ADAPTIVE SKELETON COARSENING FOR TEXT-ATTRIBUTED GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-attributed graphs combine relational structure with rich node text, but their large scale and high-dimensional text semantics make downstream graph learning costly. Existing graph compression methods are primarily designed around topology or generic node attributes and typically do not treat pretrained text semantics as a primary signal for compression decisions, limiting their ability to exploit the rich semantics encoded in node text. We present Adaptive Skeleton Coarsening (ASC), a semantic-first framework that uses pretrained text embeddings as the primary compression signal and structural descriptors as complementary signals. A joint encoder maps these signals into a shared assignment space, where ASC adaptively learns sparse soft node-to-anchor assignments primarily through semantic reconstruction, together with an auxiliary graph-level property prediction objective. In downstream evaluation, the compressed graph replaces the original graph during model training. At a compression ratio of $r = 0.1$, ASC achieves the highest mean classification accuracy on all 11 benchmarks, while substantially reducing the peak memory consumption and training time of downstream GCNs on the evaluated graphs. Under extreme compression at $r = 0.01$, ASC still maintains robust and superior classification performance.

1 INTRODUCTION

Text-attributed graphs are widely used to model citation networks, product relations, web pages, and social interactions, where each node is associated with rich textual content (Feng et al., 2024). Recent advances in pretrained text encoders have made it possible to transform node texts into high-quality semantic embeddings, substantially improving graph learning on such data (He et al., 2024; Zolnai-Lucas et al., 2024; Wu et al., 2025). However, combining dense semantic features with large-scale graph structures also makes repeated graph learning, especially graph neural network (GNN) training, increasingly expensive in both memory and computation. Sampling methods reduce the cost of individual training runs (Chiang et al., 2019; Zeng et al., 2020); hierarchical support graphs and virtual or fractal nodes improve information flow without replacing the original graph (Vonessen et al., 2024; Southern et al., 2025; Choi et al., 2026). ASC instead constructs a smaller graph for downstream training.

Graph compression constructs a smaller attributed graph for downstream training. Existing coarsening methods mainly target spectral properties, graph cuts, or local connectivity (Hashemi et al., 2024; Loukas, 2019; Jin et al., 2020), while skeleton-style methods select structurally representative nodes (Cao et al., 2024). These criteria do not explicitly prioritize pretrained text semantics during assignment. Such embeddings provide a useful signal for assessing semantic compatibility between nodes and candidate representatives. Under aggressive compression, fewer representatives must summarize the same set of nodes, motivating ASC to use semantic reconstruction as the primary objective for learning assignments.

Graph condensation instead optimizes synthetic training signals for specific tasks (Jin et al., 2022b; Gupta et al., 2025), which can require substantial optimization. We seek a compressed graph that balances predictive accuracy and construction cost while serving as a fixed training input for different downstream models.

We propose Adaptive Skeleton Coarsening (ASC), a semantic-first compression framework for text-attributed graphs. ASC first selects semantic medoid anchors from pretrained text embeddings using

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

spherical K-means, then learns sparse soft assignments from original nodes to these anchors. In this process, frozen pretrained text embeddings serve as the primary assignment signal, while structural descriptors provide complementary evidence. The learned assignment matrix aggregates semantic features and contracts the original adjacency into a weighted attributed skeleton graph. To provide additional graph-level supervision, ASC incorporates graph-level property prediction as an auxiliary training objective.

Our contributions are summarized as follows:

- We formulate text-attributed graph compression as a semantic-first anchor-based coarsening problem, explicitly using pretrained text semantics to guide node assignment while incorporating structural descriptors as complementary evidence.
- We design a trainable sparse soft-assignment mechanism that constructs a reusable weighted attributed skeleton graph, with semantic reconstruction as the primary objective and graph-level property prediction as an auxiliary objective.
- ASC achieves the highest mean accuracy on all 11 benchmarks at $r = 0.1$, reduces downstream GCN time and memory, and has higher mean accuracy than UGC, AH-UGC, and Graph-Skeleton at $r = 0.01$.

2 Related Work

2.1 Graph Compression

Classical graph coarsening targets cuts, spectral properties, or local variation (Dhillon et al., 2007; Loukas, 2019; Jin et al., 2020). Feature-aware methods further incorporate node attributes or learned representations: FGC jointly optimizes coarse features and structure, Cohen and Talmon preserve inner products between feature signals, and TACO uses learned representation proximity to guide coarsening (Kumar et al., 2023; Cohen & Talmon, 2025; Han et al., 2024). UGC and AH-UGC address broader compression settings, while Graph-Skeleton and SGBGC use target-node or label-guided aggregation (Kataria et al., 2024; 2025; Cao et al., 2024; Xia et al., 2025). DiffPool learns hierarchical soft assignments within an end-to-end graph-prediction model (Ying et al., 2018). Joly and Keriven study message passing on coarsened graphs (Joly & Keriven, 2024). For text-attributed graphs, ASC exports a sparse attributed graph for downstream training, using pretrained text embeddings to guide node-to-anchor assignments.

Graph condensation learns synthetic training sets by matching gradients, receptive-field distributions, or training trajectories (Gao et al., 2025; Jin et al., 2022b;a; Liu et al., 2022; Zheng et al., 2023). These objectives can couple condensation to the downstream task or model and require substantial optimization.

2.2 Text-Attributed Graph Learning

Text-attributed graph learning combines natural-language node content with relational structure for graph prediction tasks (Feng et al., 2024). Existing approaches differ mainly in how these two information sources are integrated. Model-level approaches either couple Transformer-based text encoding with graph aggregation or alternate between language and graph models in iterative frameworks (Yang et al., 2021; Zhao et al., 2023). Representation-level approaches improve node embeddings through network-aware pretraining, label-free LLM supervision, parameter-efficient adaptation, contrastive objectives, or generated explanations (Jin et al., 2023; Chen et al., 2024; Zhu et al., 2024; He et al., 2024; Zolnai-Lucas et al., 2024; Fang et al., 2024).

The benefits depend on the text, graph structure, and downstream setting (Feng et al., 2024; Wu et al., 2025). These methods mainly improve representations or predictors while retaining the original graph. ASC instead coarsens the graph through an explicit sparse node-to-anchor mapping, guided by pretrained semantics and complementary structural descriptors, to support reusable downstream training.

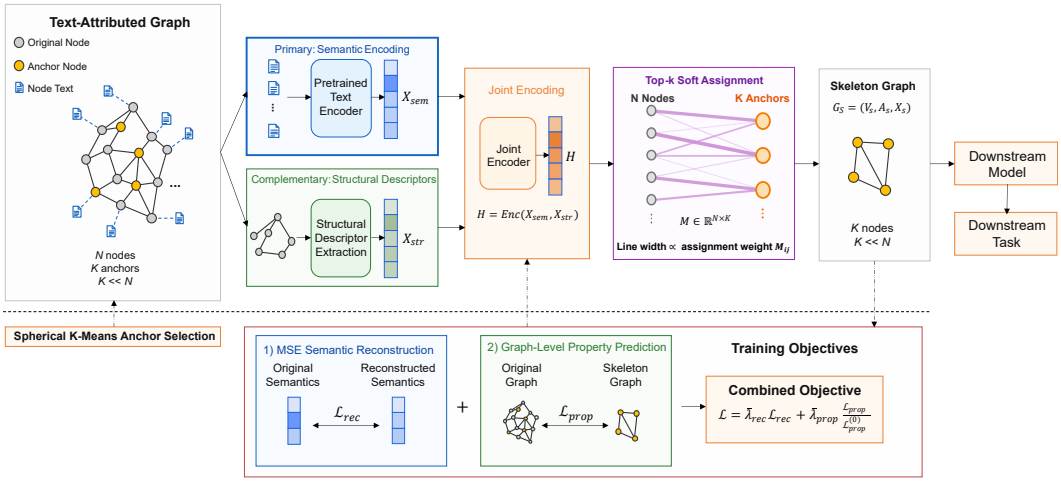


Figure 1: Overview of ASC. Frozen text embeddings provide the main semantic signal, structural descriptors provide complementary assignment signals, and graph-level property prediction on $\mathbf{A}_s^{\text{pre}}$ supplies an auxiliary training signal. ASC learns a sparse top-k assignment from original nodes to semantic medoid anchors, constructs the skeleton graph by aggregating node features and contracting edges through the assignment matrix, and optimizes semantic reconstruction together with auxiliary graph-level property prediction. The resulting skeleton graph is then used by standard GNNs for node classification.

3 Methodology

3.1 Overview

ASC couples semantic reconstruction with sparse graph construction: it learns which nodes can share representatives, enriches them with assigned context, and controls the resulting edge expansion.

Let $G = (\mathcal{V}, \mathbf{A}, \{t_i\}_{i=1}^N)$ be a text-attributed graph, where $N = |\mathcal{V}|$, $\mathcal{V}$ is the node set, t_i is the raw text associated with node i, and $\mathbf{A} \in \mathbb{R}_{\geq 0}^{N \times N}$ is the weighted adjacency matrix. We treat input graphs as undirected and store the two directions separately in COO format; self-loops are added by the downstream GNN. A frozen text encoder produces semantic features $\mathbf{X}_{\text{sem}} \in \mathbb{R}^{N \times d}$, where d is the semantic feature dimension. The structural descriptor matrix $\mathbf{X}_{\text{str}} \in \mathbb{R}^{N \times d_0}$ contains Laplacian positional encodings, local structural statistics, and Louvain community descriptors (Dwivedi et al., 2022; Page et al., 1999; Blondel et al., 2008); d_0 denotes its input dimension. All structural channels are pre-computed and standardized, with their exact definitions deferred to the appendix.

We define the node compression ratio as $r = K/N$. Given a target ratio r, ASC sets $K = \max(1, \lfloor rN \rfloor)$ and clusters the pretrained semantic features with spherical K-means, selecting the node nearest each cluster center as an anchor $s_j \in \mathcal{S} = \{s_1, \ldots, s_K\}$. Spherical K-means aligns anchor selection with the cosine-based assignment metric, while choosing medoids keeps anchors as observed nodes rather than synthetic cluster centers. Alternative selectors are evaluated in Appendix D.1. ASC then learns a sparse soft assignment $\mathbf{M} \in \mathbb{R}_{\geq 0}^{N \times K}$ from original nodes to anchors, where M_{ij} is the assignment weight from node i to anchor s_j. The resulting weighted attributed skeleton graph is $G_s = (\mathcal{V}_s, \mathbf{A}_s, \mathbf{X}_s)$, with $|\mathcal{V}_s| = K$, $\mathbf{A}_s \in \mathbb{R}_{\geq 0}^{K \times K}$, and $\mathbf{X}_s \in \mathbb{R}^{K \times d}$ (Figure 1). Unselected nodes contribute to skeleton features and weighted edges through their assignments. Node labels are not used for anchor selection, assignment learning, or skeleton construction.

For each input graph and target ratio, ASC exports a skeleton graph G_s and assignment matrix $\mathbf{M}$. Both remain fixed when training different downstream models on that graph; Section 4.1 describes the evaluation protocol.

3.2 Semantic-First Joint Encoding and Sparse Assignment

We first map semantic features and structural descriptors into a shared assignment space:

$$\begin{aligned} \mathbf{X}'_{\mathrm{str}} &= \mathrm{Lift}(\mathbf{X}_{\mathrm{str}}) \in \mathbb{R}^{N\times d_t}, \\ \mathbf{H} &= \mathrm{Enc}([\mathrm{LN}(\mathbf{X}_{\mathrm{sem}}); \mathbf{X}'_{\mathrm{str}}]) \in \mathbb{R}^{N\times d_h}, \end{aligned} \tag{1}$$

where $[\cdot;\cdot]$ denotes concatenation, $\mathrm{LN}(\bullet)$ denotes LayerNorm, $\mathrm{Lift}(\bullet)$ projects the structural descriptors, and d_t and d_h are the lifted structural-descriptor dimension and shared assignment dimension, respectively. $\mathrm{Enc}(\bullet)$ is a two-layer joint encoder with LayerNorm and Gaussian error linear unit (GELU) activations. The joint encoder learns the geometry used for assignment, while skeleton features and reconstruction targets remain in the pretrained semantic space. Architectural dimensions are deferred to the appendix.

For each node i, let $\mathbf{h}_i$ be the i-th row of $\mathbf{H}$, set $k_{\mathrm{eff}} = \min(k, K)$, and let $\mathcal{T}_i \subseteq \{1,\dots,K\}$ contain the indices of its k_{eff} anchors with the largest cosine similarities. ASC assigns node i only to these anchors. For $j \in \mathcal{T}_i$, let $u_{ij} = \exp(\cos(\mathbf{h}_i, \mathbf{h}_{s_j})/\tau)$ and $\eta = 10^{-4}$. The implemented assignment is

$$M_{ij} = \begin{cases} \dfrac{u_{ij}/\sum_{\ell\in\mathcal{T}_i} u_{i\ell} + \eta}{1 + k_{\mathrm{eff}}\eta}, & j \in \mathcal{T}_i, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where k controls assignment sparsity and τ controls assignment sharpness. The assignment satisfies $M_{ij} \geq 0$, $\sum_{j=1}^{K} M_{ij} = 1$, and $\mathrm{nnz}(\mathbf{M}) \leq N k_{\mathrm{eff}}$, where nnz counts stored nonzero entries. Sparse soft assignment lets a node distribute its mass among a small set of compatible anchors; $k = 1$ gives hard assignment. The mass floor η keeps selected paths active before row normalization. $\mathbf{M}$ is not an independently optimized parameter; it is recomputed from the encoder outputs at each forward pass and changes as the joint encoder is updated. Gradients propagate through the retained cosine similarities and their softmax weights; the discrete top-k indices are recomputed at each forward pass and are not differentiated through.

3.3 Weighted Attributed Skeleton Construction

Each skeleton node combines an anchor-specific semantic base with context from its assigned nodes. We first enrich the anchor with a learned residual, $\widetilde{\mathbf{X}}_s = \mathbf{X}_{\mathrm{sem}}|_{\mathcal{S}} + \mathbf{H}|_{\mathcal{S}}\mathbf{W}_a$, and aggregate assigned-node context as $\mathbf{C}_s = \mathbf{M}^\top \mathbf{X}_{\mathrm{sem}}$, where $\widetilde{\mathbf{X}}_s, \mathbf{C}_s \in \mathbb{R}^{K\times d}$ and $|_{\mathcal{S}}$ selects the anchor rows, $\mathbf{W}_a \in \mathbb{R}^{d_h\times d}$ is a learned projection. A learned sigmoid gate

$$\mathbf{\Gamma}_s = g([\widehat{\widetilde{\mathbf{X}}}_s; \widehat{\mathbf{C}}_s]), \tag{3}$$

controls how the assignment-weighted context is added to each anchor:

$$\mathbf{X}_s = \widetilde{\mathbf{X}}_s + \mathbf{\Gamma}_s \odot \widehat{\mathbf{C}}_s \odot \rho(\widetilde{\mathbf{X}}_s), \tag{4}$$

where g is a multilayer perceptron (MLP) followed by a sigmoid, and $\mathbf{\Gamma}_s \in (0,1)^{K\times d}$ is the resulting gate, $\odot$ denotes element-wise multiplication, $\widehat{\cdot}$ denotes row-wise ℓ_2 normalization, and $\rho(\widetilde{\mathbf{x}}) = \max(\|\widetilde{\mathbf{x}}\|_2, 1)$ is applied row-wise to restore the anchor-feature scale. Context normalization and scale restoration control the magnitude of the contextual update, while the residual path retains the anchor-specific representation.

ASC separates feature aggregation from edge contraction to retain multi-anchor semantic context while limiting the expansion of each original edge. Define $\bar{\mathbf{M}} = \mathrm{TopQ}(\mathbf{M}, q)$, where $q = \min(2, k_{\mathrm{eff}})$ and TopQ retains the q largest entries in each row without renormalization. An original edge (u, v) then contributes $\bar{M}_{ui} A_{uv} \bar{M}_{vj}$ to skeleton edge (i, j). The pre-pruning adjacency is

$$\mathbf{A}_s^{\mathrm{pre}} = \bar{\mathbf{M}}^\top \mathbf{A} \bar{\mathbf{M}}, \tag{5}$$

and the exported adjacency is

$$\mathbf{A}_s = \mathrm{Sym}(\mathrm{Prune}_b(\mathbf{A}_s^{\mathrm{pre}}))\,. \tag{6}$$

Here, Prune_b retains up to the b strongest outgoing weights for each skeleton node. $\mathrm{Sym}(\bullet)$ inserts reverse edges and sums duplicate entries; when both orientations of an edge survive pruning, their

weights are added in each direction. Thus each original edge produces at most q^2 weighted anchor-pair contributions and $\mathrm{nnz}(\mathbf{A}_s) \leq 2Kb$. The budgets k, q, and b separately control assignment support, edge-contraction expansion, and exported graph size.

3.4 Learning Objectives

ASC is trained with two complementary objectives. The primary objective is semantic reconstruction: after constructing the skeleton graph, we lift its features back to the original node space, $\widehat{\mathbf{X}}_{\mathrm{sem}} = \mathbf{M}\mathbf{X}_s$, and minimize

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{Nd}\|\mathbf{X}_{\mathrm{sem}} - \widehat{\mathbf{X}}_{\mathrm{sem}}\|_F^2. \tag{7}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm. This objective trains the encoder and feature fusion to reconstruct the original semantic features from the compressed representatives, tying assignment learning to semantic fidelity. During optimization, we estimate this objective on randomly sampled node batches and accumulate gradients; the complete assignment is recomputed when exporting the final skeleton graph.

The auxiliary objective uses a scale-normalized graph-level property profile $\phi(\mathbf{A}) \in \mathbb{R}^m$, where $m = 7$. A lightweight property prediction head f_θ, parameterized by θ, predicts this profile from the current skeleton representation:

$$\mathcal{L}_{\mathrm{prop}} = \frac{1}{m}\|\phi(\mathbf{A}) - f_\theta(\mathbf{X}_s, \mathbf{A}_s^{\mathrm{pre}})\|_2^2. \tag{8}$$

This term supplies an auxiliary graph-level training signal beyond node-wise semantic reconstruction. During optimization, the prediction head receives the pre-pruning adjacency $\mathbf{A}_s^{\mathrm{pre}}$; top-$b$ pruning is applied when exporting the final graph. The encoder, feature-fusion module, and property prediction head are optimized jointly. The property definitions, sampling procedure, and prediction-head architecture are provided in the appendix. The total objective is

$$\mathcal{L} = \bar{\lambda}_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \bar{\lambda}_{\mathrm{prop}}\frac{\mathcal{L}_{\mathrm{prop}}}{\mathcal{L}_{\mathrm{prop}}^{(0)}}, \tag{9}$$

where λ_{rec} and λ_{prop} are nonnegative loss weights, $\bar{\lambda}_\ell = \lambda_\ell/(\lambda_{\mathrm{rec}} + \lambda_{\mathrm{prop}})$ for $\ell \in \{\mathrm{rec}, \mathrm{prop}\}$, and $\mathcal{L}_{\mathrm{prop}}^{(0)}$ is the initial property-prediction loss. Semantic reconstruction is the primary optimization signal, while property prediction provides auxiliary graph-level supervision.

3.5 Computational Complexity

Let T_{desc} denote the one-time cost of computing structural descriptors and selecting semantic medoid anchors by spherical K-means, and let T_{enc} denote the joint-encoder cost. For one full-graph construction pass, computing node–anchor similarities costs $\mathcal{O}(NKd_h)$, sparse feature aggregation costs $\mathcal{O}(Nkd)$, and edge contraction costs $\mathcal{O}(\mathrm{nnz}(\mathbf{A})q^2)$, where $\mathrm{nnz}(\mathbf{A})$ denotes the number of nonzero entries in the adjacency matrix $\mathbf{A}$. The total construction cost is therefore

$$\mathcal{O}\big(T_{\mathrm{desc}} + T_{\mathrm{enc}} + NKd_h + Nkd + \mathrm{nnz}(\mathbf{A})q^2\big)\,. \tag{10}$$

With a node chunk size B, the corresponding memory requirement is $\mathcal{O}(N(d + d_0 + d_h) + BK + Nk + \mathrm{nnz}(\mathbf{A})q^2 + Kb)$, including the pre-pruning edge contributions. ASC never materializes a dense $N \times K$ assignment or dense $K \times K$ skeleton adjacency; assignments and contracted edges are stored sparsely.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate ASC on 11 text-attributed graph benchmarks: PubMed, OGBN-Arxiv, WikiCS, TAPE, Product-subset, CiteSeer, Cora, Photo, Computer, Actor, and Arxiv-year. We follow recent text-attributed graph benchmark collections for node text and preprocessing (He et al.,

Table 1: Overall node classification performance. For each compression method, a valid entry reports original-node test accuracy after training on its compressed graph and applying the learned GNN parameters to the original graph, shown as mean ± standard deviation (%) over three runs at an actual compression ratio within $[0.09, 0.11]$. Original denotes the $r = 1.0$ result with the same Qwen3 features and GCN evaluator. SKM-Soft denotes the training-free spherical K-means baseline with sparse soft assignment, and G-Skel. abbreviates Graph-Skeleton. The compressed method with the highest mean accuracy on each dataset is highlighted in bold. OOT denotes timeout (> 180 minutes), OOM denotes failure within the 32-GB GPU memory budget, and N/A indicates that the method cannot reach the specified ratio range under its available parameter settings.

Dataset	Original	**ASC**	SKM-Soft	UGC	AH-UGC	SGBGC	GCond	G-Skel.	FGC
PubMed	90.62±0.1	**88.36±0.5**	84.13±0.2	86.22±0.2	88.01±0.4	86.89±0.6	83.83±0.2	86.86±0.6	85.25±0.4
OGBN-Arxiv	75.40±0.3	**74.01±0.3**	69.02±0.2	56.63±0.2	65.84±0.9	OOT	OOM	37.61±5.0	OOM
WikiCS	84.98±1.0	**82.22±1.0**	78.27±0.3	61.73±1.1	76.59±0.7	62.71±0.7	71.25±0.4	47.11±2.8	57.95±0.4
TAPE	78.14±0.3	**74.52±0.6**	71.32±0.2	63.07±1.1	70.58±0.3	N/A	OOM	74.08±0.6	48.12±0.5
Product-subset	86.17±0.5	**83.85±0.8**	80.22±0.5	77.49±1.1	78.36±0.8	82.05±0.9	OOM	82.58±1.3	44.47±3.6
CiteSeer	80.51±1.0	**77.90±0.6**	74.21±0.2	75.13±1.7	76.65±0.2	N/A	70.62±0.6	77.65±0.2	69.19±0.5
Cora	89.85±1.3	**87.88±2.0**	84.25±1.2	81.06±2.6	84.19±1.1	80.26±0.3	81.77±1.2	85.50±1.4	80.50±1.1
Photo	86.00±0.2	**79.02±0.6**	74.58±0.5	60.42±1.3	73.61±2.7	78.88±0.4	OOM	40.17±3.2	61.34±0.5
Computer	89.29±0.2	**85.64±0.3**	81.42±0.3	75.95±1.3	80.10±0.5	OOT	OOM	31.92±0.3	OOM
Actor	73.49±0.5	**70.19±1.4**	66.06±0.4	65.61±0.8	69.33±1.3	67.53±0.2	67.38±1.0	50.01±1.1	55.46±10.3
Arxiv-year	48.38±0.1	**43.71±0.1**	43.59±0.1	39.48±0.1	40.77±0.3	OOT	OOM	36.30±0.9	OOM

2024; Liu et al., 2024; Wu et al., 2025; Li et al., 2025) and use OGB as the source of the Arxiv and product graphs (Hu et al., 2020). Product-subset is the OGBN-Products subset released by TAPE (He et al., 2024). Our Arxiv-year variant reuses OGBN-Arxiv and converts the official publication years into five quantile classes, following the task design of Lim et al. (2021). These datasets span different homophily levels. The largest evaluated graphs, OGBN-Arxiv and Arxiv-year, each contain 169,343 nodes; Product-subset contains 54,025 nodes. For every dataset, we generate a stratified random train/validation/test split of 60%/20%/20% using the run seed. Unless otherwise specified, experiments are repeated with three different random seeds, and we report the mean and sample standard deviation.

Text features. ASC assumes semantically informative pretrained node representations as its primary input. Unless otherwise specified, all experiments consistently use frozen Qwen3-Embedding-0.6B features (Zhang et al., 2025). The encoder is not part of ASC's optimization. Each node is represented by a 1024-dimensional embedding, providing a common feature dimension across datasets. Native dataset features are used only in feature ablations.

Baselines. We compare ASC with three training-free controls. Random and PR-Anchor select anchors uniformly or by PageRank, respectively, assign each node to its cosine-nearest anchor in the fixed input-feature space, average the assigned features, and contract the original edges. SKM-Soft runs spherical K-means on the fixed semantic features, selects one medoid per cluster, and uses sparse soft assignments with the same edge contraction and pruning as ASC. Unlike ASC, these controls do not learn a joint semantic-structural assignment space. The graph-coarsening baselines include UGC (Kataria et al., 2024), AH-UGC (Kataria et al., 2025), local-variation neighborhood coarsening (VAN) (Loukas, 2019), SGBGC (Xia et al., 2025), FGC (Kumar et al., 2023), and Graph-Skeleton (G-Skel.) (Cao et al., 2024). We also include GCond, a trainable graph-condensation method based on gradient matching (Jin et al., 2022b). All baseline comparisons use the same two-layer GCN architecture and training hyperparameters.

Evaluation protocol. We use transductive node classification. The stratified original-node split described above is created before compression and remains fixed throughout all stages. ASC can access the features and topology of the full graph but does not use node labels. ASC is evaluated at target compression ratios $r \in \{0.5, 0.1, 0.05, 0.01\}$, while $r = 1.0$ denotes the original graph. Classification accuracy is the primary metric. The default downstream model is a two-layer GCN, with GAT, GraphSAGE, and MLP used for backbone sensitivity analysis (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017). Let h_ψ denote the downstream classifier parameterized by ψ. During compressed-graph training, it produces skeleton-node logits $\mathbf{Z}_s = h_\psi(\mathbf{X}_s, \mathbf{A}_s)$, which are lifted to original-node logits as $\mathbf{Z} = \mathbf{M}\mathbf{Z}_s$. Cross-entropy is computed over the original

training nodes. For validation and testing, the learned parameters are applied to the original graph, and the resulting logits are evaluated using the fixed original validation and test masks. For efficiency, we report compression time, downstream GCN speedup, and the ratio of peak GPU memory for compressed-graph training to that for full-graph training where applicable. Runs exceeding 180 minutes are marked as OOT, and runs requiring more than 32 GB of GPU memory are marked as OOM. All baselines use the same node features and data splits. Compression time covers method-specific preprocessing through in-memory compressed-graph construction.

Implementation details. Each dataset–ratio pair is compressed by a separately trained model whose training length follows its convergence curve; all other compressor settings remain unchanged. Complete dataset statistics, feature-preparation details, compressor and GNN hyperparameters, and the ratio-matching protocol for baselines are provided in the appendix. We implement ASC in PyTorch and PyTorch Geometric and run all experiments on a server with an NVIDIA GeForce RTX 5090 GPU, 16 vCPUs from an Intel Xeon Gold 6459C CPU, and 90 GB of RAM.

4.2 Overall Performance

Table 1 compares ASC with compression and condensation baselines at $r = 0.1$, retaining only results with actual ratios in $[0.09, 0.11]$. Random, PR-Anchor, and VAN are reported in Appendix C.1.

ASC obtains the highest mean test accuracy on all 11 datasets, including against the training-free SKM-Soft baseline; its margin on Arxiv-year is 0.12 percentage points. ASC stays within 5 percentage points of the original-graph result on 10 datasets, showing consistent retention across graphs with different scales and homophily levels.

GCond, SGBGC, and FGC fail on at least one graph under the resource limits; ASC completes all 11 datasets.

ASC also outperforms AH-UGC on PHEME (Kochkina et al., 2018) and TAGFN-PF (Liu et al., 2025) graph classification, using a label-free compressor and a common GIN evaluator (Xu et al., 2019) (Appendix C.3).

4.3 Sensitivity to Compression Ratio and Downstream Backbone

Figure 2 shows accuracy declines of 2.5–4.5 percentage points from $r = 1.0$ to $r = 0.01$ on four datasets. At $r = 0.01$, ASC has higher mean accuracy than UGC, AH-UGC, and Graph-Skeleton on all six datasets in Table 2, with actual ratios in $[0.008, 0.012]$. All three baselines support this extreme ratio; all 11 datasets are reported in Appendix C.1.

We then replace the default GCN with GAT, GraphSAGE, and MLP to evaluate whether the ASC skeleton graph supports different downstream architectures. Table 3 shows that the relative performance of the tested backbones varies by dataset. GraphSAGE has the highest mean accuracy

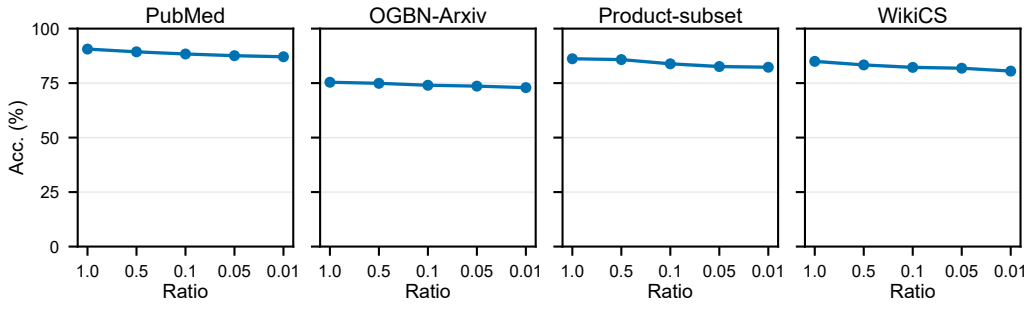


Figure 2: Node classification accuracy under different compression ratios on representative datasets. Compression ratios are treated as discrete experimental settings and shown with equal spacing on the x-axis for readability. Curves show the mean over three runs.

Table 2: Extreme compression performance at target ratio $r = 0.01$. Results are node classification accuracy (%) with actual compression ratios within $[0.008, 0.012]$, reported as mean $\pm$ standard deviation over three seeds. G-Skel. denotes Graph-Skeleton.

Method	PubMed	OGBN-Arxiv	WikiCS	Product-subset	Cora	Computer
ASC	**87.09±0.81**	**72.93±0.56**	**80.51±0.34**	**82.28±0.67**	**83.83±1.38**	**79.06±4.96**
UGC	69.07±8.60	33.05±0.84	42.39±1.64	32.23±1.68	19.78±0.22	46.93±1.19
AH-UGC	76.16±11.75	28.98±1.09	26.12±4.51	20.30±0.55	39.42±13.45	25.79±0.00
G-Skel.	86.25±0.26	42.86±5.92	47.46±1.19	80.23±1.15	76.35±0.75	34.08±0.96

Table 3: Downstream performance of ASC with different downstream backbones at $r = 0.1$. Results are node classification accuracy (%), reported as mean $\pm$ standard deviation over three runs.

Backbone	PubMed	OGBN-Arxiv	WikiCS	TAPE	Product-subset	Actor
GCN	88.36±0.5	74.01±0.3	82.22±1.0	74.52±0.6	83.85±0.8	70.19±1.4
GAT	86.83±1.1	69.10±0.2	81.99±0.6	74.93±0.3	83.66±0.6	69.46±2.1
GraphSAGE	89.98±1.5	74.14±0.0	84.11±0.9	74.28±0.6	84.46±0.3	71.42±1.7
MLP	89.56±0.7	71.16±0.3	80.42±0.5	76.71±0.2	81.59±1.2	70.63±1.0

on PubMed, OGBN-Arxiv, WikiCS, Product-subset, and Actor, whereas MLP leads on TAPE. The competitive performance of MLP further suggests that the compressed node features retain useful semantic information even without message passing. Complete results for GCN, GAT, GraphSAGE, and MLP on all 11 datasets are provided in Appendix C.2.

4.4 Efficiency

Figure 3 compares compression time. Several training-free baselines are faster than ASC but generally less accurate (Table 1). FGC and SGBGC complete several smaller datasets but incur long runtimes or OOM/OOT failures on larger graphs. ASC is not uniformly fastest, but completes all datasets within the resource limits.

Table 4 compares ASC with UGC and Graph-Skeleton under the same GCN evaluator. ASC uses only 3.5%–16.5% of full-graph peak memory while achieving $1.81\times$–$3.26\times$ speedups. ASC uses less peak memory than both alternatives on all five datasets while achieving substantially higher accuracy. Appendix C.4 additionally reports stored edge counts: top-b pruning leaves 4,680–67,736 edges, fewer than both alternatives throughout. UGC retains every nonzero inter-supernode connection in $\mathbf{M}^\top \mathbf{A}\mathbf{M}$ and can remain dense. Appendix C.4 reports ASC's three-run compression times on all datasets.

4.5 Ablation Studies

We ablate the training objectives and input signals in Table 5. The complete model attains the highest mean accuracy on all five datasets. Removing either reconstruction or graph-level property

Table 4: GCN training efficiency at $r = 0.1$, excluding validation/test inference. Spd.: speedup over full-graph training; Mem.: peak memory as a percentage of the full-graph value. G-Skel. denotes Graph-Skeleton.

Dataset	**ASC**		UGC		G-Skel.	
	Spd.	Mem.	Spd.	Mem.	Spd.	Mem.
OGBN-Arxiv	2.88×	16.5%	1.30×	88.1%	2.64×	18.3%
WikiCS	1.81×	3.5%	1.44×	27.9%	1.64×	4.7%
Photo	2.89×	7.6%	1.52×	49.5%	2.64×	8.1%
Computer	3.26×	8.9%	1.30×	50.4%	3.18×	9.3%
Arxiv-year	2.87×	15.9%	1.38×	88.0%	2.70×	17.7%

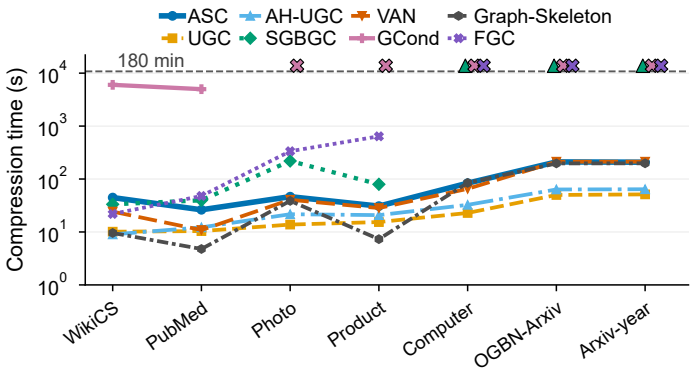


Figure 3: Compression time across datasets ordered by node count (log-scale y-axis). Dashed line: 180-minute timeout; $\times$ and $\triangle$: OOM and OOT, plotted above the threshold for visibility. G-Skel.: Graph-Skeleton.

Table 5: Ablation study of ASC components at $r = 0.1$. Entries report means over three runs. Full gives the absolute node classification accuracy (%), while the remaining columns give the mean percentage-point change relative to Full. Rec., Prop., Struct., and Sem. denote semantic reconstruction, graph-level property prediction, structural descriptors, and semantic features, respectively. Complete mean $\pm$ standard deviation results are provided in Appendices D.2–D.3.

Dataset	Full	Δ Rec.	Δ Prop.	Δ Struct.	Δ Sem.
OGBN-Arxiv	**74.01**	-3.84	-1.91	-2.02	-58.51
WikiCS	**82.22**	-21.25	-1.83	-6.58	-62.54
Photo	**79.02**	-12.59	-3.13	-0.19	-37.19
Computer	**85.64**	-18.38	-4.63	-0.98	-58.72
Actor	**70.19**	-3.75	-0.43	-1.90	-21.32

prediction consistently reduces accuracy, showing that both terms improve downstream performance in this comparison.

Removing semantic features causes the largest drop on every dataset, reflecting their roles in both assignment learning and skeleton feature construction. Removing structural descriptors has a smaller, dataset-dependent effect. Appendices D.2–D.4 further examine individual graph-level properties, input feature sources, assignment sparsity and temperature, and skeleton-edge budgets. The input ablation highlights the role of semantic features, while the assignment-budget results support sparse multi-anchor assignment; increasing the skeleton-edge budget beyond the default does not improve mean accuracy on the four datasets reported in the appendix.

5 Conclusion

We presented ASC, a semantic-first compression framework for text-attributed graphs. It learns sparse node-to-anchor assignments from pretrained semantic features and complementary structural descriptors, then constructs a reusable weighted attributed skeleton graph. ASC achieves the highest mean accuracy on all 11 datasets at $r = 0.1$, has higher mean accuracy than UGC, AH-UGC, and Graph-Skeleton on all benchmarks at $r = 0.01$, and reduces downstream GCN memory and time. Chunking reduces peak similarity memory but not the $\mathcal{O}(NKd_h)$ node-to-anchor computation or the need for per-graph optimization. Future work will explore approximate assignment search and reusable compressors.

AI USE STATEMENT

We used generative AI tools to assist with translation and language polishing of the manuscript, writing portions of dataset-loading code, scripts for conducting ablation and batch experiments, scripts for plotting and visualizing experimental results, and checking the formatting and placement of citations and references. We have not used generative AI tools to design the core research methodology or experimental design, to implement the core method or model architecture, or to generate, modify, or fabricate experimental data or results, and the rest of the required disclosure tasks are not applicable to this work. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main text specifies the compression method and evaluation protocol. The appendix documents dataset preparation, hyperparameters, and additional experimental details. An anonymized code package provides the ASC implementation, experiment scripts, and instructions for obtaining datasets and preparing text embeddings.

REFERENCES

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi: 10.1088/1742-5468/2008/10/P10008. URL https://doi.org/10.1088/1742-5468/2008/10/P10008.

Linfeng Cao, Haoran Deng, Yang Yang, Chunping Wang, and Lei Chen. Graph-skeleton: ˜1% nodes are sufficient to represent billion-scale graph. In *Proceedings of the ACM Web Conference 2024*, pp. 570–581, 2024. doi: 10.1145/3589334.3645452. URL https://dl.acm.org/doi/10.1145/3589334.3645452.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hESD2NJFg8.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 257–266, 2019. doi: 10.1145/3292500.3330925. URL https://dl.acm.org/doi/10.1145/3292500.3330925.

Jeongwhan Choi, Seungjun Park, Sumin Park, Sung-Bae Cho, and Noseong Park. Are graph transformers necessary? efficient long-range message passing with fractal nodes in MPNNs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(25):20553–20561, 2026. doi: 10.1609/aaai.v40i25.39191. URL https://ojs.aaai.org/index.php/AAAI/article/view/39191.

Ido Cohen and Ronen Talmon. Spectral graph coarsening using inner product preservation and the grassmann manifold. In *Advances in Neural Information Processing Systems*, volume 38, pp. 117415–117446, 2025. doi: 10.52202/085713-3915. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/aa4bee5e2720d7e8f073d762ee8c7fd4-Paper-Conference.pdf.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007. doi: 10.1109/TPAMI.2007.1115. URL https://doi.org/10.1109/TPAMI.2007.1115.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International*

Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=wTTjnvGphYj.

Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. GAugLLM: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 747–758, 2024. doi: 10.1145/3637528.3672035. URL https://doi.org/10.1145/3637528.3672035.

Jiarui Feng, Hao Liu, Lecheng Kong, Mingfang Zhu, Yixin Chen, and Muhan Zhang. TAGLAS: An atlas of text-attributed graph datasets in the era of large graph and language models. *arXiv preprint arXiv:2406.14683*, 2024. URL https://arxiv.org/abs/2406.14683.

Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. Graph condensation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1819–1837, 2025. doi: 10.1109/TKDE.2025.3535877.

Mridul Gupta, Samyak Jain, Vansh Ramani, Hariprasad Kodamana, and Sayan Ranu. Bonsai: Gradient-free graph condensation for node classification. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5x88lQ2MsH.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.

Xiaoxue Han, Zhuo Feng, and Yue Ning. A topology-aware graph coarsening framework for continual graph learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 132491–132523, 2024. doi: 10.52202/079017-4212. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ef283d62b4bce30854a8d4827f331229-Paper-Conference.pdf.

Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B. Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8058–8066, 2024. doi: 10.24963/ijcai.2024/891. URL https://www.ijcai.org/proceedings/2024/891.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RXFVcynVe1.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. Patton: Language model pretraining on text-rich networks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 7005–7020, 2023. doi: 10.18653/v1/2023.acl-long.387. URL https://aclanthology.org/2023.acl-long.387/.

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022a. doi: 10.1145/3534678.3539429. URL https://doi.org/10.1145/3534678.3539429.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=WLEx3Jo4QaB.

Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4452–4462. PMLR, 2020. URL https://proceedings.mlr.press/v108/jin20a.html.

Antonin Joly and Nicolas Keriven. Graph coarsening with message-passing guarantees. In *Advances in Neural Information Processing Systems*, volume 37, pp. 114902–114927, 2024. doi: 10.52202/079017-3648. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d041d6bb47c01a4ce327a1773703e9a0-Paper-Conference.pdf.

Mohit Kataria, Sandeep Kumar, and Jayadeva. UGC: Universal graph coarsening. In *Advances in Neural Information Processing Systems*, volume 37, pp. 63057–63081, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/733209a1f12071a7ec979e8ffaeb1d99-Abstract-Conference.html.

Mohit Kataria, Shreyash Bhilwade, Sandeep Kumar, and Jayadeva. AH-UGC: Adaptive and heterogeneous-universal graph coarsening. *arXiv preprint arXiv:2505.15842*, 2025. URL https://arxiv.org/abs/2505.15842.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3402–3413, Santa Fe, New Mexico, USA, 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1288/.

Manoj Kumar, Anurag Sharma, Shashwat Saxena, and Sandeep Kumar. Featured graph coarsening with similarity guarantees. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17953–17975. PMLR, 2023. URL https://proceedings.mlr.press/v202/kumar23a.html.

Shujie Li, Yuxia Wu, Chuan Shi, and Yuan Fang. HeTGB: A comprehensive benchmark for heterophilic text-attributed graphs. *arXiv preprint arXiv:2503.04822*, 2025. URL https://arxiv.org/abs/2503.04822.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ae816a80e4c1c56caa2eb4e1819cbb2f-Abstract.html.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4IT2pgc9v6.

Kay Liu, Yuwei Han, Haoyan Xu, Henry Peng Zou, Yue Zhao, and Philip S. Yu. TAGFN: A text-attributed graph dataset for fake news detection in the age of LLMs. *arXiv preprint arXiv:2511.21624*, 2025. doi: 10.48550/arXiv.2511.21624. URL https://arxiv.org/abs/2511.21624.

Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697*, 2022. URL https://arxiv.org/abs/2206.13697.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019. URL https://jmlr.org/papers/v20/18-680.html.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999. URL https://ilpubs.stanford.edu/422/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes F. Lutzeyer. Understanding virtual nodes: Oversquashing and node heterogeneity. In *International Conference on Learning Representations*, pp. 27512–27540, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/448abd486677165ceedfa790e9a61802-Paper-Conference.pdf.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Carlos Vonessen, Florian Grötschla, and Roger Wattenhofer. Next level message-passing with hierarchical support graphs. *arXiv preprint arXiv:2406.15852*, 2024. doi: 10.48550/arXiv.2406.15852. URL https://arxiv.org/abs/2406.15852.

Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. When do LLMs help with node classification? a comprehensive analysis. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67623–67649. PMLR, 2025. URL https://proceedings.mlr.press/v267/wu25y.html.

Shuyin Xia, Xinjun Ma, Zhiyuan Liu, Cheng Liu, Sen Zhao, and Guoyin Wang. Graph coarsening via supervised granular-ball for scalable graph neural network training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12):12872–12880, 2025. doi: 10.1609/aaai.v39i12.33404. URL https://ojs.aaai.org/index.php/AAAI/article/view/33404.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: GNN-nested transformers for representation learning on textual graph. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f18a6d1cde4b205199de8729a6637b42-Abstract.html.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, volume 31, pp. 4800–4810, 2018.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=q0nmYciuuZN.

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/13183a224208671a6fc33ba1aa661ec4-Abstract-Conference.html.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. ENGINE: Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 5734–5742, 2024. URL https://www.ijcai.org/proceedings/2024/634.

Aaron Zolnai-Lucas, Jack Boylan, Chris Hokamp, and Parsa Ghaffari. STAGE: Simplified text-attributed graph embeddings using pre-trained LLMs. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models*, pp. 92–104. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.kallm-1.10. URL https://aclanthology.org/2024.kallm-1.10/.

A Method and Implementation Details

This section supplements the method description with the structural input, the sparse construction procedure, and graph-level property prediction.

A.1 Structural Descriptors and Joint Encoder

As in the main paper, let $G = (\mathcal{V}, \mathbf{A}, \{t_i\}_{i=1}^N)$ be the input graph, where $N = |\mathcal{V}|$ and t_i is the text of node i. Its semantic features form $\mathbf{X}_{\text{sem}} \in \mathbb{R}^{N\times d}$. ASC additionally constructs a structural descriptor matrix $\mathbf{X}_{\text{str}} \in \mathbb{R}^{N\times d_0}$ from the undirected graph. In our implementation, $d = 1024$ and $d_0 = 24$. The first eight structural channels are the nontrivial eigenvectors associated with the smallest eigenvalues of the normalized Laplacian (Dwivedi et al., 2022). Seven role-oriented channels contain degree, log-degree, local clustering coefficient, PageRank (Page et al., 1999), the number of nodes reachable within two hops, and two sampled truncated-distance features with maximum depths of two and three. The remaining nine channels consist of an eight-dimensional, seed-fixed embedding of the node's Louvain community (Blondel et al., 2008) and the fraction of its neighbors that belong to the same community. Each channel is standardized independently within a graph. The sampled distance features use 256 and 128 sources, respectively, which avoids all-pairs shortest-path computation on large graphs.

Let $\mathbf{x}_{\text{sem},i}$ and $\mathbf{x}_{\text{str},i}$ denote the i-th rows of $\mathbf{X}_{\text{sem}}$ and $\mathbf{X}_{\text{str}}$, respectively. ASC first applies LayerNorm to the semantic feature and lifts the structural descriptor to 128 dimensions with a two-layer MLP. Their concatenation is mapped to a 512-dimensional joint representation:

$$\mathbf{h}_i = \text{Enc}([\text{LN}(\mathbf{x}_{\text{sem},i}) \| \, \text{Lift}(\mathbf{x}_{\text{str},i})]) \, . \tag{11}$$

Both MLPs use LayerNorm and GELU between their linear layers. The resulting joint representation is used to compute node-to-anchor compatibility.

A.2 Sparse Assignment and Skeleton Construction

Given a target ratio r, ASC sets $K = \max(1, \lfloor rN \rfloor)$ and runs spherical K-means on $\mathbf{X}_{\text{sem}}$. From each cluster, it selects the actual node closest in cosine similarity to the cluster center, yielding anchors $\mathcal{S} = \{s_1, \ldots, s_K\}$. The default uses 20 iterations and one initialization. PageRank, random, and degree-based alternatives are evaluated in Table 14. The joint node and anchor representations are ℓ_2-normalized before cosine similarities are computed; specifically, $\bar{\mathbf{h}}_i = \mathbf{h}_i / \max(\|\mathbf{h}_i\|_2, \epsilon)$, where $\epsilon > 0$ prevents division by zero. Each node retains its $k_{\text{eff}} = \min(k, K)$ most similar anchors ($k = 3$), whose indices form $\mathcal{T}_i$, and the retained scores are normalized with a temperature-τ softmax ($\tau = 0.5$). Writing $u_{ij} = \exp(\langle \bar{\mathbf{h}}_i, \bar{\mathbf{h}}_{s_j} \rangle / \tau)$ for $j \in \mathcal{T}_i$, the mass-floored assignment is

$$M_{ij} = \begin{cases} \dfrac{u_{ij} / \sum_{\ell \in \mathcal{T}_i} u_{i\ell} + \eta}{1 + k_{\text{eff}}\eta}, & j \in \mathcal{T}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Gradients propagate through the retained similarities and softmax weights, whereas the discrete top-k indices are recomputed at each forward pass and are not differentiated through. Thus, every row of $\mathbf{M} \in \mathbb{R}^{N\times K}$ sums to one and has at most three nonzero entries, with $\eta = 10^{-4}$ as in the main text.

The semantic context assigned to each anchor is $\mathbf{C}_s = \mathbf{M}^\top \mathbf{X}_{\text{sem}}$. A learned gate combines the anchor's semantic feature and joint-encoder residual with the normalized context, producing the skeleton feature matrix $\mathbf{X}_s$. Feature aggregation uses the full sparse assignment $\mathbf{M}$. For edge contraction, the implementation forms $\bar{\mathbf{M}} = \mathrm{TopQ}(\mathbf{M}, q)$ with $q = \min(2, k_{\text{eff}})$ by retaining the q strongest endpoint assignments and setting the remainder to zero without renormalization. The unpruned skeleton adjacency is therefore $\mathbf{M}^\top \mathbf{A} \bar{\mathbf{M}}$ rather than the full $\mathbf{M}^\top \mathbf{A} \mathbf{M}$. It is constructed directly as a sparse COO tensor, and each original edge produces at most four weighted anchor-pair contributions. Contributions are processed in edge chunks and coalesced, so the dense $N \times K$ assignment and dense $K \times K$ adjacency are never materialized. Finally, ASC applies Prune_b with $b = 4$ to retain the four largest outgoing weights in each row, and then applies Sym to add reverse edges and coalesce duplicates. Thus, b is a row-wise pruning budget rather than a strict bound on the final undirected degree. The output is a sparse, weighted attributed skeleton graph $G_s = (\mathcal{V}_s, \mathbf{A}_s, \mathbf{X}_s)$ with $|\mathcal{V}_s| = K$, which can be consumed by downstream models. Stored skeleton-edge counts use the COO representation, count the two directions separately, and exclude self-loops added later by the downstream GNN.

Node–anchor similarities are also computed in chunks. The initial chunk size is selected from available GPU memory and is halved automatically after a CUDA out-of-memory exception. Sparse matrix multiplications are performed in float32 outside automatic mixed precision to avoid unsupported half-precision sparse kernels.

A.3 Graph-Level Property Prediction

For the input graph defined above, ASC uses the seven-dimensional profile $\phi(\mathbf{A})$ of scale-adjusted graph properties as the regression target for an auxiliary graph-level objective: the Fiedler value and spectral radius of the normalized Laplacian, average clustering coefficient, Wiener-index proxy, diameter proxy, maximum-to-mean degree ratio, and degree assortativity. Before regression, the two spectral statistics and Wiener proxy use $\log(1 + x)$, the degree ratio uses a logarithm, and the diameter is divided by $\log(1 + N)$. Clustering is sampled from at most 2,048 eligible nodes with at most 24 neighbors per node. The distance-based statistics use breadth-first searches from 24 sampled non-isolated nodes. These approximations make the target computation feasible.

Directly differentiating through these graph algorithms is impractical. ASC therefore computes the fixed target $\phi(\mathbf{A})$ once from the complete input graph. With the default update interval of one epoch, the property branch independently samples 60% of $\mathcal{V}$ at every epoch and constructs the corresponding induced subgraph and skeleton representation. The property prediction head uses the pre-pruning skeleton adjacency and receives mean- and max-pooled skeleton features together with six compact graph summaries: log node count, density, log mean degree, log degree standard deviation, log maximum degree, and log edge-weight standard deviation. A three-layer MLP implements $f_\theta(\mathbf{X}_s, \mathbf{A}_s^{\text{pre}}) \in \mathbb{R}^7$, and the auxiliary loss is

$$\mathcal{L}_{\text{prop}} = \frac{1}{7} \left\| f_\theta(\mathbf{X}_s, \mathbf{A}_s^{\text{pre}}) - \phi(\mathbf{A}) \right\|_2^2 . \tag{13}$$

The sampled skeleton changes across epochs, whereas the target is not recomputed from the sampled subgraph. The loss is normalized by $\mathcal{L}_{\text{prop}}^{(0)}$, its value at the first update; the implementation clamps this denominator below at 10^{-3} for numerical stability.

ASC combines this auxiliary branch with its primary semantic reconstruction objective. The latter lifts the skeleton features back to the original node space as $\widehat{\mathbf{X}}_{\text{sem}} = \mathbf{M}\mathbf{X}_s$ and is defined as

$$\mathcal{L}_{\text{rec}} = \frac{1}{Nd} \left\| \mathbf{X}_{\text{sem}} - \widehat{\mathbf{X}}_{\text{sem}} \right\|_F^2 . \tag{14}$$

The total objective is $\bar{\lambda}_{\text{rec}} \mathcal{L}_{\text{rec}} + \bar{\lambda}_{\text{prop}} \mathcal{L}_{\text{prop}} / \mathcal{L}_{\text{prop}}^{(0)}$, where λ_{rec} and λ_{prop} are the unnormalized reconstruction and property-prediction loss weights, respectively, and $\bar{\lambda}_\ell = \lambda_\ell / (\lambda_{\text{rec}} + \lambda_{\text{prop}})$ for $\ell \in \{\text{rec}, \text{prop}\}$. The property prediction head and compressor are optimized jointly, and the resulting objective supplies an auxiliary graph-level training signal.

Algorithm 1 ASC compression and skeleton export

Require: Graph structure $(\mathcal{V}, \mathbf{A})$, semantic features $\mathbf{X}_{\text{sem}}$, compression ratio r, epoch budget T
Ensure: Sparse assignment $\mathbf{M}$ and skeleton graph $G_s = (\mathcal{V}_s, \mathbf{A}_s, \mathbf{X}_s)$

```
Compute structural descriptors X_str and the graph-property target
Set K = max(1, ⌊r|V|⌋); run spherical K-means on X_sem and select one medoid anchor per cluster
for t = 1, ..., T do
    Encode (X_sem, X_str) and recompute the sparse top-k assignment M
    Construct current skeleton features X_s and pre-pruning adjacency A_s^pre
    Estimate L_rec on sampled node batches
    Sample an induced subgraph and evaluate L_prop
    Update the encoder, feature-fusion module, and property head using λ̄_rec L_rec + λ̄_prop L_prop / L_prop^(0)
end for
Recompute the complete M and X_s with the selected model state
Contract edges with M̄ = TopQ(M, q) and form A_s^pre = M̄^⊤ A M̄
Export A_s = Sym(Prune_b(A_s^pre)), X_s, and M
```

Table 6: Dataset statistics. Edge counts treat an undirected pair as one edge.

Dataset	Nodes	Edges	Classes
PubMed	19,717	44,324	3
OGBN-Arxiv	169,343	1,157,799	40
WikiCS	11,701	215,603	10
TAPE	57,471	60,974	38
Product-subset	54,025	72,319	42
CiteSeer	3,186	4,225	6
Cora	2,708	5,278	7
Photo	48,362	436,891	12
Computer	87,229	628,274	10
Actor	4,416	12,120	5
Arxiv-year	169,343	1,157,799	5

B Experimental Protocol

This section documents the datasets, feature preparation, and the training and evaluation settings used in the reported experiments.

B.1 Datasets and Feature Preparation

Table 6 summarizes the 11 benchmarks. We use the text and graph preprocessing from recent text-attributed graph benchmark collections (He et al., 2024; Liu et al., 2024; Wu et al., 2025; Li et al., 2025). Product-subset is the OGBN-Products subset released by TAPE (He et al., 2024). Arxiv-year reuses the OGBN-Arxiv graph and reads publication years from OGB's official `node_year.csv`. The 20th, 40th, 60th, and 80th percentiles define five year classes following the task design of Lim et al. (2021). For every dataset, we construct a stratified random train/validation/test split of 60%/20%/20% using the run seed.

Unless stated otherwise, node text is encoded once with the frozen Qwen3-Embedding-0.6B model (Zhang et al., 2025) through the Sentence-Transformers interface (Reimers & Gurevych, 2019). We concatenate the available title and body or abstract fields before encoding, store the resulting vectors as float32 arrays, and reuse them in all methods. The resulting 1,024-dimensional feature space gives every dataset a common semantic input dimension; the text encoder is not fine-tuned during compression or downstream training. The code option named `bow` denotes each dataset's native feature matrix and does not imply that every native matrix is a bag-of-words representation. Native features are used only in the feature-quality analysis.

B.2 Training, Evaluation, and Hyperparameters

ASC is trained separately for each dataset and compression ratio, without class labels. Compression uses Adam with learning rate 8×10^{-3}, weight decay 2×10^{-4}, and a StepLR schedule that multiplies the learning rate by 0.5 every 60 epochs. The shared configuration uses a 512-dimensional joint representation, at most three nonzero assignments per node, at most two endpoint assignments per node during edge contraction, temperature 0.5, and a row-wise pre-symmetrization pruning budget of four skeleton edges. The reconstruction and property-prediction coefficients are 1,500 and 50 before normalization. Automatic mixed precision is enabled for dense operations. The reconstruction branch uses a sampling budget of 16,384 node instances per epoch (`--main-target-total 16384`), distributed across gradient-accumulation steps according to the available memory. The model state with the lowest total compression loss within the fixed epoch budget is used to produce the final artifact.

The compressor epoch budgets are selected from the convergence study and then held fixed across the three runs with different random seeds. They are 320 for PubMed and WikiCS; 240 for CiteSeer and Computer; 200 for Product-subset, OGBN-Arxiv, and Cora; 120 for Photo and Actor; and 80 for TAPE and Arxiv-year. Other compressor hyperparameters are shared across datasets.

For downstream evaluation, we use a two-layer GCN with 256 hidden units, ReLU, dropout 0.5, and label smoothing 0.05. It is optimized with Adam for at most 300 epochs using learning rate 6×10^{-3}, weight decay 5×10^{-7}, and a StepLR factor of 0.8 every 50 epochs. Training stops after 60 epochs without a validation-accuracy improvement greater than 10^{-4}. During training, the GCN produces skeleton logits $\mathbf{Z}_s$ and ASC lifts them to the original node space as $\mathbf{Z} = \mathbf{M}\mathbf{Z}_s$; cross-entropy is computed on the original training mask. For validation and testing, the GCN parameters learned from the compressed graph are applied directly to the original feature matrix and adjacency, and metrics are computed with the original validation and test masks. We report mean accuracy and sample standard deviation over the three seeds. The efficiency experiment is different: it measures the downstream training stage on the compressed graph and excludes validation and test evaluation.

The downstream-backbone experiment retains the same optimizer, hidden width, dropout, scheduler, stopping rule, and two-layer depth. GAT uses four attention heads with averaged head outputs (`concat=False`); GraphSAGE uses mean aggregation; and MLP uses two linear layers with ReLU and dropout between them. GCN and GAT consume the learned skeleton-edge weights, whereas GraphSAGE and MLP use the skeleton connectivity and node features, respectively.

For compression runtime comparisons, timing begins after the graph and precomputed node features have been loaded. It includes method-specific preprocessing (including ASC's structural descriptors and graph-property target), all compression optimization epochs, and in-memory compressed-graph construction, but excludes offline text encoding, input loading, artifact serialization, hyperparameter search, and downstream evaluation.

All experiments run on Ubuntu 22.04 with Python 3.12.3, PyTorch 2.8.0 with CUDA 12.8, PyTorch Geometric 2.7.0, NumPy 1.26.4, SciPy 1.12.0, and NetworkX 3.5. The hardware consists of one NVIDIA GeForce RTX 5090 GPU, 16 Intel Xeon Gold 6459C vCPUs, and 90 GB of RAM.

B.3 Ratio Matching

For methods that cannot directly control the retained-node ratio, we search their primary ratio-controlling hyperparameter and select the configuration whose achieved ratio is closest to the target. A result is accepted for the $r = 0.1$ comparison only when its actual ratio lies in $[0.09, 0.11]$; the corresponding interval for the $r = 0.01$ comparison is $[0.008, 0.012]$. Otherwise, the result is reported as N/A.

C Complete Experimental Results

This section provides the complete three-run results for compression-ratio, extreme-compression, and downstream-backbone experiments across all 11 datasets.

Table 7: Complete compression-ratio results for ASC. Entries are mean node classification accuracy (%) ± sample standard deviation over three runs. Original denotes the uncompressed graph with the same semantic features and evaluator.

Dataset	Original	$r = 0.5$	$r = 0.1$	$r = 0.05$	$r = 0.01$
PubMed	90.62 ± 0.13	89.35 ± 0.45	88.36 ± 0.51	87.58 ± 0.74	87.09 ± 0.81
OGBN-Arxiv	75.40 ± 0.25	74.90 ± 0.21	74.01 ± 0.32	73.61 ± 0.30	72.93 ± 0.56
WikiCS	84.98 ± 0.97	83.34 ± 0.52	82.22 ± 0.99	81.83 ± 0.44	80.51 ± 0.34
TAPE	78.14 ± 0.25	77.38 ± 0.30	74.52 ± 0.63	73.96 ± 0.32	72.48 ± 1.00
Product-subset	86.17 ± 0.53	85.79 ± 0.33	83.85 ± 0.79	82.61 ± 1.05	82.28 ± 0.67
CiteSeer	80.51 ± 1.01	78.42 ± 0.92	77.90 ± 0.57	76.33 ± 0.56	76.02 ± 1.91
Cora	89.85 ± 1.29	88.44 ± 0.77	87.88 ± 2.03	85.55 ± 3.00	83.83 ± 1.38
Photo	86.00 ± 0.15	82.94 ± 1.73	79.02 ± 0.64	77.01 ± 0.35	78.26 ± 1.09
Computer	89.29 ± 0.15	87.62 ± 0.23	85.64 ± 0.28	84.09 ± 0.37	79.06 ± 4.96
Actor	73.49 ± 0.46	71.54 ± 1.73	70.19 ± 1.39	68.39 ± 0.88	64.96 ± 7.02
Arxiv-year	48.38 ± 0.14	45.56 ± 0.37	43.71 ± 0.08	41.15 ± 0.15	41.11 ± 0.07

Table 8: Additional baseline accuracy at target ratio $r = 0.1$. Entries are mean node classification accuracy (%) ± standard deviation over three runs, with actual compression ratios in $[0.09, 0.11]$. PR-Anchor is the nontrainable PageRank-anchor baseline.

Dataset	Random	PR-Anchor	VAN
PubMed	77.03 ± 1.1	78.24 ± 0.8	86.01 ± 0.5
OGBN-Arxiv	39.43 ± 0.4	40.77 ± 0.4	31.38 ± 2.8
WikiCS	58.21 ± 1.9	65.90 ± 0.4	42.13 ± 15.2
TAPE	31.68 ± 3.8	38.47 ± 2.7	50.02 ± 4.0
Product-subset	59.45 ± 0.3	61.05 ± 0.2	59.08 ± 5.4
CiteSeer	70.22 ± 0.4	69.45 ± 0.1	62.38 ± 3.1
Cora	59.08 ± 8.1	61.80 ± 4.7	71.53 ± 1.5
Photo	76.48 ± 0.2	74.80 ± 0.3	52.38 ± 7.6
Computer	79.41 ± 0.3	79.39 ± 0.3	59.16 ± 0.7
Actor	48.70 ± 0.0	48.70 ± 0.0	56.90 ± 4.5
Arxiv-year	38.40 ± 0.3	38.03 ± 0.1	32.08 ± 4.7

C.1 Complete Accuracy and Compression-Ratio Results

Table 7 extends the compression-ratio analysis to all 11 datasets. Each entry is the mean accuracy over three runs with different random seeds. Mean accuracy generally decreases as the graph is compressed, although the trend is not strictly monotonic on every dataset. The magnitude of the reduction is dataset dependent: from $r = 0.1$ to $r = 0.01$, OGBN-Arxiv decreases by 1.08 percentage points, whereas TAPE and Cora decrease by 2.04 and 4.05 points, respectively.

For completeness, Table 8 reports the training-free Random and PR-Anchor baselines and VAN, omitted from the main comparison table for legibility.

Table 9 gives the complete three-seed comparison at $r = 0.01$, with all achieved ratios lying within $[0.008, 0.012]$. ASC has higher mean accuracy than all three baselines on all 11 datasets; the margin over Graph-Skeleton is only 0.44 percentage points on CiteSeer. UGC and AH-UGC show substantial seed sensitivity on PubMed and CiteSeer, and AH-UGC also varies considerably on Cora.

C.2 Complete Downstream Backbone Results

Table 10 reports the three-run mean and standard deviation for GCN, GAT, GraphSAGE, and MLP on all datasets. GraphSAGE has the highest mean accuracy on PubMed, OGBN-Arxiv, WikiCS, Product-subset, Photo, and Actor; MLP leads on TAPE; and GCN leads on CiteSeer, Cora, Computer, and Arxiv-year. These results show that the highest-mean backbone varies by dataset.

Table 9: Complete extreme-compression results at target ratio $r = 0.01$ over three seeds. Accuracy is reported in percent as mean $\pm$ standard deviation. All achieved ratios lie within $[0.008, 0.012]$. G-Skel. denotes Graph-Skeleton.

Dataset	ASC	UGC	AH-UGC	G-Skel.
PubMed	**87.09±0.81**	69.07±8.60	76.16±11.75	86.25±0.26
OGBN-Arxiv	**72.93±0.56**	33.05±0.84	28.98±1.09	42.86±5.92
WikiCS	**80.51±0.34**	42.39±1.64	26.12±4.51	47.46±1.19
TAPE	**72.48±1.00**	37.82±1.41	37.78±2.64	62.78±2.28
Product-subset	**82.28±0.67**	32.23±1.68	20.30±0.55	80.23±1.15
CiteSeer	**76.02±1.91**	37.24±8.48	40.82±21.96	75.58±1.94
Cora	**83.83±1.38**	19.78±0.22	39.42±13.45	76.35±0.75
Photo	**78.26±1.09**	42.89±0.23	41.54±0.00	33.04±0.74
Computer	**79.06±4.96**	46.93±1.19	25.79±0.00	34.08±0.96
Actor	**64.96±7.02**	48.71±0.00	48.71±0.00	55.49±2.20
Arxiv-year	**41.11±0.07**	35.60±0.85	37.59±0.52	38.98±1.08

Table 10: Complete downstream-backbone results at $r = 0.1$. Entries are node classification accuracy (%), reported as mean $\pm$ standard deviation over three runs.

Dataset	GCN	GAT	GraphSAGE	MLP
PubMed	88.36±0.51	86.83±1.11	89.98±1.54	89.56±0.74
OGBN-Arxiv	74.01±0.32	69.10±0.16	74.14±0.04	71.16±0.33
WikiCS	82.22±0.99	81.99±0.58	84.11±0.86	80.42±0.54
TAPE	74.52±0.63	74.93±0.26	74.28±0.57	76.71±0.19
Product-subset	83.85±0.79	83.66±0.61	84.46±0.31	81.59±1.17
CiteSeer	77.90±0.57	76.64±0.72	77.64±0.64	74.97±1.49
Cora	87.88±2.03	87.27±1.12	87.45±1.33	83.33±0.10
Photo	79.02±0.64	81.31±0.28	82.74±0.29	73.25±0.03
Computer	85.64±0.28	85.55±0.83	80.17±6.83	70.49±0.08
Actor	70.19±1.39	69.46±2.12	71.42±1.70	70.63±0.99
Arxiv-year	43.71±0.08	41.60±0.19	42.71±0.05	41.51±0.23

C.3 Graph Classification

We additionally evaluate whether ASC can compress collections of graphs rather than a single node-classification graph. For PHEME (Kochkina et al., 2018), we use a stratified $60/20/20$ graph split; for TAGFN-PolitiFact (TAGFN-PF) (Liu et al., 2025), we use its provided 62/31/221 train/validation/test split. For each dataset, we jointly optimize a single set of ASC parameters over the training graphs without using graph labels. For each graph, we compute structural descriptors, select semantic medoid anchors by spherical K-means, and construct its assignment matrix and skeleton graph. The joint encoder, feature-fusion module, and property-prediction head are shared across graphs. Graph-wise compression losses are averaged within each graph batch to update these shared parameters. After training, the compressor is frozen and applied independently to the training, validation, and test graphs; neither graph labels nor validation/test graphs are used to update it. Original denotes the same two-layer GIN trained on the uncompressed graphs (Xu et al., 2019), while AH-UGC independently coarsens each graph. All three conditions use the same downstream GIN evaluator. The shared ASC compressor is trained for 320 epochs on PHEME and 480 epochs on TAGFN-PF. Each compressor epoch samples at most 256 training graphs and processes them in batches of 16; the remaining ASC hyperparameters follow Appendix B.2. The downstream GIN uses the same two-layer configuration and training hyperparameters as the other downstream GNNs: 256 hidden units, dropout 0.5, Adam with learning rate 6×10^{-3} and weight decay 5×10^{-7}, a StepLR factor of 0.8 every 50 epochs, label smoothing 0.05, and at most 300 epochs with patience 60. Graph batches contain 64 graphs, and global mean pooling followed by a linear classifier produces graph-level predictions.

Table 11: Graph-classification accuracy (%) at $r = 0.1$, reported as mean $\pm$ standard deviation over three runs. Original uses uncompressed graphs; the compressed method with the highest mean accuracy is shown in bold.

Dataset	Original GIN	ASC	AH-UGC
PHEME	84.07 ± 0.57	$\mathbf{85.35 \pm 1.30}$	78.81 ± 1.93
TAGFN-PF	80.54 ± 2.72	$\mathbf{74.36 \pm 1.05}$	70.74 ± 4.06

Table 12: ASC compression time at $r \approx 0.1$, reported as mean $\pm$ sample standard deviation over three runs.

Dataset	Epochs	Time (s)
PubMed	320	26.24 ± 0.14
OGBN-Arxiv	200	210.53 ± 1.71
WikiCS	320	44.65 ± 0.59
TAPE	80	23.47 ± 0.07
Product-subset	200	30.79 ± 0.91
CiteSeer	240	35.94 ± 1.15
Cora	200	31.28 ± 1.42
Photo	120	46.40 ± 0.50
Computer	240	83.45 ± 1.02
Actor	120	15.68 ± 0.50
Arxiv-year	80	207.88 ± 1.36

Table 11 shows that ASC achieves slightly higher mean accuracy than the uncompressed-graph result on PHEME and exceeds AH-UGC by 6.54 percentage points. On TAGFN-PF, compression reduces accuracy relative to the original graphs, but ASC remains 3.62 percentage points above AH-UGC.

C.4 Detailed Efficiency and Failure Results

Tables 12 and 13 report ASC's compression time under the dataset-specific epoch budgets and the absolute downstream training costs and stored edge counts on representative graphs. The compression-timing scope follows Appendix B.2. All downstream measurements use the same two-layer GCN and a fixed 300-epoch budget with early stopping disabled. Time per epoch is the total training time divided by 300, and memory is the peak CUDA memory allocated during training. Compression, data loading, and validation/test evaluation are excluded from both measurements. The timed training step still lifts skeleton logits through $\mathbf{M}$ to compute the loss on the original training mask. Each undirected edge is stored in both directions in COO format, and self-loops are excluded from the edge count.

ASC has the lowest per-epoch time on OGBN-Arxiv, WikiCS, and Photo, while Graph-Skeleton is fastest on Computer and Arxiv-year. ASC consistently uses less peak memory than both alternatives and reduces the gap to Graph-Skeleton to between 20 and 113 MB, while requiring substantially less memory than UGC. These results show that ASC's main downstream efficiency advantage is its compact memory footprint; it does not uniformly provide the lowest per-epoch runtime.

Across the 11 datasets, ASC requires 15.68–210.53 seconds for compression under the dataset-specific epoch budgets. These values represent the complete compression stage defined in Appendix B.2, rather than per-epoch optimization time. Consequently, they reflect both graph-dependent preprocessing costs and the number of compression epochs used for each dataset.

Table 13 further shows that ASC stores $33.1\times$–$65.5\times$ fewer COO edges than UGC and reduces its peak memory by 81.3%–87.5% on the five evaluated graphs. Compared with Graph-Skeleton, ASC stores $1.3\times$–$4.1\times$ fewer edges while using slightly less memory. Thus, ASC's downstream efficiency is associated primarily with the compactness of the exported skeleton graph rather than uniformly faster per-epoch computation.

Table 13: Downstream GCN training cost and stored graph size at $r \approx 0.1$. The suffixes K and M indicate thousands and millions; reverse edge directions are counted separately, and downstream self-loops are excluded.

Metric	Dataset	ASC	UGC	G-Skel.
Time (ms)	OGBN-Arxiv	20.31	52.20	22.36
	WikiCS	11.10	14.63	12.44
	Photo	11.24	22.55	12.54
	Computer	12.85	38.16	13.17
	Arxiv-year	15.92	38.46	16.92
Memory (MB)	OGBN-Arxiv	996	5,328	1,109
	WikiCS	86	690	116
	Photo	294	1,923	315
	Computer	511	2,884	531
	Arxiv-year	952	5,282	1,065
COO edges	OGBN-Arxiv	67.7K	2.24M	123.7K
	WikiCS	4.7K	307.7K	19.1K
	Photo	19.3K	838.9K	30.1K
	Computer	34.9K	1.23M	45.8K
	Arxiv-year	67.7K	2.24M	123.7K

Compression failures use the definitions in the main paper: OOM denotes failure within the 32-GB memory budget, OOT denotes a run exceeding the 180-minute time limit, and N/A denotes failure to construct a valid compressed graph within the accepted ratio interval. The corresponding method–dataset failures are reported in the main accuracy table and compression-time figure.

D Additional Ablations and Sensitivity

This section reports completed component-level analyses of ASC. All multi-seed tables use three different random seeds. For consistency across the paper, entries corresponding to the default ASC configuration use the results from the main comparison; the remaining entries report their dedicated ablation runs. In the following tables, "Full" denotes this default ASC configuration.

D.1 Anchor Selection

Table 14 compares four anchor selectors within the trainable ASC pipeline. The PageRank and Random columns change only the anchor selector; they are distinct from the training-free PR-Anchor and Random baselines in Table 8. Semantic medoids obtain the highest mean on seven of the 11 datasets. Random anchors lead on TAPE and Product-subset, whereas PageRank leads on Photo and Actor. We use semantic medoids as a label-free cross-dataset default rather than tuning the selector per dataset.

D.2 Training Objectives and Graph-Level Properties

Table 17(a) jointly summarizes the objective and input-signal ablations on all datasets. Removing semantic reconstruction reduces accuracy on every dataset. Removing graph-level property prediction also lowers the mean on every dataset, although its marginal effect is generally smaller and varies by graph. This supports its use as an auxiliary training signal rather than as the primary compression objective.

To examine whether the auxiliary property profile is driven by a single quantity, Table 17(b) removes each predicted property in turn on four representative datasets covering different graph domains and scales. The Full model attains the highest mean on all four datasets, while removing the entire property branch lowers the mean on all four. Individual removals have dataset-dependent marginal effects, indicating that the properties form a complementary and partially redundant graph-level profile rather than seven independently indispensable signals.

Table 14: Anchor-selection ablation at $r = 0.1$. Entries are mean node-classification accuracy (%) $\pm$ standard deviation over three runs.

Dataset	SKM-Medoid	PageRank	Random	Degree
PubMed	$\mathbf{88.36 \pm 0.51}$	88.35 ± 0.57	88.24 ± 0.72	88.00 ± 0.65
OGBN-Arxiv	$\mathbf{74.01 \pm 0.32}$	73.65 ± 0.57	30.09 ± 3.21	71.68 ± 0.36
WikiCS	$\mathbf{82.22 \pm 0.99}$	82.17 ± 0.77	82.12 ± 0.13	81.69 ± 0.37
TAPE	74.52 ± 0.63	74.44 ± 0.37	$\mathbf{75.38 \pm 0.07}$	73.26 ± 0.41
Product-subset	83.85 ± 0.79	83.89 ± 0.74	$\mathbf{84.15 \pm 0.14}$	83.61 ± 0.55
CiteSeer	$\mathbf{77.90 \pm 0.57}$	77.49 ± 1.01	76.85 ± 0.09	76.38 ± 1.31
Cora	$\mathbf{87.88 \pm 2.03}$	87.52 ± 1.92	87.39 ± 1.84	87.70 ± 1.38
Photo	79.02 ± 0.64	$\mathbf{79.34 \pm 0.55}$	79.18 ± 0.75	74.74 ± 0.70
Computer	$\mathbf{85.64 \pm 0.28}$	84.89 ± 0.24	83.63 ± 0.31	83.54 ± 0.58
Actor	70.19 ± 1.39	$\mathbf{70.53 \pm 0.59}$	70.30 ± 2.70	68.91 ± 1.90
Arxiv-year	$\mathbf{43.71 \pm 0.08}$	43.57 ± 0.08	35.25 ± 3.23	40.59 ± 0.10

Table 15: Assignment hyperparameter sensitivity at $r = 0.1$. Entries report changes in three-run mean accuracy, in percentage points, relative to the corresponding default. Bold columns denote the defaults.

Param.	Value	PubMed	Arxiv	Cora	Computer
Temperature τ	0.1	-0.1	-1.3	-0.7	-1.4
	0.5	**0.0**	**0.0**	**0.0**	**0.0**
	1.0	-0.3	-0.6	-0.7	-1.1
	2.0	-0.1	-0.5	-0.7	-0.9
Assign. budget k	1	-2.1	-11.4	-4.6	-3.6
	2	$+0.3$	-5.9	-0.8	-2.4
	3	**0.0**	**0.0**	**0.0**	**0.0**
	5	$+0.2$	-1.4	$+0.0$	$+0.2$

Table 16: Accuracy sensitivity to the skeleton-edge budget b at $r = 0.1$. Entries are mean node-classification accuracy (%) $\pm$ sample standard deviation over three runs. The default is $b = 4$.

Dataset	$b = 2$	$\mathbf{b = 4}$ (default)	$b = 8$
PubMed	88.45 ± 0.56	$\mathbf{88.36 \pm 0.51}$	88.04 ± 0.67
OGBN-Arxiv	72.33 ± 0.37	$\mathbf{74.01 \pm 0.32}$	72.59 ± 0.25
Cora	87.88 ± 1.20	$\mathbf{87.88 \pm 2.03}$	87.70 ± 1.79
Computer	84.73 ± 0.51	$\mathbf{85.64 \pm 0.28}$	84.66 ± 0.99

D.3 Input Feature Quality

Table 17(a) also separates semantic features from structural descriptors. Removing semantic features causes the largest drop on every dataset, whereas the effect of removing structural descriptors is much smaller and dataset-dependent. The complete model nevertheless has the highest mean on all 11 datasets in this matched component ablation. These results establish the importance of semantic inputs to the complete ASC pipeline, while structural descriptors provide complementary assignment signals.

Table 17(c) compares frozen Qwen3 embeddings with each dataset's native input features under a matched three-seed protocol. The four conditions use the same data splits, seeds, and downstream GCN settings; the native-feature conditions rerun ASC with the corresponding input features. We use "native features" rather than "BOW" because the provided feature type differs across datasets. The Qwen3 Original and Qwen3 ASC columns use the default results reported in the main comparison, while the native-feature columns report the dedicated feature-source runs.

Table 17: Objective, input-signal, graph-property, and feature-source analyses at $r = 0.1$. Entries are mean node-classification accuracy (%) $\pm$ standard deviation over three runs. In (a), "w/o Rec.", "w/o Prop.", "w/o Struct.", and "w/o Sem." remove semantic reconstruction, graph-level property prediction, structural descriptors, and semantic features, respectively. In (b), "w/o All" removes the complete property-prediction branch; the remaining abbreviations denote the individual graph properties defined in Appendix A.3. Panel (c) compares Qwen3 and native features with and without compression.

(a) Core objective and input-signal ablations

Dataset	Full	w/o Rec.	w/o Prop.	w/o Struct.	w/o Sem.
PubMed	$\mathbf{88.36 \pm 0.51}$	83.83 ± 1.23	87.74 ± 0.38	87.61 ± 0.43	44.80 ± 5.60
OGBN-Arxiv	$\mathbf{74.01 \pm 0.32}$	70.17 ± 1.13	72.10 ± 0.22	71.99 ± 0.35	15.50 ± 1.57
WikiCS	$\mathbf{82.22 \pm 0.99}$	60.97 ± 11.35	80.39 ± 0.80	75.64 ± 0.43	19.68 ± 1.46
TAPE	$\mathbf{74.52 \pm 0.63}$	58.40 ± 6.93	74.30 ± 0.27	72.27 ± 3.64	24.37 ± 2.43
Product-subset	$\mathbf{83.85 \pm 0.79}$	66.04 ± 6.17	82.74 ± 0.62	83.36 ± 0.66	15.74 ± 4.95
CiteSeer	$\mathbf{77.90 \pm 0.57}$	74.61 ± 0.72	77.14 ± 0.88	76.38 ± 1.76	23.67 ± 2.46
Cora	$\mathbf{87.88 \pm 2.03}$	80.63 ± 2.09	87.33 ± 1.68	87.15 ± 1.43	29.77 ± 6.10
Photo	$\mathbf{79.02 \pm 0.64}$	66.43 ± 8.18	75.89 ± 3.69	78.83 ± 0.84	41.83 ± 0.28
Computer	$\mathbf{85.64 \pm 0.28}$	67.26 ± 9.34	81.01 ± 3.08	84.66 ± 0.29	26.92 ± 3.55
Actor	$\mathbf{70.19 \pm 1.39}$	66.44 ± 0.86	69.76 ± 2.61	68.29 ± 1.80	48.87 ± 0.00
Arxiv-year	$\mathbf{43.71 \pm 0.08}$	37.16 ± 0.54	39.43 ± 0.21	43.49 ± 0.37	30.10 ± 2.28

(b) Leave-one-property-out ablation

Setting	OGBN-Arxiv	Product-subset	Cora	Actor
Full	$\mathbf{74.01 \pm 0.32}$	$\mathbf{83.85 \pm 0.79}$	$\mathbf{87.88 \pm 2.03}$	$\mathbf{70.19 \pm 1.39}$
w/o Fied.	72.30 ± 0.39	83.40 ± 0.78	87.52 ± 1.71	67.91 ± 1.64
w/o Spec.	72.57 ± 0.26	83.50 ± 0.62	85.92 ± 2.03	69.38 ± 0.23
w/o Clust.	72.31 ± 0.53	83.24 ± 1.38	87.45 ± 1.46	69.19 ± 2.46
w/o Wiener	72.74 ± 0.25	83.72 ± 0.60	86.84 ± 2.77	69.01 ± 2.17
w/o Diam.	72.60 ± 0.27	83.84 ± 0.64	85.67 ± 1.59	69.04 ± 3.43
w/o Deg. Rat.	72.37 ± 0.11	83.00 ± 0.55	85.24 ± 2.23	68.06 ± 2.21
w/o Assort.	70.01 ± 4.42	83.58 ± 0.26	86.04 ± 1.57	69.57 ± 1.67
w/o All	71.56 ± 0.61	82.71 ± 0.67	87.33 ± 1.36	69.19 ± 1.74

(c) Input-feature comparison

Dataset	Qwen3 Original	Qwen3 ASC	Native Original	Native ASC
PubMed	90.62 ± 0.13	88.36 ± 0.51	87.91 ± 1.00	83.98 ± 1.69
OGBN-Arxiv	75.40 ± 0.25	74.01 ± 0.32	68.34 ± 0.06	63.31 ± 0.80
WikiCS	84.98 ± 0.97	82.22 ± 0.99	80.34 ± 0.14	49.79 ± 21.84
TAPE	78.14 ± 0.25	74.52 ± 0.63	66.60 ± 0.20	50.99 ± 8.67
Product-subset	86.17 ± 0.53	83.85 ± 0.79	82.88 ± 0.48	57.89 ± 9.75
CiteSeer	80.51 ± 1.01	77.90 ± 0.57	69.12 ± 0.22	70.28 ± 1.30
Cora	89.85 ± 1.29	87.88 ± 2.03	82.12 ± 0.17	78.67 ± 0.94
Photo	86.00 ± 0.15	79.02 ± 0.64	86.46 ± 0.09	79.48 ± 0.44
Computer	89.29 ± 0.15	85.64 ± 0.28	89.77 ± 0.16	84.38 ± 0.16
Actor	73.49 ± 0.46	70.19 ± 1.39	71.47 ± 0.06	60.37 ± 1.41
Arxiv-year	48.38 ± 0.14	43.71 ± 0.08	43.44 ± 0.22	38.76 ± 0.24

Qwen3 improves original-graph accuracy on 9 of 11 datasets and ASC accuracy on 10 of 11 datasets. The compression-stage differences are particularly large on WikiCS, TAPE, and Product-subset, where native-feature ASC is also substantially less stable across seeds. Photo is the only dataset on which native features produce higher ASC accuracy. The results therefore support the use of high-quality pretrained semantics as ASC's primary input, but also show that the method is sensitive to feature quality rather than independent of the text representation.

D.4 Hyperparameter and Structural Sensitivity

Tables 15 and 16 summarize three sensitivity analyses on four representative datasets. The default $\tau = 0.5$ attains the highest mean for every dataset, with a maximum range of 1.4 percentage points over the tested temperatures. Top-1 assignment is markedly worse, especially on OGBN-Arxiv and Cora, whereas $k = 3$ provides a balanced sparse default among the multi-anchor settings. For the four datasets in Table 16, $b = 8$ does not improve mean accuracy over the default $b = 4$; increasing the edge budget therefore does not yield a monotonic accuracy gain in these comparisons.