Rethinking Tokenized Graph Transformers for Node Classification

Jinsong Chen, Chenyang Li, Gaichao Li, 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. 2025

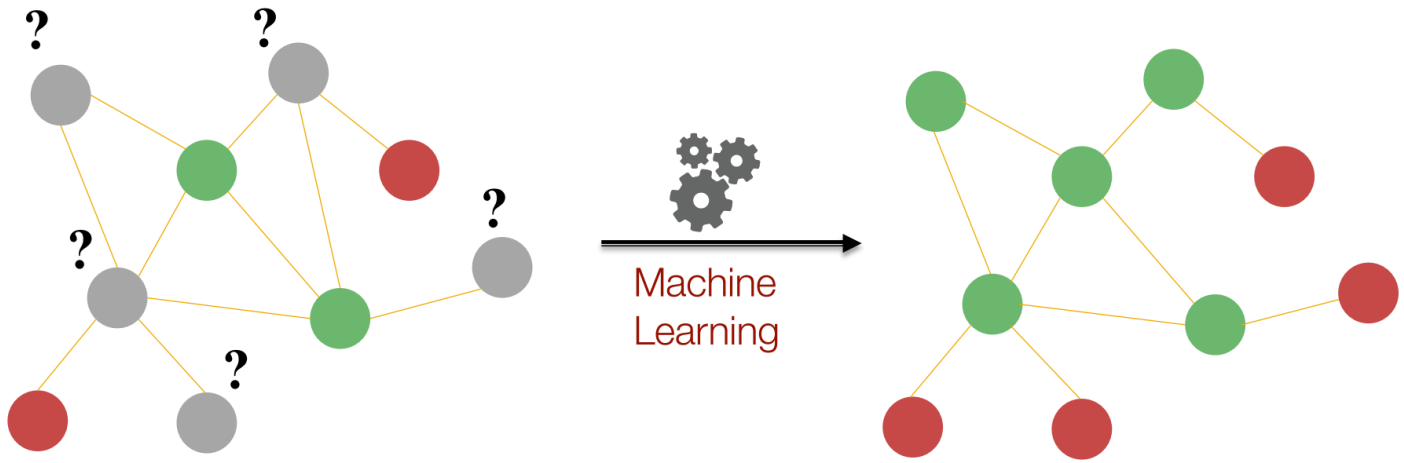
Background

- **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

- **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:**

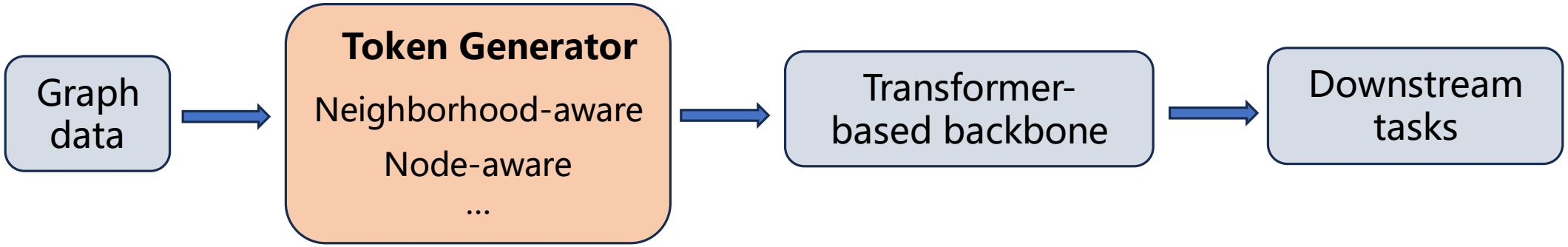
Transforming the input graph into token sequences for feeding Transformer to learn node representations.

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

Jinsong Chen, Siyu Jiang, Kun He. NTFormer: A Composite Node Tokenized Graph Transformer for Node Classification. arXiv, 2024.

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.

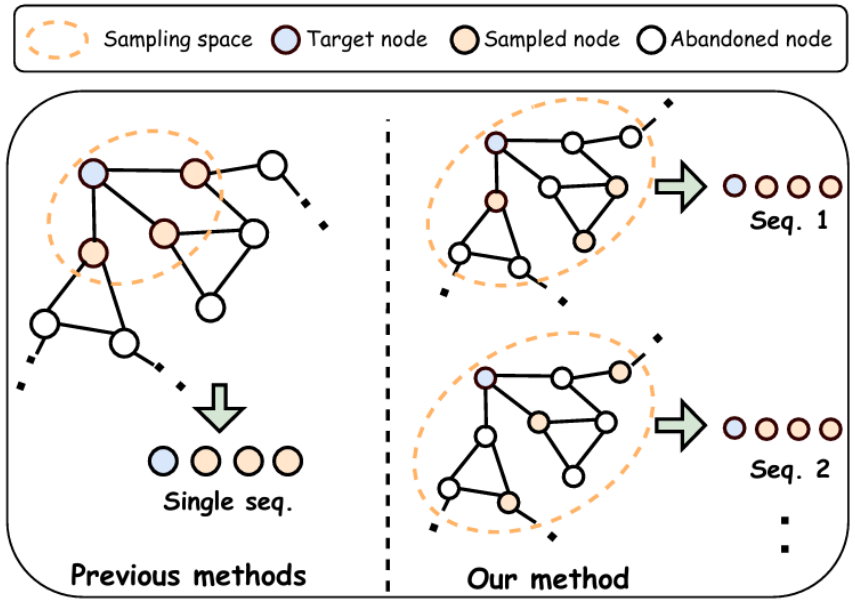
  - **Step 2: Node sampling**

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 Tokenized Graph Transformers**



Using the Top-K sampling is equivalent to selecting only first-order neighbors on the constructed K-NN graph.

How to fully utilize neighbors beyond first-order on constructed K-NN graph?

Figure 1: The toy example of token generation on the constructed k-NN graph. Previous methods only focus on 1-hop neighborhood to construct a single token sequence. While our method can flexibly select tokens from multi-hop neighborhoods to generate diverse token sequences.

Jinsong Chen, et al. Rethinking Tokenized Graph Transformers for Node Classification. NeurIPS 2025.

SwapGT

- **Key idea**

  Using token swapping to enlarge sampling space and construct multiple token sequences for model training.

- **Main steps**

  - Token Swapping:

    Swap semantically relevant tokens in different token sequences to generate new sequences.

  - Multi-sequence learning:

    Learn node representations from multiple token sequences by Transformer.

  - Center Alignment Loss:

    Introduce auxiliary alignment loss function for constraining model training.

SwapGT

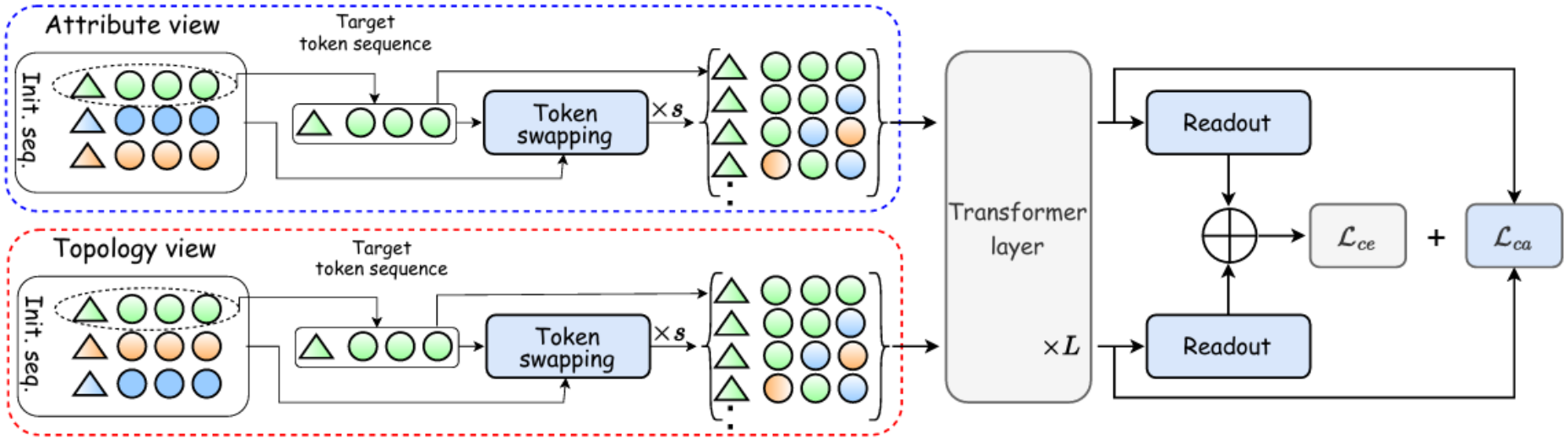


Figure 2: The overall framework of SwapGT. First, we generate the initial token sequences from both the attribute view and topology view. Then, we utilize the proposed token swapping operation to generate new token sequences for each target node. These generated token sequences are then fed into a Transformer-based backbone to learn node representations and generate predicted labels. Additionally, a center alignment loss is adopted to further constrain the representations extracted from different token sequences.

- **Token Sampling**
  - 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 initial token set

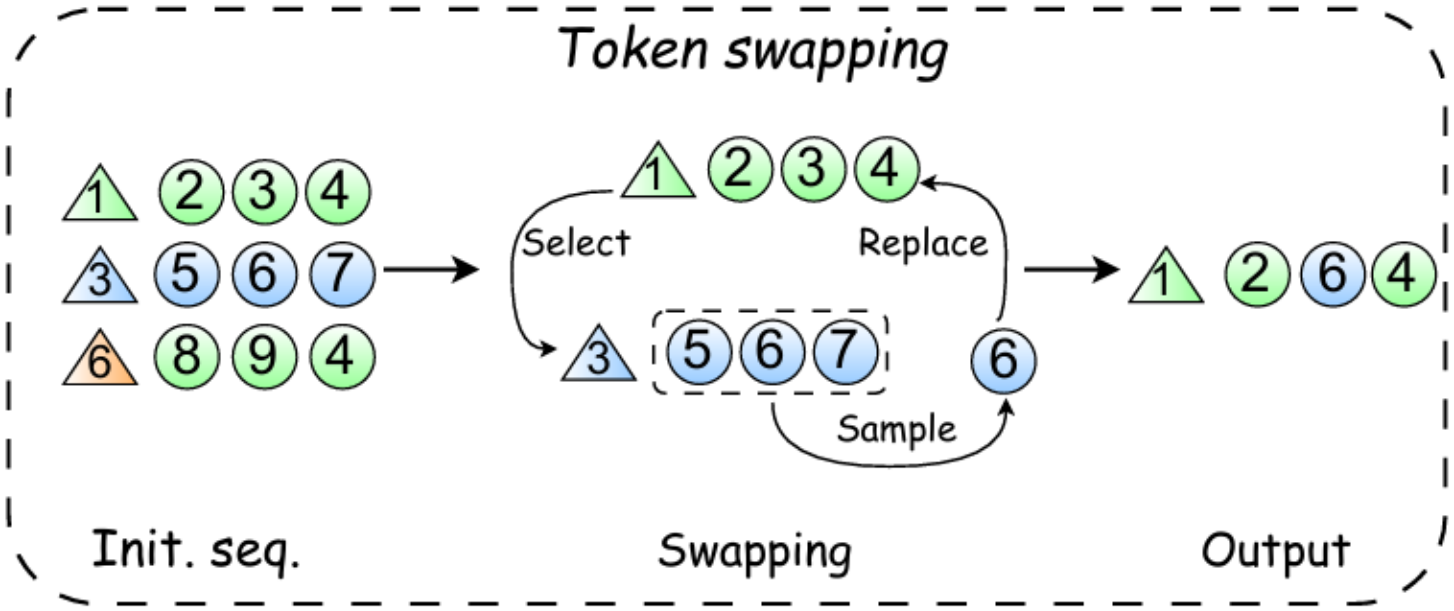
$$N_i = \{v_j | v_j \in \mathrm{Top}(\mathbf{S}_i, k)\}$$

SwapGT

- **Token Swapping**

Randomly select tokens from node $\mathbf{v}_i's$ token set N_i, and perform swapping operation to form a new token set N_i'.

$$N_i' = \{\delta(N_j) | v_j \in N_i\}$$



Swapping once can extend sampling space on K-NN graph from 1-order to 2-order.

SwapGT

- **Token Swapping**

Algorithm 1 The Token Swapping Algorithm

Input: Sampled token set of all nodes $N \in \mathbb{R}^{n \times k}$; Target node v_i; Probability p; Swapping times t
Output: The new token set $N_i' \in \mathbb{R}^{1 \times k}$ of v_i

Initialize $N_i' = N_i$;
for $t_0 = 1$ **to** t **do**
    Initialize $N^{new} = \{\}$;
    for $v_j \in N_i'$ **do**
        if $random(0, 1) > p$ **then**
            $N^{new} = N^{new} \cup \{v_j\}$;
        else
            $v^{new} = \zeta(N_j)$;
            $N^{new} = N^{new} \cup \{v^{new}\}$;
        end if
    end for
    $N_i' = N^{new}$;
end for
return N_i'

Totally, swap t times to expand the sampling space to (t+1) order.

SwapGT

- **Token sequence construction**

For node v_i, one token set N_i can generate one token sequence $\mathbf{Z}_i^{in}$ as:

$$\mathbf{Z}_i^{in} = [\mathbf{X}_i, \mathbf{X}_{N_{i,0}}, \ldots, \mathbf{X}_{N_{i,k-1}}]$$

By performing token swapping Algorithm for s times, we have 1+s token sets for v_i. We combine all token sequences to obtain the mode input $\mathbf{Z}_i \in \mathbb{R}^{(1+s)\times(1+k)\times d}$

For attribute and topology, we have $\mathbf{Z}_i^A$ and $\mathbf{Z}_i^T$ respectively.

SwapGT

- **Multi-sequence learning**
  - Transformer-based backbone:

$$\mathbf{Z}_i'^{A,(l)} = \mathrm{MSA}\left(\mathbf{Z}_i^{A,(l-1)}\right) + \mathbf{Z}_i^{A,(l-1)}$$

$$\mathbf{Z}_i^{A,(l)} = \mathrm{FFN}\left(\mathbf{Z}_i'^{A,(l)}\right) + \mathbf{Z}_i'^{A,(l)}$$

  - Readout function based on multi token sequences:

$$\mathbf{Z}_i^{A,F} = \mathbf{Z}_0^{A,i} \| \left(\frac{1}{s}\sum_{j=1}^{s} \mathbf{Z}_j^{A,i}\right)$$

SwapGT

- **Multi-sequence learning**
  - Fusing node representations in attribute and topology view:

$$\mathbf{Z}_i^F = \alpha \cdot \mathbf{Z}_i^{A,F} + (1-\alpha) \cdot \mathbf{Z}_i^{T,F}$$

  - Predicting labels of nodes:

$$\hat{\mathbf{Y}}_i = \mathrm{MLP}\left(\mathbf{Z}_i^F\right)$$

$$\mathcal{L}_{ce} = -\sum_{i \in V_L} \mathbf{Y}_i \ln \hat{\mathbf{Y}}_i$$

SwapGT

- **Multi-sequence Center Alignment Loss**
  - Center representation of multi sequences:

$$\mathbf{Z}_c^i = \frac{1}{s+1}\sum_{j=0}^{s}\mathbf{Z}_j^i$$

  - Center alignment loss:

$$\mathrm{CAL}\left(\mathbf{Z}^i, \mathbf{Z}_c^i\right) = 1 - \frac{1}{s+1}\sum_{j=0}^{s}\mathrm{Cosine}\left(\mathbf{Z}_j^i, \mathbf{Z}_c^i\right)$$

  - Overall loss function:

$$\mathcal{L}_{ca} = \mathrm{CAL}\left(\mathbf{Z}^{A,i}, \mathbf{Z}_c^{A,i}\right) + \mathrm{CAL}\left(\mathbf{Z}^{T,i}, \mathbf{Z}_c^{T,i}\right)$$

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \cdot \mathcal{L}_{ca}$$

Experiments

- **Datasets**

Table 3: Statistics of datasets, ranked by the homophily level.

Dataset	# nodes	# edges	# features	# labels	$\mathcal{H} \downarrow$
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
Citeseer	3,327	4,552	3,703	6	0.74
WikiCS	11,701	216,123	300	10	0.66
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

We use two strategies to split datasets:

Dense splitting: 50%/25%/25% and Sparse splitting: 2.5%/2.5%/95%

SwapGT

- **Performance comparison (dense splitting)**

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

Dataset	Photo	ACM	Computer	Citeseer	WikiCS	BlogCatalog	UAI2010	Flickr
$\mathcal{H}$	0.83	0.82	0.78	0.74	0.66	0.40	0.36	0.24
SGC	93.74±0.07	93.24±0.49	88.90±0.11	76.81±0.26	76.67±0.19	72.61±0.07	69.87±0.17	47.48±0.40
APPNP	94.98±0.41	93.00±0.55	91.31±0.29	77.52±0.22	81.96±0.14	94.77±0.19	77.41±0.47	84.66±0.31
GPRGNN	94.57±0.44	93.42±0.20	90.15±0.34	77.59±0.36	82.43±0.29	94.36±0.29	76.94±0.64	85.91±0.51
FAGCN	94.06±0.03	93.37±0.24	83.17±1.81	76.19±0.62	79.89±0.93	79.92±4.39	72.17±1.57	82.03±0.40
BM-GCN	95.10±0.20	93.68±0.34	91.28±0.96	77.91±0.58	83.90±0.41	94.85±0.42	77.39±1.13	83.97±0.87
ACM-GCN	94.56±0.21	93.04±1.28	85.19±2.26	77.62±0.81	83.95±0.41	94.53±0.53	76.87±1.42	83.85±0.73
NAGphormer	95.47±0.29	93.32±0.30	90.79±0.45	77.68±0.73	83.61±0.28	94.42±0.63	76.36±1.12	86.85±0.85
SGFormer	92.93±0.12	93.79±0.34	81.86±3.82	77.86±0.76	79.65±0.31	94.33±0.19	57.98±3.95	61.05±0.68
Specformer	95.22±0.13	93.63±1.94	85.47±1.44	77.96±0.89	83.74±0.62	94.21±0.23	73.06±0.77	86.55±0.40
VCR-Graphormer	95.38±0.51	93.11±0.79	90.47±0.58	77.21±0.65	80.82±0.72	94.19±0.17	76.08±0.52	85.96±0.55
CoBFormer	95.29±0.35	93.82±0.58	90.21±0.89	77.74±0.72	83.28±0.68	93.98±0.72	76.85±0.69	86.84±0.78
PolyFormer	95.45±0.21	94.27±0.44	90.87±0.74	78.03±0.86	83.79±0.75	95.08±0.43	77.92±0.82	87.01±0.57
SwapGT	**95.92±0.18**	**94.98±0.41**	**91.73±0.72**	**78.49±0.95**	**84.52±0.63**	**95.93±0.56**	**79.06±0.73**	**87.56±0.61**

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

SwapGT

- **Performance comparison (sparse splitting)**

Table 2: Comparison of all models in terms of mean accuracy ± stdev (%) under sparse splitting. The best results appear in **bold**. The second results appear in underline.

Dataset	Photo	ACM	Computer	Citeseer	WikiCS	BlogCatalog	UAI2010	Flickr
$\mathcal{H}$	0.83	0.82	0.78	0.74	0.66	0.40	0.36	0.24
SGC	91.90±0.35	89.57±0.28	86.79±0.19	66.41±0.59	74.99±0.19	71.23±0.06	51.61±0.41	39.43±0.50
APPNP	92.24±0.28	89.91±0.89	87.64±0.39	66.70±0.11	77.42±0.31	81.76±0.38	61.65±0.71	71.39±0.62
GPRGNN	92.13±0.32	89.47±0.90	86.38±0.44	66.50±0.62	77.59±0.49	84.57±0.35	58.75±0.75	71.89±0.89
FAGCN	92.02±0.18	88.47±0.31	83.99±1.95	64.54±0.66	75.21±0.84	76.38±0.82	54.67±0.96	63.68±0.72
BM-GCN	91.19±0.39	90.11±0.60	86.14±0.51	66.11±0.47	77.39±0.37	84.05±0.54	57.51±1.14	60.82±0.76
ACM-GCN	91.71±0.64	89.68±0.45	86.64±0.59	64.85±1.19	77.68±0.57	77.17±1.34	56.05±2.11	64.58±1.53
NAGphormer	91.65±0.80	89.73±0.48	85.31±0.65	63.66±1.68	76.93±0.75	79.19±0.41	58.36±1.01	67.48±1.04
SGFormer	90.13±0.56	88.03±0.60	80.07±0.21	62.41±0.94	74.69±0.52	78.15±0.69	50.19±1.72	51.01±1.05
Specformer	90.57±0.55	88.20±1.05	85.55±0.63	62.64±1.54	75.24±0.71	79.75±1.29	57.42±1.06	56.94±1.48
VCR-Graphormer	91.39±0.75	86.81±0.84	85.06±0.64	57.61±0.60	72.81±1.44	74.90±1.18	56.43±1.10	50.93±1.12
CoBFormer	91.21±0.62	89.18±0.89	85.06±0.79	63.82±1.34	76.28±1.28	79.44±0.98	58.23±0.82	66.94±1.18
PolyFormer	91.52±0.78	89.83±0.62	85.75±0.78	64.77±1.27	75.12±1.16	81.02±0.81	58.89±0.77	67.85±1.43
SwapGT	**92.93**±0.26	**90.92**±0.69	**88.14**±0.52	**69.91**±1.02	**78.11**±0.83	**88.11**±0.58	**63.96**±1.09	**72.16**±1.19

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

SwapGT

- **Study on center alignment loss**

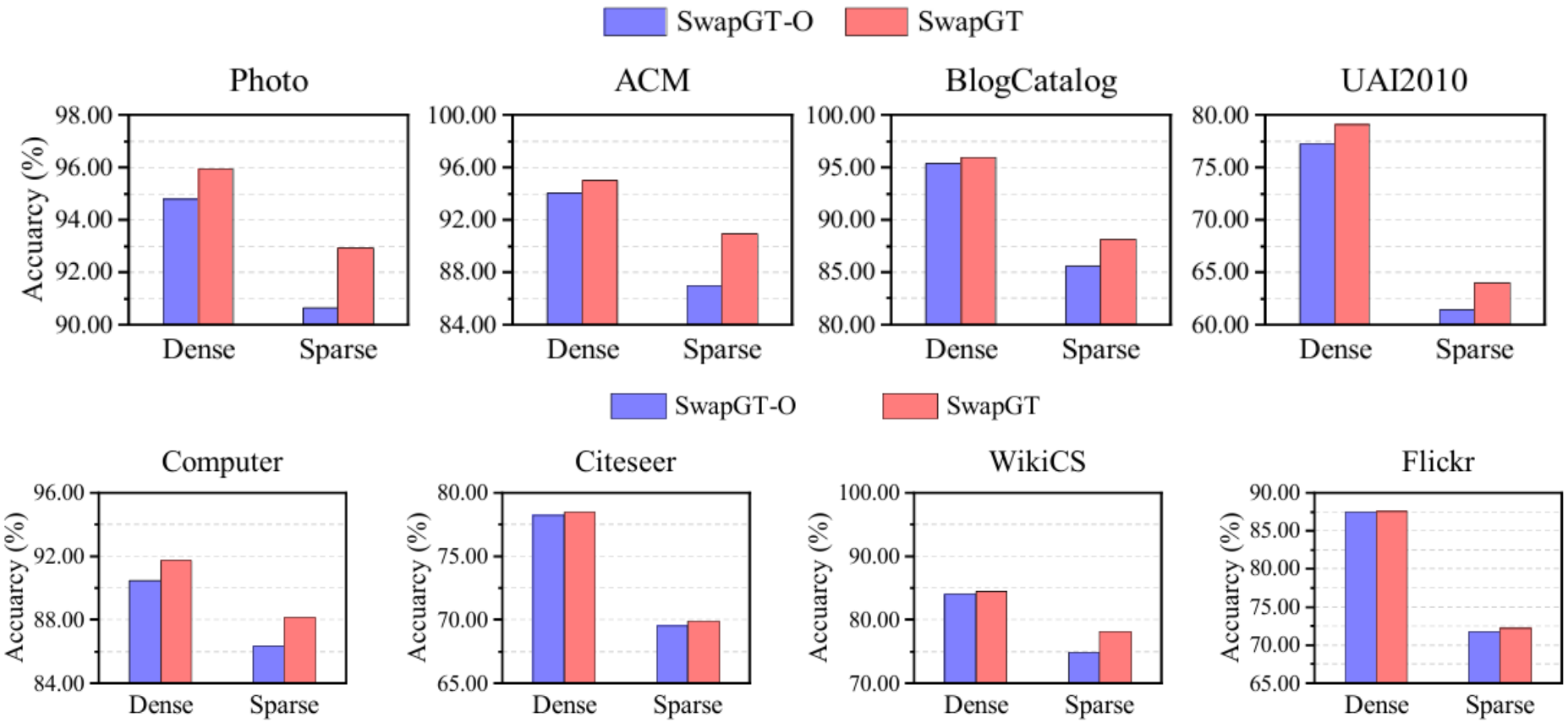


Figure 9: Performances of SwapGT with or without the center alignment loss.

SwapGT

- **Study on token sequence generation**

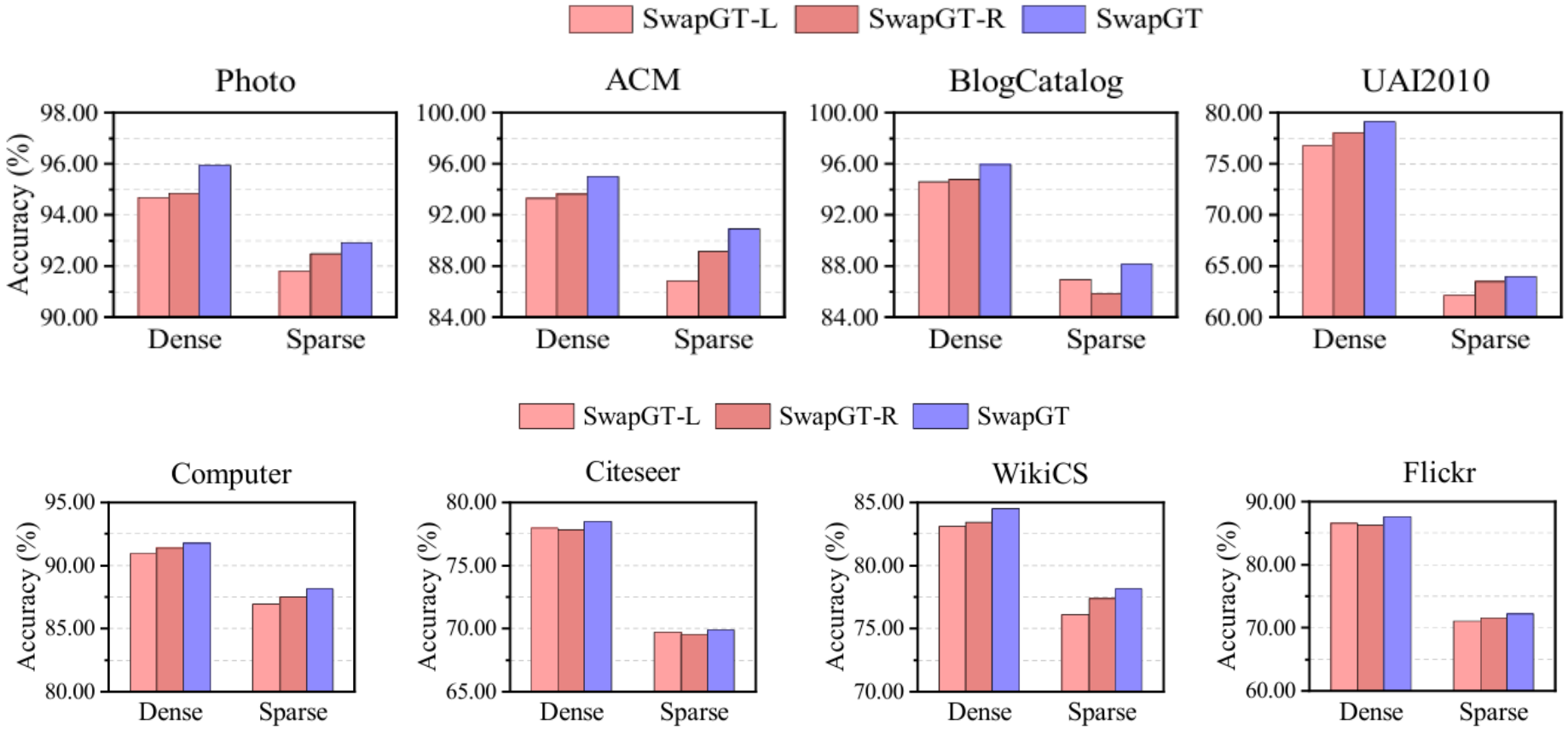


Figure 10: Performances of SwapGT with different token sequence generation strategies.

SwapGT

- **Study on token swapping times t**

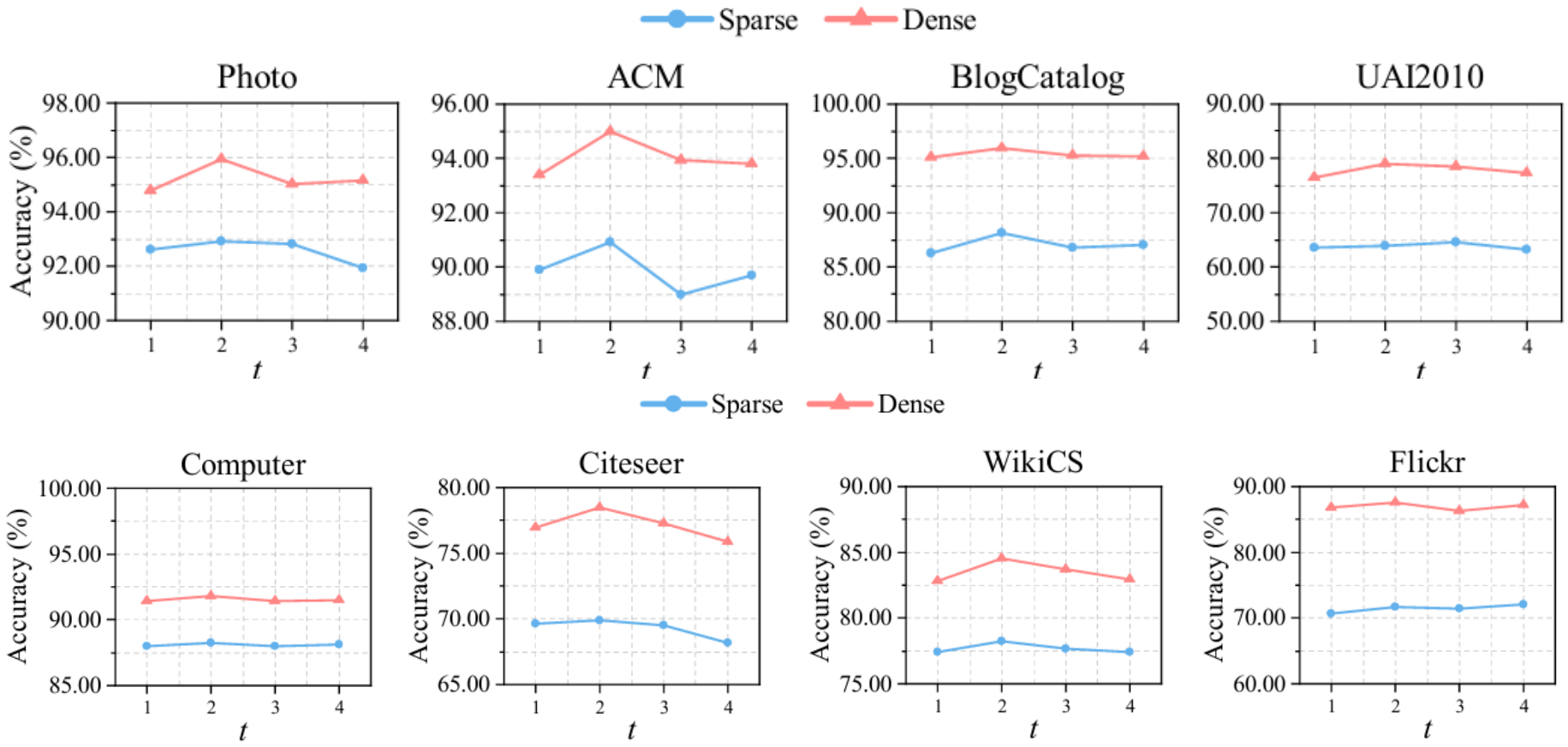


Figure 13: Performances of SwapGT with varying t.

Future work

- **Rethinking SwapGT**

A potential limitation in SwapGT could be that SwapGT applies a uniform swapping probability ρ to all tokens in the sequence.

A hierarchical probability swapping framework may be a better solution, e.g., the top k/3 nodes are assigned higher swapping probabilities, while others in top-k range receive lower probabilities.

This refinement could mitigate noise interference.

End

Thanks for your attention!