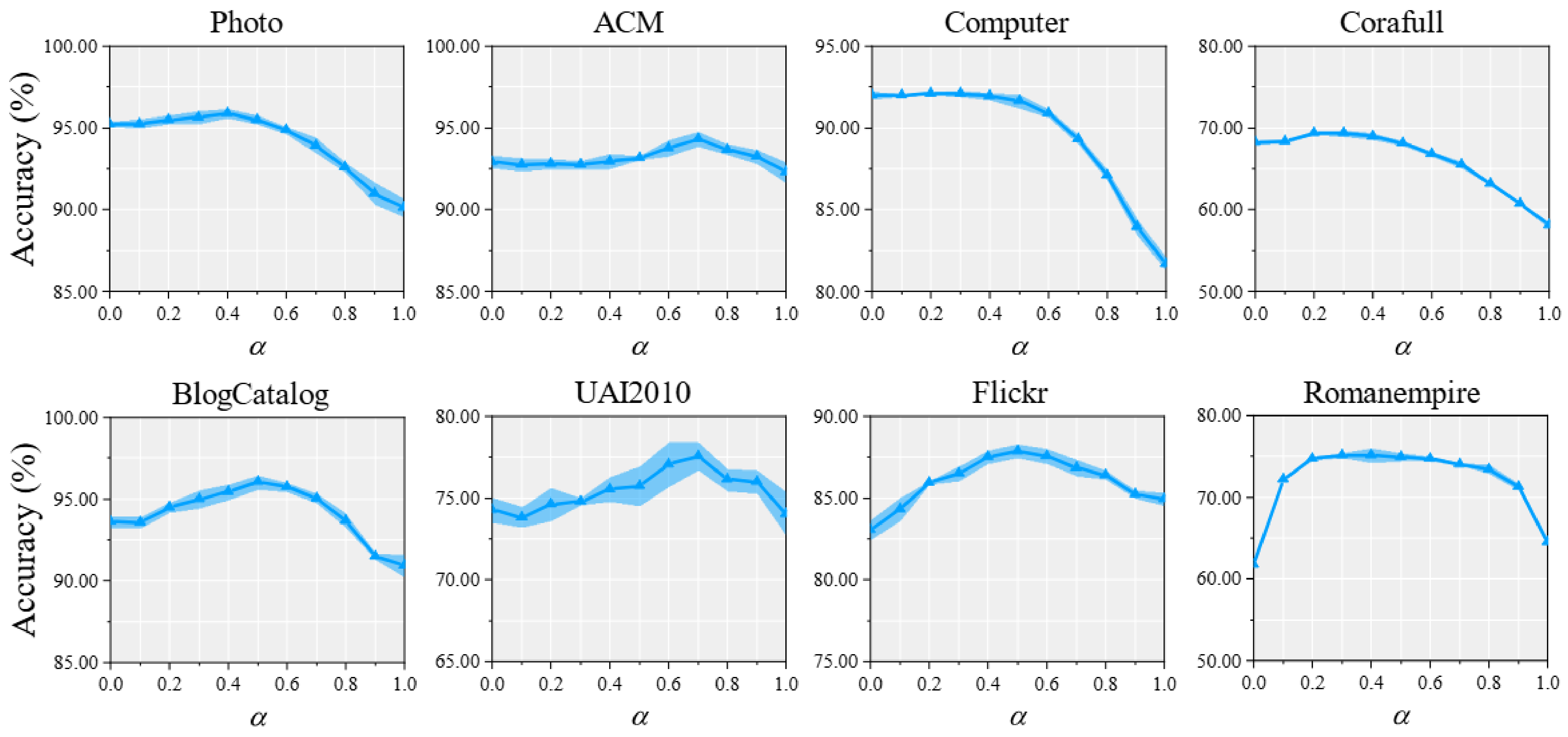


Figure 3: Performance of GCFormer with different α on all datasets.

Future work

- **Rethinking GCFormer**

The main limitation of GCFormer is the unified sampling strategy for different types of graphs.

Experimental results show that the performance of GCFormer is sensitive to the sampling size on different graphs.

The phenomenon implies that an adaptive sampling strategy is required to improve the performance and stability of GCFormer on diverse graphs.

End

Thanks for your attention!