CR-GRAPHORMER: From Cascades to Tokens via Mesoscopic Graph Rewiring

Meher Chaitanya⋆
Division of Theoretical Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden 114 28
mcpi@kth.se

My Le⋆
Department of Applied Mathematics and Statistics
Johns Hopkins University
Baltimore, MD 21210
mle19@jh.edu

Luana Ruiz
Department of Applied Mathematics and Statistics
Johns Hopkins University
Baltimore, MD 21210
lrubini1@jh.edu

Abstract

Graph transformers (GTs) match or surpass GNN performance by applying global self-attention, yet their quadratic memory requirement makes them impractical, and their receptive field is often limited by neighborhood aggregation, as fine-grained structural signals—especially in heterophilous graphs—are lost. We propose the **Cascade-Rewired Graph Transformer** (CR-GRAPHORMER), which balances computational efficiency with rich structural awareness. This is achieved by constructing an auxiliary network in which each node is assigned a token based on a "mesoscopic edge rewiring" process generated through deterministic contagion cascades initiated from its ego-network. Replacing long multi-hop paths with direct edges in the auxiliary network yields a backbone that captures higher-order structures while retaining sparsity. The rewiring amplifies homophilous ties and preserves critical heterophilous connections present in the extended neighborhoods of each node, providing every vertex with a compact, information-rich context that reflects local motifs and global reach. Each node retains a fixed-length token list drawn from its mesoscopic neighbors; because self-attention is confined to these constant-size sequences, CR-GRAPHORMER achieves $\mathcal{O}(E)$ complexity in graph tokenization, producing an expressive yet scalable model. We evaluate our proposed approach on 14 benchmark datasets spanning both homophilic and heterophilic settings and observe improvements in node classification accuracy on 10 of them. These results demonstrate that tokenizing over the "mesoscopic rewired graph" introduces a strong inductive bias that enhances graph learning.

1 Introduction

Graph-structured data arise in a wide range of domains, from physical systems to virtual platforms, including applications in anomaly detection [Deng and Hooi, 2021], traffic forecasting [Kong *et al.*, 2024], and recommendation systems [Agrawal *et al.*, 2024]. Graph Neural Networks (GNNs) are widely adopted for these tasks owing to their ability to capture local structure through message passing [Hamilton *et al.*, 2017]. However, their reliance on neighborhood aggregation introduces

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Advancing Graph Machine Learning.

well-known limitations, including over-smoothing [Chen *et al.*, 2020] and over-squashing [Alon and Yahav, 2020], which can hinder their ability to model long-range dependencies.

Graph transformers (GTs) have emerged as a powerful alternative for graph representation learning because their global self-attention can directly capture long-range dependencies: treating each node feature vector as a token, a GT allows every node to attend to every other node within a single layer.

Yet, unlike words in a sentence or pixels on a grid, graph nodes possess no canonical ordering; without additional signals, the attention mechanism cannot distinguish whether two tokens are adjacent, several hops apart, or connected by specific edge types. To remedy this, modern GTs inject *positional encodings* that embed structural information into the input tokens before attention is applied. These encodings range from global descriptors such as Laplacian eigenvectors to local cues like node degrees, shortest-path distances, or edge attributes that convey pairwise relations. By serving as inductive biases, positional encodings enable transformers to interpret a graph's irregular topology and have been key to their empirical performance [Kreuzer *et al.*, 2021; Zhao *et al.*, 2021; Ying *et al.*, 2021].

Despite their expressive capacity, dense self-attention in GTs scales poorly. Computing all token-wise interactions incurs a prohibitive $\mathcal{O}(n^2)$ cost in both memory and runtime [Dwivedi and Bresson, 2020; Mialon *et al.*, 2021; Wu *et al.*, 2021]. This inefficiency is compounded by the lack of structure in the attention pattern, which can lead to noisy representations by weighting irrelevant or weakly connected node pairs [Zhou *et al.*, 2025]. In practice, the very mechanism that enables long-range reasoning may hinder performance on large or sparse graphs.

To address these concerns, sparse attention has emerged as a scalable alternative. Rather than allocating one token per node, sparse GTs construct compact, node-specific sequences that encode only the most structurally relevant context, reducing attention complexity to nearly linear [Zhou *et al.*, 2025]. For instance, NAGphormer [Chen *et al.*, 2023] leverages spectral features to define token neighborhoods, while VCR-Graphormer [Fu *et al.*, 2024] uses Personalized PageRank (PPR) to guide token selection. However, these methods introduce their own trade-offs: spectral encodings are often costly to compute, and PPR tends to concentrate attention around nearby high-degree nodes [Andersen *et al.*, 2006], which can narrow the model's receptive field.

The localized tokenization employed not only by VCR but also by sparse attention in general, while efficient, can obscure long-range structural dependencies. Multi-hop tokens often compress information from distant nodes into coarse representations, diminishing resolution and limiting expressiveness. This loss of granularity is particularly problematic in heterophilic graphs, where nodes with similar features are typically far apart and sparsely connected [Zhou *et al.*, 2025].

To overcome these limitations, we propose the Cascade-Rewired Graph Transformer (CR-GRAPHORMER), a new architecture inspired by contagion dynamics in social systems [Granovetter, 1978; Centola and Macy, 2007; Centola, 2018]. Rather than relying on first-order random walks or static neighborhood heuristics, CR-GRAPHORMER generates node-specific token sets using cascades—interpretable as higher-order random walks—that adaptively propagate through the graph. We prove that these cascades quantify connectivity between node pairs in a way that preserves fine-grained local structure without introducing locality bias (Theorem 3.1). Crucially, they can be computed in constant time per node [Chaitanya *et al.*, 2025], making the method scalable to large graphs while retaining sensitivity to both nearby and distant interactions.

2 Preliminaries

This section provides a high-level overview of contagion dynamics and describes the specific cascade mechanisms we adopt for tokenization.

Let $G = (V, E, W)$ be an undirected, weighted graph with $n = |V|$, $m = |E|$, and a weight function $W : E \to \mathbb{R}$ assigning weights to the edges in E. For $i \in V$, denote $N(i) = \{\, j \in V : (i, j) \in E \,\}$, $N[i] = N(i) \cup \{i\}$, $d(i) = |N(i)|$.

Contagion Model. A contagion on G is a process $\{\mathbf{x}_t\}_{t=0}^{T}$, where $\mathbf{x}_t \in \{0, 1\}^n$ denotes the state of the contagion at time t, and $[\mathbf{x}_t]_i = 1$ indicates that node i is activated.

The monotone *active set* for a vertex v at time T, denoted S_t^v, is defined as $S_t^v = \{ i \in V \mid [\mathbf{x}_t]_i = 1,\ 0 \le t \le T \}$. For an inactive vertex $u \notin S_t^v$, its instantaneous *neighbor support* is $\sigma_t(u) = |N(u) \cap S_t^v|$.

Mesoscopic Rewiring via Deterministic Cascades. For every vertex u and a subset R of its ego-neighborhood, we launch fixed-length *deterministic contagion cascades*—memoryless higher-order random walks—under the two activation rules of Chaitanya *et al.* [2025], each resulting in a discrete-time tokenization scheme:

- **Adaptive-Threshold Contagion (MAS)**: activating the inactive vertex with the maximum active-neighbor support (i.e., the threshold is *adaptive*);
- **Absolute-Threshold Contagion (TAS)**: activating a vertex once at least τ of its neighbors are active (given an *absolute threshold* τ).

Executing these cascades over multiple random R produces constant-length activation traces. For each source node, we tally the most frequently co-activated vertices, keep the top k, and connect them to the source node to build a mesoscopic rewired structural auxiliary graph G^*. The rewired graph G^* turns multi-hop yet structurally salient pairs into direct neighbors while pruning weak links [Centola and Macy, 2007]. The constructed network increases effective homophily—even in originally homophilic or heterophilic settings—thereby enabling accurate predictions across a diverse range of network types. This leads to better node classification performance across both homophilic and heterophilic benchmarks. Notably, the entire cascade-based rewiring process operates in $\mathcal{O}(|E|)$ time, providing a significantly more efficient and structurally grounded alternative to traditional sparse graph tokenization approaches.

Self-Attention. Following Vaswani *et al.* [2017], we use the standard self-attention layer:

$$\mathbf{H}' = \mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d'}}\Big)\mathbf{V} \in \mathbb{R}^{n \times d'}, \tag{1}$$

where the queries $\mathbf{Q} = \mathbf{H}\mathbf{W}_Q$, the keys $\mathbf{K} = \mathbf{H}\mathbf{W}_K$, and the values $\mathbf{V} = \mathbf{H}\mathbf{W}_V$, $\mathbf{H} \in \mathbb{R}^{n \times d}$ is the input feature matrix, $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d'}$ are learnable projections, and the output feature matrix is $\mathbf{H}' \in \mathbb{R}^{n \times d'}$.

3 Cascade-Rewired Graphormer (CR-GRAPHORMER) Architecture

In this section, we extend the mini-batch training strategy proposed by Fu *et al.* [2024], adapting it to our cascade-based framework for effective transformer training on both homophilic and heterophilic networks. For a given undirected graph $G = (V, E, W)$, we construct two *cascade-rewired* auxiliary graphs based on local activation dynamics, G^* and $\tilde{G}$, capturing higher-order structural and feature-aware signals. We then describe how fixed-length token lists are derived from these rewired graphs, enabling efficient and expressive mini-batch processing.

3.1 Mesoscopic Edge Rewiring

To capture homophily, we perform adaptive and absolute threshold-activation simulations on G. For each node $v \in V$, we generate cascades of a constant *walk length* $\ell \in \mathbb{N}$ with ℓ activation steps under two regimes:

- **Adaptive-Threshold Contagion (MAS)**: For each node $v \in V$, draw a random subset $R \subseteq N(v)$ from its immediate neighbors. The cascade is initialized by activating $S_0^v = \{v\} \cup R$. At each step $t \in [0, \ell - 1]$, compute $\tau_t = \max_{u \notin S_t^v} \sigma_t(u)$ and activate $u_t^\star \in \arg\max_{u \notin S_t^v} \sigma_t(u)$, the most strongly reinforced inactive vertex, by adding it to the active set. The active set of v is updated as $S_{t+1}^v = S_t^v \cup \{u_t^\star\}$. In our experiments, we refer to this procedure as CR-ADAPTIVE.

  Since $\tau_{t+1} \le \tau_t$ for all t, the reinforcement required for activation decreases monotonically, so vertices highly embedded in the seed's neighborhood join the cascade early. In contrast, vertices in sparser regions are activated only after the threshold has fallen sufficiently. Thus, MAS exhaustively explores densely connected regions before crossing weak cuts, and the contagion process propagates preferentially through cohesive clusters.

- **Absolute-Threshold Contagion (TAS)**: For each node $v \in V$, select a random subset $R \subseteq N(v)$ of its first-order neighbors. The contagion is initialized by activating $S_0^v = \{v\} \cup R$. At each subsequent step $t \in [0, \ell - 1]$, find $\mathcal{A}_t = \{ u \in V \setminus S_t^v \mid \sigma_t(u) \geq \tau \}$. If $\mathcal{A}_t \neq \varnothing$, select one $u_t^\star \in \mathcal{A}_t$ (e.g., uniformly at random) to activate and set $S_{t+1}^v = S_t^v \cup \{u_t^\star\}$, regulating a node to be activated once at least τ of its neighbors are in the active set. This procedure is repeated for $\tau \in \{1, \dots, N\}$, with $N = 5$ as the default value. In our experiments, we refer to this procedure as CR-ABSOLUTE.

  When $\tau = 1$, the procedure collapses to breadth-first search, whereas larger τ values confine propagation to vertices receiving strong multi-neighbor reinforcement [Granovetter, 1978; Centola, 2018]. These cascades can also be interpreted as a *higher-order, memoryless random walk* whose state depends only on the currently activated frontier.

Starting from the target node $\mathbf{v}$, together with different (random) subsets of its immediate neighborhood, we record every active set (cascade) as an ordered list of activated vertices until either no further activations occur or the walk length limit ℓ, a constant, is reached using the CR-ADAPTIVE and CR-ABSOLUTE procedures. These sequences reveal which vertices repeatedly influence one another through multi-hop paths. Below, we describe the construction of the auxiliary graphs using the CR-ADAPTIVE and CR-ABSOLUTE procedures.

Cascade-Rewired Auxiliary Graph Construction. Once a cascade procedure—either CR-ADAPTIVE or CR-ABSOLUTE—is selected, each execution yields an *activation sequence*, that is, an ordered active set denoted by $S_\ell^v = (v, u_1, \dots, u_\ell)$. Let $\mathcal{M}_v$ be the multiset of vertices appearing in the union of all active sets rooted at v. Define the frequency map $f_v(u) =$ multiplicity of u in $\mathcal{M}_v, u \in V$. Retain the k most frequent vertices for each $v \in V$:

$$F_k(v) = \arg \operatorname*{top\text{-}k}_{u \in V} \{f_v(u)\}, \tag{2}$$

and construct an auxiliary activation graph with edges $E_{\text{act}}^v = \{(v, u) \mid u \in F_k(v),\ \forall v \in V\}$. This process is performed for all $v \in V$. The *cascade-rewired structural auxiliary graph* is then given by $G^* = (V^*, E^*, W^*)$, where $E^* = \bigcup_{v \in V} E_{\text{act}}^v$ and $V^* = V$. Because vertices across weak cuts appear rarely in the cascades, G^* accentuates cohesive community structure. Early positions in the activation order correspond to nodes tightly coupled to the source, while later positions may mark peripheral or weakly connected vertices. Because both absolute and adaptive contagions privilege multi-edge reinforcement, the sequences naturally reveal community boundaries and higher-order motifs. A noteworthy by-product of the mesoscopic rewiring step, G^*, prior to any contextual-edge augmentation, is a **measurable rise in node-level homophily for heterophilic networks**, as reported in Table 2.

How to Choose the Walk Length ℓ? The walk length ℓ sets the spatial resolution of the rewiring procedure. Restricting each contagion to at most ℓ propagation steps focuses the analysis on a mesoscopic neighborhood of the seed—small enough to prevent global dilution, yet wide enough to reveal multi-hop motifs such as triangles, cliques, and other short cycles. Intuitively, we seek to minimize the effective resistance between the starting active set associated with v, S_0^v, and any vertex u that becomes activated within a walk of length at most ℓ. To formalize this intuition, we derive the following upper bound:

Theorem 3.1. *Let $G = (V, E)$ be an undirected graph, and let $\boldsymbol{\theta} : V \to \mathbb{N}$ be a threshold function governing a deterministic contagion process with uniform thresholds. Let $S_0 := \{v\} \cup R$ for some node $v \in V$ and subset $R \subseteq N_G(v)$. Denote by S_t^v the set of nodes activated until time t when the contagion starts from v together with a subset of its neighborhood R. Let $G' \subseteq G$ be the subgraph activated by S_0^v.*

Suppose a node $u \in V \setminus S_0$ satisfies:

(i) $\boldsymbol{\theta}(u) = \tau$ *for some* $\tau \geq 1$,

(ii) $u \in S_\ell^v$*, i.e., u becomes active in at most ℓ time steps.*

Then the effective resistance between the active set S_0^v and u in G', denoted $R_{\text{eff}}^{G'}(S_0^v, u)$, satisfies $R_{\text{eff}}^{G'}(S_0^v, u) \leq \frac{\ell}{\tau}$.

Theorem 3.1, whose proof is provided in Appendix B, follows directly from the following two lemmas:

Lemma 3.2. *If u activates at threshold τ in at most ℓ time steps, then G' contains τ edge-disjoint S_0^v–u paths of length $\leq \ell$.*

Lemma 3.3. *If G' contains τ edge-disjoint S_0^v–u paths $P_1, \ldots, P_\tau$ with $|P_i| \leq \ell$ for all i, then $R_{\text{eff}}^{G'}(S_0^v, u) \leq \frac{\ell}{\tau}$.*

Remark. If u activates within ℓ steps, Lemma 3.2 provides the τ disjoint paths, and Lemma 3.3 yields $R_{\text{eff}}^{G'}(S_0^v, u) \leq \frac{\ell}{\tau}$. If G' consists solely of τ length-ℓ chains in parallel between S_0^v and u, equality $R_{\text{eff}}^{G'} = \ell/\tau$ is attained, showing that the bound is tight. The proofs of Lemmas 3.2 and 3.3 are provided in Appendix B.

Theorem 3.1 implies that if a vertex becomes activated by $S_0 = \{v\} \cup R$ under a higher threshold τ, its activation must be supported by multiple short paths originating from v, reflecting strong local reinforcement. In other words, a higher activation threshold signals that the vertex is more cohesively connected to v. Furthermore, since the upper bound on effective resistance grows proportionally with the walk length ℓ, we select a *smaller value* of ℓ in our experiments to promote tighter structural connectivity.

This bound is also crucial for maintaining sparsity: a vertex, together with its chosen random subset of neighbors, inspects no more than the ℓ-hop frontier before selecting its top-k targets, thereby reducing the number of auxiliary edges added to the graph.

Time Complexity. Our time-complexity analysis mirrors that of Chaitanya *et al.* [2025]. Throughout the generation of contagion cascades, we maintain a dictionary that records each node's activation frequency count and sort these counts in descending order for each corresponding source node. Since the active set at any vertex (source) is bounded by the fixed constants—the walk length ℓ, the number of neighbors k, the number of permutations N, and, in the case of Absolute Threshold Contagions, the threshold τ—this sorting step requires only constant time. Consequently, the overall time complexities of Adaptive and Absolute Threshold Contagions are $\mathcal{O}(n + m)$, since $\{\ell, k, N, \tau\} \in \mathcal{O}(1)$, and their linear dependence on the number of edges ensures computational efficiency.

3.2 Token-List Formation

The token list for a target node v is constructed using its original features, $\mathbf{x}_v$, and the neighbors of v in the cascade-rewired auxiliary graph G^*. These lists are then self-attended by the standard transformer [Vaswani *et al.*, 2017] to produce the target node representation vector. The complete fixed-length token list for node v is given by

$$\mathbf{T_v} = \{ \underbrace{\mathbf{x}_v \| 1}_{\text{self}}, \quad \underbrace{(\mathbf{F}^* \mathbf{X})(v, :) \| \omega_v^*}_{\text{Freq. agg. of } v\text{'s nbhd. in } G^*} \} \tag{3}$$

where "$\|$" denotes the concatenation operation, "nbhd" stands for the neighborhood of the vertex, and $\mathbf{F}^*$ denotes the frequency matrix for G^*. To calculate $\mathbf{F}^*$, we first identify the number of times a target node v with various (random) subsets of its neighborhood activates another node u, denoted as a frequency $\mathbf{f}_{vu}$. Here, we can opt to further normalize $\mathbf{f}_{vu}$, either globally $\mathbf{f}_{vu} = \frac{\mathbf{f}_{vu}}{\max_{s,t} \mathbf{f}_{st}}$ with s and t ranging over all nodes in G^*, or locally $\mathbf{f}_{vu} = \frac{\mathbf{f}_{vu}}{\max_{t \in N(v)} \mathbf{f}_{vt}}$ with t ranging over v's neighbors—*cross-validation* can be employed to select the strategy that best suits the input data. Next, we calculate the entries of $\mathbf{F}^*$ as $\mathbf{F^*}_{uv} = \mathbf{F^*}_{vu} = \frac{1}{2}(\mathbf{f}_{vu} + \mathbf{f}_{uv})$ and, finally, set ω_v^* to $\mathbf{F}^*(:, v)$.

3.3 Transformer Encoder

With $\mathbf{Z}_v^{(0)} = \mathbf{T}_v$ as the node-specific input sequence, each encoder layer follows the standard formulation:

$$\begin{aligned} \tilde{\mathbf{Z}}_v^{(t)} &= \text{MHA}\big(\text{LN}(\mathbf{Z}_v^{(t-1)})\big) + \mathbf{Z}_v^{(t-1)}; \\ \mathbf{Z}_v^{(t)} &= \text{FFN}\big(\text{LN}(\tilde{\mathbf{Z}}_v^{(t)})\big) + \tilde{\mathbf{Z}}_v^{(t)}, \end{aligned} \tag{4}$$

MHA is multi-head self-attention, FFN is a position-wise feed-forward network, and LN is layer normalization. After T layers, a read-out (e.g., mean pooling) over $\mathbf{Z}_v^{(T)}$ yields the final embedding of node v. We integrate *feature-cluster information* and *mesoscopic rewiring* (capturing long-range influence) within the framework. Because all tokens are pre-selected, training remains mini-batchable with sequence length $1 + k$, independent of the original graph size, where k is the degree of the mesoscopic auxiliary graph, G^*.

4 Experiments

Experimental Setup. We evaluate the node classification performance of our proposed models, CR-ADAPTIVE and CR-ABSOLUTE, on 14 publicly available benchmark networks from the DGL library, which serve as standard benchmarks for node classification across both homophilic and heterophilic network settings[1]. For the CR-ADAPTIVE and CR-ABSOLUTE procedures, we fix the parameters as follows: neighborhood subset size $|R| = 5$, cascade walk length $\ell = 10$, and the k in equation (2) is set to the average degree of the graph. For CR-ABSOLUTE, we additionally sweep the threshold parameter τ over the range $[1, 5]$. *The constructed mesoscopic rewired auxiliary graph is split into training, validation, and test sets* in a 50%–25%–25% ratio and passed through a transformer model. Note that we use a different split percentage than the traditional one to thoroughly evaluate the robustness of all methods. We run experiments using 20 random data splits per graph to ensure thorough robustness of all the models. The results are provided in Table 1.

Method	Actor	Chameleon	Community	Computer	Cornell	Cycle	Grid
PPRGO	31.36 ± 1.14	45.83 ± 2.62	39.43 ± 2.19	**91.13 ± 0.58**	46.91 ± 7.09	59.27 ± 2.73	58.19 ± 1.76
GRAND+	30.20 ± 1.08	45.54 ± 2.73	39.37 ± 2.01	90.85 ± 0.69	45.11 ± 6.46	59.27 ± 2.73	58.19 ± 1.76
SAN	34.07 ± 1.59	42.56 ± 4.18	40.16 ± 2.09	84.49 ± 0.89	**67.34 ± 10.29**	59.27 ± 2.73	58.19 ± 1.76
GraphGPS	**35.21 ± 0.90**	60.89 ± 2.65	46.21 ± 2.66	89.32 ± 0.68	64.47 ± 6.41	59.00 ± 3.02	68.01 ± 3.12
NAGphormer	31.69 ± 1.10	43.00 ± 1.88	51.66 ± 2.23	88.47 ± 0.72	58.30 ± 6.34	**97.28 ± 1.82**	**75.19 ± 4.57**
Exphormer	28.70 ± 3.31	40.10 ± 7.73	40.19 ± 2.84	OOM	60.00 ± 10.77	52.24 ± 9.62	56.10 ± 5.87
VCR-Graphormer	33.18 ± 1.34	45.96 ± 2.60	42.27 ± 2.35	89.49 ± 1.24	58.40 ± 6.49	66.62 ± 10.90	66.07 ± 5.02
CR-Adaptive	31.86 ± 1.50	62.69 ± 2.24	48.39 ± 2.30	89.50 ± 0.99	57.23 ± 5.16	58.38 ± 2.76	69.92 ± 2.18
CR-Absolute	34.17 ± 1.04	**64.77 ± 2.12**	**63.57 ± 3.62**	89.70 ± 0.71	65.00 ± 6.75	62.76 ± 3.04	65.45 ± 2.96

Method	Photo	Pubmed	Shape	Squirrel	Texas	Wiki	Wisconsin
PPRGO	**95.09 ± 0.63**	86.72 ± 0.39	43.11 ± 2.97	30.22 ± 1.56	58.51 ± 5.46	**84.16 ± 0.98**	55.63 ± 4.90
GRAND+	94.82 ± 0.59	85.95 ± 0.43	43.11 ± 2.97	30.23 ± 1.54	56.49 ± 6.34	83.38 ± 0.89	53.20 ± 5.33
SAN	91.12 ± 0.61	85.81 ± 0.96	43.11 ± 2.97	28.23 ± 2.48	72.23 ± 7.42	79.84 ± 1.55	**77.19 ± 5.30**
GraphGPS	94.59 ± 0.61	87.17 ± 0.55	66.83 ± 6.53	41.05 ± 1.11	**74.57 ± 6.27**	82.94 ± 0.81	72.66 ± 4.23
NAGphormer	94.25 ± 0.70	87.48 ± 0.54	59.74 ± 6.09	31.50 ± 1.36	62.34 ± 7.81	82.81 ± 1.06	66.17 ± 3.94
Exphormer	81.75 ± 6.97	OOM	40.03 ± 8.10	27.13 ± 6.13	70.11 ± 11.25	OOM	72.66 ± 5.90
VCR-Graphormer	94.24 ± 1.17	86.28 ± 0.73	48.31 ± 10.16	35.49 ± 1.69	65.96 ± 5.39	83.52 ± 1.03	71.02 ± 6.01
CR-Adaptive	94.17 ± 0.78	87.66 ± 0.61	55.26 ± 5.63	45.89 ± 2.14	65.21 ± 5.79	83.05 ± 0.88	70.63 ± 4.98
CR-Absolute	94.15 ± 0.55	**87.79 ± 0.52**	**76.26 ± 8.01**	**47.85 ± 2.26**	72.66 ± 6.07	83.20 ± 0.82	75.39 ± 4.81

Table 1: Node classification accuracies (mean ± std, in %) across datasets. OOM indicates that results could not be obtained due to an out-of-memory error. The best overall performance is highlighted in **bold**, while the second-best-performing method is shaded in **gray**.

Key Results. Compared with seven other competitive baselines, our proposed methods achieve either the best or second-best performance on 10 out of the 14 benchmark networks, even without any fine-tuning. The improvements are especially pronounced compared to the state-of-the-art Tokenized Graph Transformers (VCR-Graphormer and NAGphormer) on the challenging benchmarks: *Chameleon* (+18.8%), *Community* (+11.9%), *Cornell* (+6.6%), *Shape* (+16.5%), *Squirrel* (+12.4%), *Texas* (+6.7%), and *Wisconsin* (+4.4%). The increase in performance can be attributed to an important byproduct of our rewired auxiliary graph, G^*—its ability to improve label homophily scores even before the incorporation of virtual edges. This enhancement is particularly notable in heterophilic graphs rather than in homophilic graphs, as shown in Table 2.

[1]All datasets are accessible through the Deep Graph Library (DGL) at https://www.dgl.ai/dgl_docs/api/python/dgl.data.html. The code for our proposed methods is provided at https://anonymous.4open.science/r/CR-graphormer-4DC0/

	Actor	Chameleon	Community	Computer	Cornell	Cycle	Grid
Original Graph	0.222	0.248	0.540	0.785	0.118	0.925	0.921
Adaptive Contagion Rewired Graph	0.213	0.346	0.386	0.780	0.306	0.635	0.735
Absolute Contagion Rewired Graph	0.216	0.313	0.444	0.770	0.242	0.908	0.911

	Photo	Pubmed	Shape	Squirrel	Texas	Wiki	Wisconsin
Original Graph	0.836	0.792	0.591	0.218	0.087	0.659	0.171
Adaptive Contagion Rewired Graph	0.829	0.759	0.364	0.260	0.347	0.635	0.356
Absolute Contagion Rewired Graph	0.822	0.770	0.453	0.260	0.301	0.602	0.302

Table 2: Homophily scores across datasets. An increase in homophily scores is observed in several auxiliary networks compared to their original counterparts.

Ablation Study on CR-Adaptive (MAS) Models. We analyze the performance sensitivity of the proposed CR-ADAPTIVE mechanism as the key hyperparameters vary in relation to node classification accuracy. Specifically, we vary the maximum degree k in (2) used in constructing the cascade-rewired auxiliary graph, by setting $k \in \{5, 10, 15, 20\}$. Additionally, we test multiple configurations of the cascade walk length $\ell \in \{10, 20, 30, 40\}$, with the neighborhood subset size fixed at $|R| = \ell/2$. For most graphs, we observe from Figure 1 that the accuracy gradually decreases with increased walk length, as supported by Theorem 3.1.

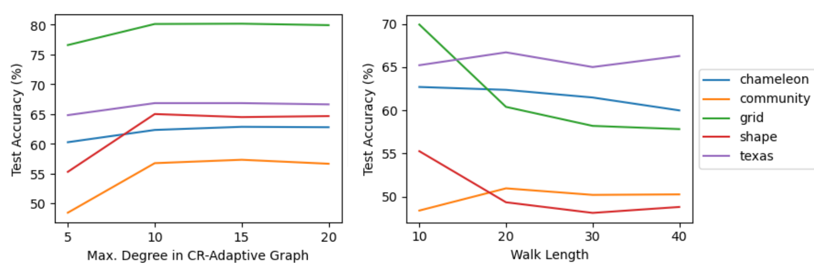

Figure 1: Ablation study on the cascade-rewired (CR) auxiliary graph with adaptive threshold.

An ablation study on CR-Absolute (TAS) models, along with dataset statistics and further experimental details, is provided in Appendix D.

5 Conclusion and Limitations

We introduced CR-GRAPHORMER, a scalable graph transformer that uses contagion-based rewiring to enrich graph structure via deterministic cascades. This process captures higher-order connectivity and reinforces homophilic and heterophilic ties through feature-aware augmentation. Our model attends to fixed-length token lists from local cascades and global context, enabling efficient mini-batch training. CR-GRAPHORMER achieves either the best or second-best performance on 10 out of the 14 benchmark networks, demonstrating the utility of cascade-based mesoscopic rewiring for structural encoding in Tokenized Graph Transformers.

One of the limitations of our proposed approaches is that, for sparse and highly regular networks such as cycles and grids, the proposed architecture may be less effective at rewiring, as the contagion-based procedures struggle to capture meaningful structural variations in such graphs. Future work could explore addressing this limitation by linking the effectiveness of the rewiring strategy to structural properties such as average degree or average path length, potentially enabling adaptive mechanisms for different graph topologies.

6 Acknowledgments

This work was partially supported by the Wallenberg AI Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Nimesh Agrawal, Anuj Kumar Sirohi, Sandeep Kumar, et al. No Prejudice! Fair Federated Graph Neural Networks for Personalized Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024.

Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and Its Practical Implications. *arXiv preprint arXiv:2006.05205*, 2020.

Reid Andersen, Fan Chung, and Kevin Lang. Local Graph Partitioning Using Pagerank Vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2464–2473, 2020.

Damon Centola and Michael Macy. Complex Contagions and the Weakness of Long Ties. *American Journal of Sociology*, 113(3):702–734, 2007.

Damon Centola. *How Behavior Spreads: The Science of Complex Contagions*, volume 3. Princeton University Press, 2018.

Meher Chaitanya, Kshitijaa Jaglan, and Ulrik Brandes. Adjacency Search Embeddings. *Transactions on Machine Learning Research*, 2025.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy Colwell, and Adrian Weller. Rethinking Attention with Performers. In *International Conference on Learning Representations*, 2021.

Ailin Deng and Bryan Hooi. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *Journal of Machine Learning Research*, 24(43):1–68, 2023. Originally released as arXiv:2003.00982 (2020).

Wenzheng Feng, Yuxiao Dong, Tinglin Huang, Ziqi Yin, Xu Cheng, Evgeny Kharlamov, and Jie Tang. Grand+: Scalable graph random neural networks. In *Proceedings of the ACM Web Conference 2022*, pages 3248–3258, 2022.

Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. VCR-Graphormer: A Mini-Batch Graph Transformer via Virtual Connections. In *The Twelfth International Conference on Learning Representations*, 2024.

Mark Granovetter. Threshold Models of Collective Behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

Florian Grötschla, Jiaqing Xie, and Roger Wattenhofer. Benchmarking Positional Encodings for GNNs and Graph Transformers. *arXiv preprint arXiv:2411.12732*, 2024.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure Transformers are Powerful Graph Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C Bayan Bruss, and Tom Goldstein. GOAT: A Global Transformer on Large-Scale Graphs. In *International Conference on Machine Learning*, pages 17375–17390. PMLR, 2023.

Weiyang Kong, Ziyu Guo, and Yubao Liu. Spatio-Temporal Pivotal Graph Neural Networks for Traffic Flow Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking Graph Transformers with Spectral Attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. *arXiv Preprint arXiv:2202.13013*, 2022.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding Graph Structure in Transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse Transformers for Graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017.

Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing Long-Range Context for Graph Neural Networks with Global Attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A Scalable Graph Structure Learning Transformer for Node Classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-Graph Transformer for Node Classification. *arXiv preprint arXiv:2110.13094*, 2021.

Zijie Zhou, Zhaoqi Lu, Xuekai Wei, Rongqin Chen, Shenghui Zhang, Pak Lon Ip, et al. Tokenphormer: Structure-Aware Multi-Token Graph Transformer for Node Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13428–13436, 2025.

Appendix

A Related Works

The existing research on graph transformers can be broadly classified into three categories:

Graph Transformer Architectures. Graph transformer (GT) models adapt the self-attention mechanism of the original transformer [Vaswani *et al.*, 2017] to graph-structured data. The Graph Transformer Network [Dwivedi and Bresson, 2020] applies dense self-attention across all $\mathcal{O}(n^2)$ node pairs, achieving considerable performance on several benchmarks. Later architectures inject graph-specific biases into the attention mechanism: Graphormer [Ying *et al.*, 2021] encodes node centrality and pairwise shortest-path distances; GraphiT [Mialon *et al.*, 2021] uses kernelized relative encodings and explicit sub-path features; SAN [Kreuzer *et al.*, 2021] and Gophormer [Zhao *et al.*, 2021] incorporate edge and node attributes; and GraphGPS [Rampášek *et al.*, 2022] interleaves message passing with global attention. Although these models achieve strong performance across domains such as recommendation, question answering, and bioinformatics, their fully connected attention still entails quadratic computational cost.

Positional Encodings for Graphs. Because graphs lack an intrinsic notion of order, GTs must rely on positional encodings (PEs) to inject structural information. Laplacian-based encodings (LapPE) [Dwivedi and Bresson, 2020], full spectra [Kreuzer *et al.*, 2021], and sign-invariant variants such as SignNet [Lim *et al.*, 2022] give each node absolute coordinates in a spectral space but can be sensitive to eigenvector permutations. Random-walk or diffusion encodings (RWSE, RWDiffusion, RRWP) [Grötschla *et al.*, 2024; Dwivedi *et al.*, 2023] capture multi-scale proximity, while shortest-path or distance encodings power Graphormer's attention biases [Ying *et al.*, 2021]. Weisfeiler–Lehman subtree encodings (WL-PE) [Morris *et al.*, 2019] and edge-aware learnable schemes such as PureGT's relative PEs [Kim *et al.*, 2022] further enrich the positional signal. Systematic studies [Rampášek *et al.*, 2022] show that combining complementary PEs (e.g., Laplacian + shortest-path) yields robust gains across heterogeneous benchmarks. Overall, well-designed PEs are now recognized as a primary driver of GT performance, enabling the model to discriminate roles, distances, and higher-order structural motifs.

Lightweight and Tokenized Graph Transformer Models. Quadratic attention limits the scalability of classical GTs, prompting a surge of efficient variants. GraphGPS replaces softmax attention with Performer's linear mechanism [Choromanski *et al.*, 2021], whereas Exphormer [Shirzad *et al.*, 2023] imposes expander-graph sparsity, and GOAT [Kong *et al.*, 2023] projects features to lower dimensions. NodeFormer [Wu *et al.*, 2022] introduces topology-aware relational biases, achieving provably linear complexity in the number of nodes. A parallel line of work tokenizes the graph to confine attention to compact, information-rich subsets. TokenGT [Kim *et al.*, 2022] treats every node and edge as an independent token and relies on node/edge IDs plus structural PEs, matching or exceeding GNNs on large molecular datasets. Tokenphormer [Zhou *et al.*, 2025] augments each node token with orthogonal random features and Laplacian frequencies to recover edge existence via token similarities. NAGphormer [Chen *et al.*, 2023] constructs a fixed-length list of hop-aggregate tokens for each node; self-attention is performed locally within each list, enabling mini-batch training and fine-grained neighborhood encoding. VCR-Graphormer [Fu *et al.*, 2024] further scales to larger graphs by sampling personalized PageRank neighborhoods and adding feature-cluster tokens, so that each node attends only to a small set of local and global context tokens. Because attention is restricted to these lists, the per-layer cost becomes linear in the number of edges while preserving expressivity for long-range and heterophilous signals. However, while VCR-Graphormer relies on virtual connections to capture dependencies, our proposed method overcomes this limitation by capturing both structure- and heterophily-driven relationships without the explicit use of virtual connections.

B Proofs of Theoretical Results

Proof of Lemma 3.2 We prove the lemma by performing induction on step/round t. In the base case, if $t^\star = 1$, then u is adjacent to at least τ seeds in S_0; these τ edges form the required paths.

Assume every vertex activated by step $t < t^\star$ has τ edge-disjoint paths of length $\leq t$ from S_0. Let x be activated at step $t+1$. Then x has τ distinct active neighbors $w_1, \ldots, w_k \in S_t^v$. For each i, choose one of the τ paths from S_0 to w_i (given by the induction hypothesis) and append the edge $w_i x$. The resulting τ paths are edge-disjoint, start in S_0, end in x, and have length $\leq t+1$. Applying this construction until $t^\star \leq \ell$ gives the desired τ paths from S_0 to u.

Proof of Lemma 3.3 Delete from G' every edge not on $\bigcup_i P_i$; call the pruned network H. Rayleigh monotonicity implies $R_{\text{eff}}^{G'}(S_0, v) \leq R_{\text{eff}}^{H}(S_0, v)$. As the graph has unit edge weights, i.e., the weight of every edge is one, in H, each path P_i is a series of $\ell_i \leq \ell$ unit resistors; thus, the resistance is ℓ_i. The τ paths lie in parallel, so $\frac{1}{R_{\text{eff}}^{H}(S_0,v)} = \sum_{i=1}^{\tau} \frac{1}{\ell_i} \geq \frac{\tau}{\ell}$, hence $R_{\text{eff}}^{G}(S_0, v) \leq \ell/\tau$.

Proof of Theorem 3.1 If v activates by round ℓ, Lemma 3.2 gives the τ edge-disjoint paths, and Lemma 3.3 yields $R_{\text{eff}}^{G'}(S_0, v) \leq \ell/\tau$.

C Pseudocodes for Obtaining the Multiset $\mathcal{M}_v$ Using CR-ADAPTIVE and CR-ABSOLUTE Models

In this section, we provide the pseudocode for CR-ADAPTIVE and CR-ABSOLUTE procedures described in Section 3. As mentioned in this work, we reinterpret the two adjacency-search rules of Chaitanya *et al.* [2025] as discrete-time contagion dynamics.

Adaptive Threshold Contagions rely on the following parameters:

- **Walk Length (ℓ)**: This parameter refers to the sequence length that is generated for every vertex in the graph. Default $= 10$.
- **Number of Neighbors (k')**: $k' = |R|$ where $R \in N(v)$. Default $= 5$.
- **Number of Permutations (N)**: This parameter assists in capturing diversified immediate neighborhoods of a vertex v. Default $= 5$.

Algorithm 1 Adaptive Threshold Contagions (MAS)

```
Input: Graph G, walk length ℓ, #neighbors k', #permutations N.
Output: Dictionary D that stores the activation frequencies of all nodes. For each starting node, the
dictionary is ordered by activation frequency in descending order.
for v in G.nodes() do
    neighbors ← v.neighbors()
    counter ← {u : 0 | u ∈ G.nodes()}
    if len(neighbors) > 0 then
        for n = 1, ..., N do
            neighbors ← permute(neighbors)
            for i = 0 to len(neighbors) with step k' do
                s ← min(i, max(0, len(neighbors) − k'))
                S ← [v] + neighbors[s : s + k']
                while len(S) < ℓ do
                    find the maximally adjacent node u to S
                    S.insert(u)
                end while
                for u in S do
                    increment counter[u]
                end for
            end for
        end for
    end if
    sort counter by its values in descending order
    D[v] ← counter
end for
return D
```

Absolute Threshold Contagions rely on the following parameters:

- **Threshold (τ)**: Default $= 5$.
- **Walk Length (ℓ)**: Default $= 10$.
- **Number of Neighbors (k')**: Default $= 5$.
- **Number of Permutations (N)**: Default $= 5$.

Algorithm 2 Absolute Threshold Contagions (TAS)

```
Input: Graph G, threshold τ, walk length ℓ, #neighbors k', #permutations N.
Output: Dictionary D that stores the activation frequencies of all nodes for each starting node, ordered by activation frequency in descending order.
for v in G.nodes() do
    neighbors ← v.neighbors()
    counter ← {u : 0 | u ∈ G.nodes()}
    if len(neighbors) > 0 then
        for τ' = 1, . . . , τ do
            for n = 1, . . . , N do
                neighbors ← permute(neighbors)
                for i = 0 to len(neighbors) with step k' do
                    s ← min(i, max(0, len(neighbors) − k'))
                    S ← [v] + neighbors[s : s + k']
                    S ← Delayed-BFS(G, S, τ', ℓ)
                    for u in S do
                        increment counter[u]
                    end for
                end for
            end for
        end for
    end if
    sort counter by its values in descending order
    D[v] ← counter
end for
return D
```

Algorithm 3 Delayed-BFS

```
Input: Graph G, node set S, threshold τ, walk length ℓ.
Output: Set of nodes visited by delayed BFS.
for u in G.nodes() do
    if u ∈ S then
        T[u] ← 0
    else
        T[u] ← min(G.degree(u), τ)
    end if
end for
initialize queue I
enqueue(I, S)
while len(I) > 0 and len(S) < ℓ do
    i ← dequeue(I)
    S.insert(i)
    for u in i.neighbors() do
        if u ∉ I then
            decrement T[u]
            if T[u] = 0 then
                enqueue(I, u)
            end if
        end if
    end for
end while
return S
```

D Experimental Details

The transformer architecture in our CR-Graphormers comprises a single attention layer with 8 heads, one hidden layer of size 512, and a dropout rate of 10%. To assess the performance of our models, we compare them against several recent state-of-the-art methods, including VCR-Graphormer [Fu *et al.*, 2024] and NAGphormer [Chen *et al.*, 2023], both of which have demonstrated strong performance over existing graph transformer architectures. We also evaluate against PPRGO [Bojchevski *et al.*, 2020], an information diffusion-based GNN method, and GRAND+ [Feng *et al.*, 2022], which precomputes a diffusion matrix using a variant of personalized PageRank. Additional baselines include SAN [Kreuzer *et al.*, 2021], GraphGPS [Rampášek *et al.*, 2022], and Exphormer [Shirzad *et al.*, 2023]. Every baseline is trained and evaluated on the identical train/validation/test splits as our models, operating on the original input graphs with parameter settings matched to those used by our models. All training is performed using the Adam optimizer with a peak learning rate of 0.01 and a weight decay of 1×10^{-5}, minimizing the negative log-likelihood loss. We employ a linear decay learning rate scheduler with 500 warm-up steps and decay from 0.01 to 0.0001 over 1000 total training steps. Early stopping is applied with a patience of 50 epochs. Each model is trained for up to 2000 epochs with a batch size of 2000.

All the experiments were conducted on a workstation equipped with a single AMD Ryzen Threadripper PRO 5955WX processor (16 cores, 4.00–4.50 GHz, 64 MB cache, PCIe 4.0), one NVIDIA GeForce RTX 4090 GPU, and 128 GB of DDR4-3200.

Metric	Actor	Chameleon	Community	Computer	Cornell	Cycle	Grid
#Nodes	7600	2277	1400	13752	183	871	1231
#Edges	29707	31421	3871	245861	280	970	1705
Avg. Degree (Undirected)	7.818	27.599	5.530	35.756	3.060	2.227	2.770
#Features	932	2325	10	767	1703	1	1
#Classes	5	5	8	10	5	2	2

Metric	Photo	Pubmed	Shape	Squirrel	Texas	Wiki	Wisconsin
#Nodes	7650	19717	700	5201	183	11701	251
#Edges	119082	44327	1760	198493	295	216123	466
Avg. Degree (Undirected)	31.133	4.496	5.029	76.329	3.224	36.941	3.713
#Features	745	500	1	2089	1703	300	1703
#Classes	8	3	4	5	5	10	5

Table 3: Dataset Statistics

Ablation Study on CR-Absolute (TAS) Models. We analyze the sensitivity of CR-ABSOLUTE models through three experiments: one with varying maximum auxiliary graph degree, one with varying walk lengths, and one with varying absolute thresholds. In the experiment varying the maximum auxiliary graph degree, we fix the walk length ℓ at 10 and the threshold τ at 5. For the experiment with varying walk lengths, we set the maximum auxiliary graph degree k to the average degree and fix the threshold τ at 5. In the experiment with varying thresholds τ, we fix the maximum auxiliary graph degree k to the average degree and the walk length ℓ to 10. Across all trials, the number of starting neighbors $|R|$ is set to $\ell/2$. Using the same model configurations and training procedure detailed in Section 4, we evaluate test accuracy over 20 different data splits and report the mean in Figure 2. For the CR-ABSOLUTE method, tuning the model parameters to increase the neighbor counts $|R|$ and k consistently boosted accuracy on both *Cycle* and *Shape* by up to 15%.

Stability of Cascade-Rewired Auxiliary Graphs Against Supernode Perturbations. Using the same model configurations and training setup described in Section 4, with the neighborhood subset size $|R|$ set to the average degree, walk length $\ell = 2|R|$, and the maximum auxiliary graph degree k set to the average degree, we partition the cascade-rewired auxiliary graph and incorporate supernodes and structure-based virtual connections (as done in VCR-Graphormer [Fu *et al.*, 2024]) into our CR-Adaptive and CR-Absolute models. As shown in Figure 3, unlike VCR, our auxiliary graph structure enhances model stability against perturbations related to supernodes. These results demonstrate that the auxiliary graph *preserves* the local topology of the original network more effectively than other models, without necessitating explicit graph partitioning to reveal the network's structural backbone.

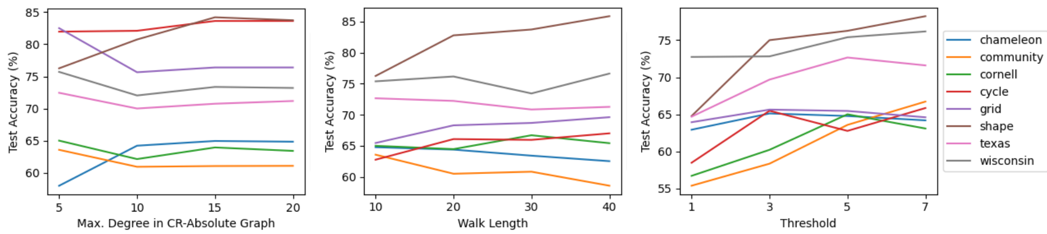

Figure 2: Ablation study on the cascade-rewired (CR) auxiliary graph with absolute threshold.

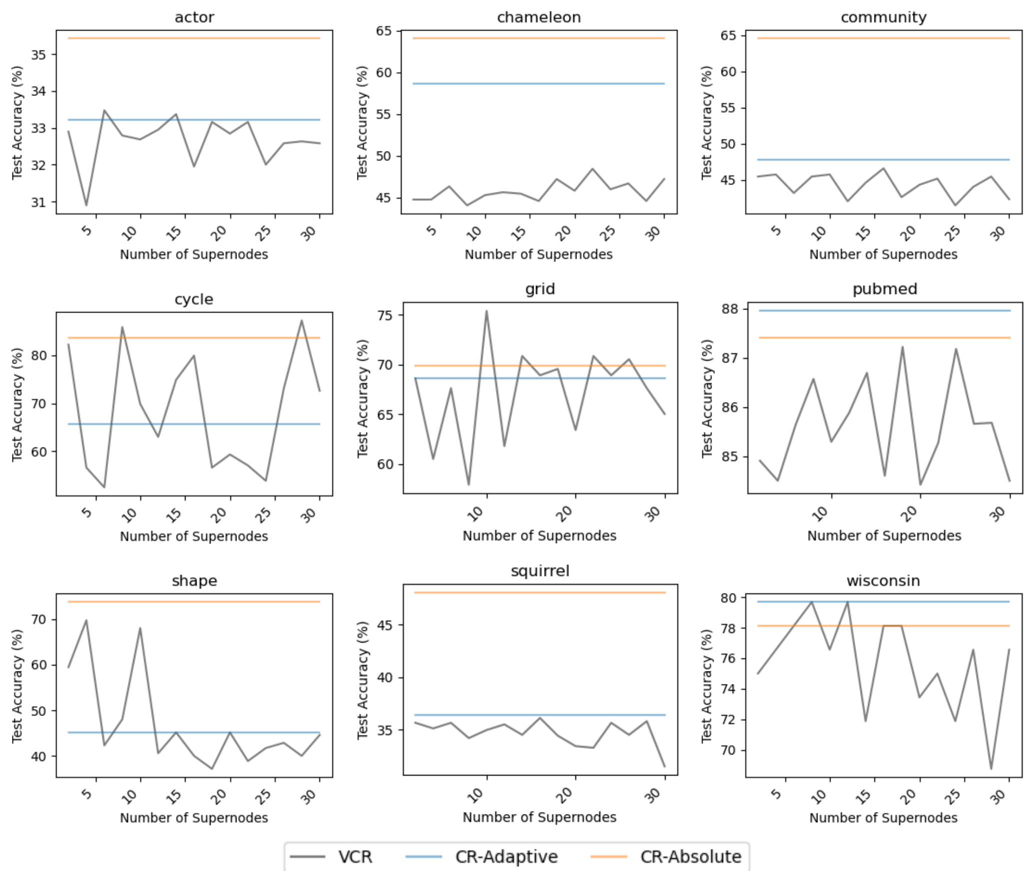

Figure 3: Performance comparison of VCR with our CR-Adaptive and CR-Absolute models across different numbers of supernodes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Sections 3 and 4 provide details of the claims made in abstract and introduction

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We mentioned in the Section 4 of the paper that our work in particular, attains considerable performance gains on heterophilic networks compared with homophilic networks. This can be considered as a limitation of our model.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All the assumptions are provided at the beginning of Theorem Statement (Theorem 3.1). The proof is provided in Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the details are provided in Section 4 and in Appendix for reproducibility of experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provided the source code in an anonymized Github repository. The URL is mentioned in the paper. We use the publicly available datasets whose URL is also mentioned in Section 4.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section 4 and Appendix for the training and test details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide mean and standard deviation for the node classification experiments conducted. Please refer to Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details are provided towards the end of Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is a classical Graph Machine Learning paper where we are showing the improvement in the accuracy pertinent to node classification task. Many societal applications could be indirectly benefited by them.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: The paper poses no such risks

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Citations are provided in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We released the code for review in an anonymous github repository: https://anonymous.4open.science/r/CR-graphormer-4DC0/

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer:[NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs were used solely for performing grammar correction.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.