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

Meher C. Pindiprolu[*]
meher.pindiprolu@alumni.ethz.ch

My Le[*]
Johns Hopkins University
mle19@jhu.edu

Luana Ruiz
Johns Hopkins University
lrubini1@jh.edu

[*]Equal contribution

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 the fine-grained structural signals, especially in heterophilous graphs, are lost. We propose **Cascade-Rewired Graph Transformer** (CR-GRAPHORMER), which balances computational efficiency with rich structural awareness. The approach unfolds in two stages. First, we assign each node a token based on "mesoscopic edge rewiring" by releasing deterministic contagion cascades from that node's ego-network. Replacing multi-hop paths with direct edges yields a backbone that captures higher-order structures while retaining sparsity. Next, on top of the rewired backbone, we insert feature-driven virtual edges that connect distant, attribute-similar nodes, thereby encoding long-range interactions and heterophilous relationships into each node's token list. The rewiring amplifies homophilous ties and preserves critical heterophilous connections, furnishing every vertex with a compact, information-rich context that reflects local motifs, global reach, and feature affinity. 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 takes $\mathcal{O}(E)$ complexity on graph tokenization producing an expressive yet scalable model. Across **14** homophilic and heterophilic benchmarks, CR-GRAPHORMER improves node-classification accuracy by up to **16.5%** over previous works, demonstrating that tokenization on the "mesoscopic rewired graph" provides a powerful inductive bias for graph learning.

1 Introduction

Graph-structured data arises 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 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.

Preprint. Under review.

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 tradeoffs: 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 5.1). 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 Present Work

We develop two complementary tokenization schemes for graph transformers, implemented in a concise two-stage pipeline.

Stage 1 – Mesoscopic rewiring via deterministic cascades. Let $G = (V, E)$ be an undirected graph. For every vertex u and a subset $R = N(u)$ of its ego-neighborhood, we launch fixed-length *deterministic contagion cascades* - memory-less higher-order random walks - under two activation rules, each of them resulting in a tokenization scheme:

1. activating a vertex once at least τ of its neighbors are active (given an *absolute threshold* τ);
2. activating the inactive vertex with maximum active-neighbor support (i.e., the threshold is *adaptive*).

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].

Stage 2 – Feature edge augmentation. We enhance the rewired structural auxiliary graph, G^*, by introducing *virtual edges*. This step is performed on the training data. Virtual edges connect non-adjacent vertex pairs exhibiting strong feature or contextual affinity, injecting additional contextual information into the graph. We repeat the same procedure from Stage 1 on this augmented graph to generate the rewired feature-augmented auxiliary graph $\tilde{G}$.

The added virtual edges preserve essential heterophilous relationships, allowing the contextually rewired graphs, G^* and $\tilde{G}$, to capture higher-order motifs (such as triangles, cliques, etc.), and also improve label homophily. This augmentation 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.

3 Related Work

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 kernelised 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 personalised 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, but suffers from locality bias, i.e., the probability mass is concentrated at hub nodes.

4 Preliminaries

This section offers 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 $n = |V|$ and $m = |E|$ and weight function $W : E \to \mathbb{R}$ assigning weights to the edges in E. For $i \in V$ denote

$$N(i) = \{j \in V : (i,j) \in E\}, \quad N[i] = N(i) \cup \{i\}, \quad d(i) = |N(i)|.$$

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 node i is activated.

The monotone *active set* for a vertex v at time T, denoted S_t^v, is defined as

$$S_t^v \; = \; \big\{ \, v \in V \mid [\mathbf{x}_t]_i = 1, 0 \le t \le T \, \big\}. \tag{1}$$

For an inactive vertex $u \notin S_t^v$ its instantaneous *neighbor support* is

$$\sigma_t(u) \; = \; |N(u) \cap S_t^v|. \tag{2}$$

Throughout we fix a *walk length* $\ell \in \mathbb{N}$ and run each cascade for ℓ activation steps. We reinterpret the two adjacency-search rules of Chaitanya *et al.* [2025] as discrete-time contagion dynamics:

Adaptive-Threshold Contagion (MAS) MAS models a diffusion whose activation requirement *adapts* to the most strongly reinforced inactive vertex. Let $v \in V$ be a designated seed vertex and draw a random subset $R \subseteq N(v)$ from its immediate neighbors. The cascade is initialized by activating $S_0^v \; = \; \{v\} \cup R$. To generate a ℓ length cascade, for every time step $t \in [0, \ell - 1]$, consider the set of inactive vertices $\{V \setminus S_t^v\}$. The instantaneous *neighbor support* of any such vertex $u \notin S_t^v$ is $\sigma_t(u) = |N(u) \cap S_t^v|$. We define the *adaptive* activation threshold for the contagion process originating at $\{v\} \cup R$ as

$$\tau_t \; = \; \max_{u \in V \setminus S_t^v} \sigma_t(u),$$

and select a vertex $u_t^\star \in \arg\max_{u \in V \setminus S_t^v} \sigma_t(u)$ to activate. The active set of v is updated by $S_{t+1}^v \; = \; S_t^v \cup \{u_t^\star\}$.

Since $\tau_{t+1} \le \tau_t \; \forall \, t$, 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) Let $v \in V$ be the seed vertex and select a random subset $R \subseteq N(v)$ of its first-order neighbors. The contagion is initialized by activating $S_0^v = \{v\} \cup R$. Fix a uniform reinforcement threshold $\tau \ge 1$, across all vertices in G. At each subsequent step $t \in [0, \ell - 1]$ every inactive vertex $u \notin S_t$ is assigned its instantaneous support $\sigma_t(u) = |N(u) \cap S_t^v|$.

Let

$$\mathcal{A}_t \; = \; \big\{ \, u \in V \setminus S_t^v \; \big| \; \sigma_t(u) \ge \tau \big\}.$$

If $\mathcal{A}_t \neq \varnothing$ we select one $u_t^\star \in \mathcal{A}_t$ (e.g. uniformly at random) and set $S_{t+1}^v = S_t \cup \{u_t^\star\}$.

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, memory-less random walk* whose state depends only on the current activated frontier [1].

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

$$\mathbf{H}' = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d'}}\Big)\mathbf{V} \in \mathbb{R}^{d'}, \tag{3}$$

where the queries $\mathbf{Q} = \mathbf{H}\mathbf{W}_Q$, the keys $\mathbf{K} = \mathbf{H}\mathbf{W}_K$, and 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 $\mathbf{H}' \in \mathbb{R}^{n\times d'}$.

5 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 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.

5.1 Mesoscopic Edge Rewiring

To capture homophily, we perform an adaptive and absolute threshold-activation simulation on G. For each node $v \in V$ we generate cascades of a constant length ℓ under two regimes:

- **Adaptive-Threshold Contagion (MAS)** For each node $v \in V$, together with a random subset of its neighborhood, $R \subseteq N_G(v)$, at step t compute $\tau_t = \max_{u \notin S_t^v} \sigma_t(u)$ and activate an arbitrary $u_t^\star \in \arg\max_{u \notin S_t^v} \sigma_t(u)$ by adding it to the active set. This process is repeated until ℓ vertices are added to the active set. This process is repeated for each node in G by choosing different random neighborhood subsets R. In our experiments, we name this procedure CR-ADAPTIVE.
- **Absolute-Threshold Contagion (TAS).** For a threshold τ, initialize $S_0^{(v)} = \{v\} \cup R$ and iteratively add any inactive $u \notin S_t^{(v)}$ satisfying $\sigma_t(u) \geq \tau$ until ℓ activations have occurred. This process is repeated for each node in G by choosing different random neighborhood subsets R and sweeping through the absolute thresholds $\tau \in \{1, \ldots, 5\}$. At each step, a node fires/activates once at least τ of its neighbors are in the active set. In our experiments, we name this procedure CR-ABSOLUTE.

Starting from the target node **v** together with different (random) subset of its immediate neighborhoods, 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 CR-ADAPTIVE and CR-ABSOLUTE procedures. These sequences obtained reveal which vertices repeatedly influence one another through multi-hop paths. Below, we describe the construction of the auxiliary graphs using 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 $S_\ell^v = (v, u_1, \ldots, u_\ell)$. Let $\mathcal{M}_v$ be the multiset of vertices appearing in the union of all active sets rooted at v. Define the frequency map

$$f_v(u) = \text{multiplicity of } u \text{ in } \mathcal{M}_v, u \in V.$$

[1]A higher-order random walk, unlike a first-order Markov process, carries information from preceding steps. Although our methods are not explicitly memory-based, their node-selection rule - picking nodes via maximum adjacency or a threshold - implicitly encodes earlier choices. Consequently, it behaves like a higher-order walk in which past decisions steer the current move.

Retain the "k" most frequent vertices for each $v \in V$

$$F_k(v) = \underset{u \in V}{\arg \text{top-k}} \, \{f_v(u)\}, \tag{4}$$

and construct an auxiliary activation graph with edges $E^v_{\text{act}} = \{(v, u) \mid u \in F_k(v), \forall v \in V\}$. This process is performed for all $v \in V$. The *cascade-rewired structural auxiliary graph* is then given by $G^* = (V^*, E^*, W^*)$, where $E^* = \bigcup_{v \in V} E^v_{\text{act}}$ 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 could 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.

The algorithms for Adaptive (MAS) and Absolute (TAS) Threshold Contagions, along with their corresponding asymptotic time complexity $\mathcal{O}(E)$, are presented in Appendix A. These algorithms are principally adapted from Chaitanya *et al.* [2025], and their linear dependence on the number of edges ensures computational efficiency.

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^v_0 , 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 5.1. *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^v_t the set of nodes activated until time t when the contagion starts from v together with a subset of its neighborhood R. Let $G^{'}$ be the subgraph in G that is activated by S^v_0.*

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

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

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

Then the effective resistance between the active set S^v_0 and u in $G^{'}$, denoted $R^{G^{'}}_{\text{eff}}(S_0, u)$, satisfies

$$R^{G^{'}}_{\text{eff}}(S_0, v) \leq \frac{\ell}{\tau}.$$

The proof is a direct consequence of the following two lemmas.

Lemma 5.2. *If u activates at threshold τ in at most ℓ time steps, then $G^{'}$ contains τ edge-disjoint S_0–v paths of length $\leq \ell$.*

Lemma 5.3. *If $G^{'}$ contains τ edge–disjoint S_0–v paths $P_1, \dots, P_\tau$ with $|P_i| \leq \ell$, then $R^{G^{'}}_{\text{eff}}(S_0, v) \leq \ell/\tau$.*

If u activates within ℓ steps, Lemma 5.2 gives the τ disjoint paths, and Lemma 5.3 yields $R^{G^{'}}_{\text{eff}}(S_0, v) \leq \ell/\tau$.

Remark. If $G^{'}$ consists solely of τ length-ℓ chains in parallel between S_0 and u, equality $R_{\text{eff}} = \ell/k$ is attained, showing the bound is tight.

The proofs of the Theorem and Lemmas are provided in the Appendix B.

Theorem 5.1 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, can inspect no more than the ℓ - hop frontier before selecting its top-k targets, thereby reducing the number of auxiliary edges added to the graph.

5.2 Feature Edge Augmentation

Here, we further enhance the cascade-rewired structural auxiliary graph G^* by incorporating context-aware virtual edges into G^*, using class labels in the *training data set*. We assume that each vertex $\mathbf{v} \in V^*$ in the *training dataset* is equipped with a d-dimensional feature vector $\{\mathbf{x}_v\}_{v\in V} \in \mathbb{R}^d$. We classify the nodes based on the labels when provided, or we use k-means, with a hyper-parameter c representing the number of feature clusters, and obtain one cluster per class label, yielding L disjoint clusters $\mathcal{C} = \{C_1, \ldots, C_L\}$ from G^*. For every cluster C_l, we add a virtual *feature-node* f_l and insert undirected edges that link f_l to all members of its cluster. Explicitly, we construct an augmented graph $G' = (V', E')$ with

$$V' = V^* \cup \{s_1, \ldots, s_L\}, \qquad E' = E^* \cup \big\{(f_\ell, v) \bigm| v \in C_l\big\}.$$

The augmented graph $G' = (V', E')$ serves as a feature-aware structural backbone that enhances homophilic connectivity while preserving the original set of nodes. Building on this augmented structure G', we construct the cascade-rewired feature graph $\tilde{G}$ by applying the procedure outlined in Section 5, wherein each node retains its top-k most frequently co-activated vertices. This rewiring process, while analogous to mesoscopic edge rewiring, is applied over the augmented graph G' to obtain $\tilde{G}$, thereby integrating both topological and feature-based influences.

5.3 Token-list formation

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

$$\boxed{\mathbf{T_v} = \{\underbrace{\mathbf{x}_v \| 1}_{\text{self}}, \quad \underbrace{\{(\mathbf{F}^*\mathbf{X})(\mathbf{v}, :) \| \omega_v^*\}}_{\text{Frequency Aggregation of } v's \text{ neighborhood in } G^*} \quad \underbrace{\{(\tilde{\mathbf{F}}\mathbf{X})(v, :) \| \tilde{\omega}_v\}}_{\text{Frequency Aggregation of } v's \text{ neighborhood in } \tilde{G}} \quad \}} \tag{5}$$

Where "$\|$" denotes concatenation, $\mathbf{F}^*$ and $\tilde{\mathbf{F}}$ denotes Frequency matrices for G^* and $\tilde{G}$ respectively and are calculated as follows: we identify the number of times a target node v with various (random) subsets of its neighborhoods activates another node u, and denote this as a frequency represented by $\dot{\mathbf{f}}_{vu}$. We then compute $\mathbf{f}_{vu}$, that is, an element in $\mathbf{F}^*$ and $\tilde{\mathbf{F}}$, using three different strategies (the method for calculating each entry in both frequency matrices is identical):

- Identity mapping: $\mathbf{f}_{vu} = \dot{\mathbf{f}}_{vu}$
- Global normalization: $\mathbf{f}_{vu} = \frac{\dot{\mathbf{f}}_{vu}}{\max_{s,t} \dot{\mathbf{f}}_{st}}$, where s and t are any two nodes in G^* or $\tilde{G}$.
- Source normalization: $\mathbf{f}_{vu} = \frac{\dot{\mathbf{f}}_{vu}}{\max_t \dot{\mathbf{f}}_{vt}}$, where $t \in N(v)$.

Once $\mathbf{f}_{vu}$ is computed for all existing pairs (v, u) in G^* and $\tilde{G}$, we symmetrize the matrices $\mathbf{F}^*$ and $\tilde{\mathbf{F}}$ by assigning $\mathbf{f}_{vu} = \mathbf{f}_{uv} = \frac{1}{2}(\mathbf{f}_{vu} + \mathbf{f}_{uv})$. Finally, ω_v^* is $\mathbf{F}^*(: v)$ and $\tilde{\omega_v}$ is $\tilde{\mathbf{F}}(: v)$. Each of the three aforementioned frequency computing strategies offers a distinct perspective on how to embed the raw frequencies $\dot{\mathbf{f}}_{vu}$ in relation to the target node v and its connectivity within the graph. We use *cross-validation* to select the strategy that best suits the input data.

5.4 Transformer encoder

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

$$\boxed{\tilde{\mathbf{Z}}_v^{(t)} = \mathrm{MHA}\big(\mathrm{LN}(\mathbf{Z}_{\mathbf{v}}^{(t-1)})\big) + \mathbf{Z}_v^{(t-1)}; \quad \mathbf{Z}_v^{(t)} \quad = \mathrm{FFN}\big(\mathrm{LN}(\tilde{\mathbf{Z}}_v^{(t)})\big) + \tilde{\mathbf{Z}}_v^{(t)},} \tag{6}$$

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 $1 + k + k$, independent of the original graph size, where k is the degree of the mesoscopic auxiliary graphs, G^* and $\tilde{G}$.

6 Experiments

Here we briefly introduce the datasets used in the experiments. The implementation of our approach can be accessed at https://anonymous.4open.science/r/CR-graphormer-4DC0/.

Datasets. We evaluate our models on 14 publicly available graph datasets, which serve as standard benchmarks for node classification across both homophilic and heterophilic network settings. All datasets are accessible through the Deep Graph Library (DGL[2]). Detailed dataset statistics are provided in the Appendix C.

Experimental Setup. We evaluate the node classification performance of our proposed models, CR-ADAPTIVE and CR-ABSOLUTE, on 14 benchmark networks from the DGL library. The results are provided in Table 1. For the CR-ADAPTIVE (MAS) and CR-ABSOLUTE (TAS) procedures, we fix the parameters as follows: neighborhood subset size $|R| = 5$, cascade walk length $\ell = 10$, and the k in equation (4) 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. The frequency weights (Section 5.3) are tuned through cross-validation; the setting that yields the highest validation performance is then fixed and applied to the held-out test set. We run experiments using 20 random data splits per graph to ensure the robustness of the models. Further details of the experimental scheme are provided in Appendix C.

To evaluate the performance of our models, we compare them against two recent state-of-the-art methods: VCR-Graphormer (VCR) [Fu *et al.*, 2024] and NAGphormer (NAG) [Chen *et al.*, 2023], both of which have demonstrated superior performance over existing graph transformer architectures such as Graphormer [Ying *et al.*, 2021], SAN [Kreuzer *et al.*, 2021], GraphGPS [Rampášek *et al.*, 2022], Vanilla GT [Dwivedi and Bresson, 2020], Gophormer [Zhao *et al.*, 2021], and Exphormer [Shirzad *et al.*, 2023]. We also include comparisons with classic graph neural network baselines, specifically the Graph Convolutional Network (GCN) [Kipf and Welling, 2016] and Graph Attention Network (GAT) [Veličković *et al.*, 2017]. 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. See Appendix C for further details on the model designs.

Method	Actor	Chameleon	Community	Computer	Cornell	Cycle	Grid
GCN	27.76 ± 1.06	61.60 ± 2.75	62.21 ± 2.14	90.94 ± 0.47	36.06 ± 4.95	59.27 ± 2.73	64.90 ± 7.68
GAT	27.89 ± 1.01	58.32 ± 4.22	65.20 ± 2.85	OOM	43.19 ± 7.44	59.27 ± 2.73	58.19 ± 1.76
NAG	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
VCR	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

Table 1 demonstrates that CR-GRAPHORMER attains competitive performance on most of the networks except for Cycle and Grid. The improvements are especially pronounced compared to the state-of-the-art Tokenized Graph Transformers on the challenging benchmarks: *Chameleon* (+18.8%),

[2]https://www.dgl.ai/dgl_docs/api/python/dgl.data.html

Method	Photo	Pubmed	Shape	Squirrel	Texas	Wiki	Wisconsin
GCN	94.16 ± 0.71	87.09 ± 0.35	48.46 ± 5.41	43.48 ± 1.32	51.06 ± 11.32	83.54 ± 0.79	45.08 ± 6.15
GAT	94.00 ± 0.66	86.40 ± 0.44	43.11 ± 2.97	39.59 ± 2.54	60.74 ± 7.98	OOM	49.53 ± 6.78
NAG	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
VCR	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 produced due to an out-of-memory error.

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 than in homophilic graphs, as shown in Table 2.

Ablation Study and Parameter Analysis. We analyze the performance sensitivity of the proposed CR-ADAPTIVE mechanism. Specifically, we assess how variations in the key hyperparameters - namely, the walk length ℓ and the maximum degree in the auxiliary networks (the top-k nodes selected for construction from (4)) that can affect node classification accuracy. The impact of these variations on test accuracy is summarized in Figure 1.

For CR-ADAPTIVE (MAS), we vary the maximum degree k (4) 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 $|R| = \ell/2$. This allows us to investigate how deeper mesoscopic exploration influences classification performance and whether increasing the walk length introduces meaningful performance gains. For most graphs, we observed that the accuracy gradually decreases with increased walk length, as supported by Theorem 5.1. Parameter analysis for CR-ABSOLUTE (TAS) is provided in Appendix D.

	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	0.334	0.406	0.783	0.302	0.682	0.820
Absolute Contagion Rewired Graph	0.216	0.313	0.444	0.770	0.242	0.907	0.910

	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	0.835	0.763	0.390	0.268	0.335	0.641	0.333
Absolute Contagion Rewired Graph	0.821	0.770	0.453	0.260	0.301	0.603	0.302

Table 2: Homophily scores across datasets.

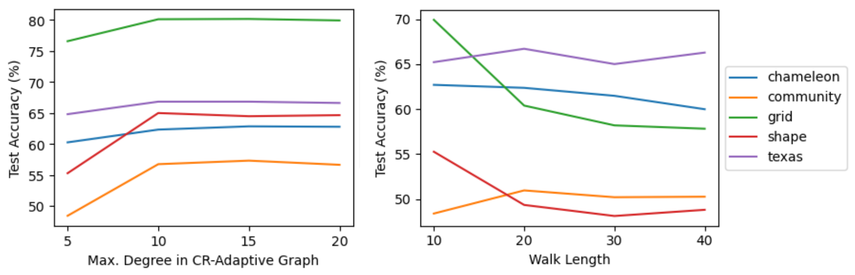

Figure 1: Ablation study on the cascade rewired (CR) auxiliary graph with adaptive threshold.

All the above 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.

7 Conclusion

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. Across 14 benchmarks, CR-GRAPHORMER improves node classification accuracy, by up to 16.5%, especially on heterophilic graphs, demonstrating the utility of cascade-based mesoscopic rewiring for structural encoding in Tokenized Graph Transformers.

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.

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).

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.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

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.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*, 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.

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

A 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 (MAS) and CR-ABSOLUTE (TAS) procedures described in Section 5. 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 (MAS) rely on the following parameters:

- **Walk Length (ℓ)**: This parameter refers to the sequence length that is generated for every vertex in the graph. In our experiments we set $\ell = 10$.
- **Number of Neighbors (k')**: $k' = |R|$ where $R \in N(v)$ (Section 5.1). In our experiments $|R| = 5$.
- **Number of Permutations (N)**: This parameter assists in capturing diversified immediate neighborhoods of a vertex v. Typically in our experiments we set $N = 8$.

Algorithm 1 Adaptive Threshold Contagions (MAS)

```
1: Input: Graph $G$, walk length $\ell$, #neighbors $k'$, #permutations $N$.
2: 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.
3: for $v$ in $G.nodes()$ do
4:     $neighbors \leftarrow v.neighbors()$
5:     $counter \leftarrow \{u : 0 \mid u \in G.nodes()\}$
6:     if len($neighbors$) $> 0$ then
7:         for $n = 1, \ldots, N$ do
8:             $neighbors \leftarrow$ permute($neighbors$)
9:             for $i = 0$ to len($neighbors$) with step $k'$ do
10:                $s \leftarrow \min(i, \max(0, \text{len}(neighbors) - k'))$
11:                $S \leftarrow [v] + neighbors[s : s + k']$
12:                while len($S$) $< \ell$ do
13:                    find the maximally adjacent node $u$ to $S$
14:                    $S$.insert($u$)
15:                end while
16:                for $u$ in $S$ do
17:                    increment $counter[u]$
18:                end for
19:            end for
20:        end for
21:    end if
22:    sort $counter$ by its values in descending order
23:    $D[v] \leftarrow counter$
24: end for
25: return $D$
```

Absolute Threshold Contagions (TAS) rely on the following parameters:

- **Threshold (τ)**: $\tau \in [1, 5]$.
- **Walk Length (ℓ)**: $\ell = 10$.
- **Number of Neighbors (k')**: $k' = |R|$.
- **Number of Permutations (N)**: $N = 8$.

Algorithm 2 Absolute Threshold Contagions (TAS)

```
1: Input: Graph $G$, threshold $\tau$, walk length $\ell$, #neighbors $k'$, #permutations $N$.
2: Output: Dictionary $D$ that stores the activation frequencies of all nodes for each starting node, ordered by
   activation frequency in descending order.
3: for $v$ in $G.nodes()$ do
4:     $neighbors \leftarrow v.neighbors()$
5:     $counter \leftarrow \{u : 0 \mid u \in G.nodes()\}$
6:     if len($neighbors$) $> 0$ then
7:         for $\tau' = 1, \ldots, \tau$ do
8:             for $n = 1, \ldots, N$ do
9:                 $neighbors \leftarrow$ permute($neighbors$)
10:                for $i = 0$ to len($neighbors$) with step $k'$ do
11:                    $s \leftarrow \min(i, \max(0, \text{len}(neighbors) - k'))$
12:                    $S \leftarrow [v] + neighbors[s : s + k']$
13:                    $S \leftarrow$ Delayed-BFS($G, S, \tau', \ell$)
14:                    for $u$ in $S$ do
15:                        increment $counter[u]$
16:                    end for
17:                end for
18:            end for
19:        end for
20:    end if
21:    sort $counter$ by its values in descending order
22:    $D[v] \leftarrow counter$
23: end for
24: return $D$
```

Algorithm 3 Delayed-BFS

```
1: Input: Graph $G$, node set $S$, threshold $\tau$, walk length $\ell$.
2: Output: Set of nodes visited by delayed BFS.
3: for $u$ in $G.nodes()$ do
4:     if $u \in S$ then
5:         $T[u] \leftarrow 0$
6:     else
7:         $T[u] \leftarrow \min(G.degree(u), \tau)$
8:     end if
9: end for
10: initialize queue $I$
11: enqueue($I, S$)
12: while len($I$) $> 0$ and len($S$) $< \ell$ do
13:     $i \leftarrow$ dequeue($I$)
14:     $S$.insert($i$)
15:     for $u$ in $i$.neighbors() do
16:         if $u \notin I$ then
17:             decrement $T[u]$
18:             if $T[u] = 0$ then
19:                 enqueue($I, u$)
20:             end if
21:         end if
22:     end for
23: end while
24: return $S$
```

A.1 Time Complexity

Our time-complexity analysis mirrors that of Chaitanya *et al.* [2025]. We keep a dictionary that records each node's activation frequency count and sort these counts, in descending order, for each of the corresponding source node. Because the active set at any vertex (source) is bounded by the fixed constants, the walk length ℓ, the neighbors k', and the number of permutations N - this sorting step takes constant time. Consequently, the overall time complexity is $\mathcal{O}(n + m)$ (since $\{l, k', \tau, N\} \in \mathcal{O}(1)$).

B Proofs of Theorem and Lemmas

Theorem B.1. *Let $G = (V, E)$ be an undirected, unit edge weight 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 $G^{'}$ be the subgraph in G that is activated by all the active sets, S_t^v, rooted at 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 $R_{\text{eff}}^{G^{'}}(S_0, u)$, satisfies

$$R_{\text{eff}}^{G^{'}}(S_0, v) \leq \frac{\ell}{\tau}.$$

The proof is the consequence of the following two lemmas.

Lemma B.2. *If u activates at threshold τ in at most ℓ time steps, then $G^{'}$ contains τ edge-disjoint S_0–v paths of length $\leq \ell$.*

Proof. We prove the Lemma by performing induction on step/round t. **Base** $t = 1$**.** If $t^\star = 1$ then u is adjacent to at least τ seeds in S_0; these τ edges form the required paths.

Induction step. 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 $w_1, \dots, w_k \in S_t^v$. 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. $\square$

Lemma B.3. *If $G^{'}$ contains τ edge–disjoint S_0–v paths $P_1, \dots, P_\tau$ with $|P_i| \leq \ell$, then $R_{\text{eff}}^{G^{'}}(S_0, v) \leq \ell/\tau$.*

Proof. 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 is of unit edge weights, i.e., weight of every edge is one, in H, each path P_i is a series of $\ell_i \leq \ell$ unit resistors, thus 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 $R_{\text{eff}}^{G}(S_0, v) \leq \ell/\tau$. $\square$

Proof of Theorem B.1

If v activates by round ℓ, Lemma B.2 gives the τ disjoint paths, and Lemma B.3 yields $R_{\text{eff}}^{G'}(S_0, v) \leq \ell/\tau$.

$\square$

C Experimental Details

The transformer architecture in our CR-Graphformers comprises a single attention layer with 10 heads, six hidden layers of size 512, and a dropout rate of 10%. GCN and GAT consist of a 6-layer architecture with hidden dimensions of 512, ReLU activation, and a dropout rate of 10% between layers. A softmax activation is applied at the output layer. The GAT models additionally employ 10 attention heads, matching the configuration used in our proposed CR-Graphormer.

All training is performed using the Adam optimizer with a peak learning rate of 0.01 and weight decay of 1×10^{-5}, minimizing the negative log-likelihood loss. We employ a linear decay learning

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

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.

D Ablation Study on CR-Absolute (TAS) Models

This ablation study comprises 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 fixed 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 $\frac{\ell}{2}$. Using the same model configurations and training procedure detailed in Section 6, 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 raise the neighbor counts $|R|$ and k consistently boosted accuracy on both "cycle" and "shape" graphs by up to 15%.

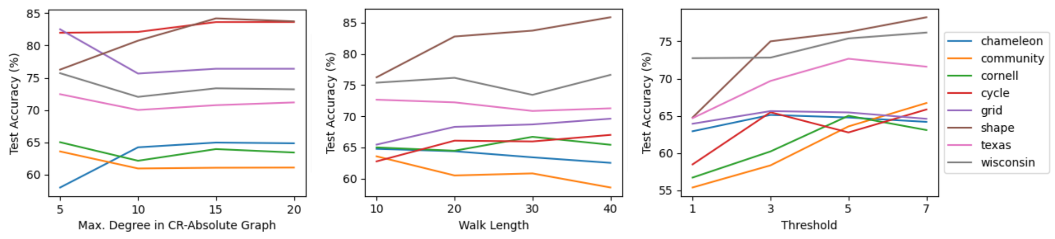

Figure 2: Ablation study on the casade rewired (CR) auxiliary graph with absolute threshold.

D.1 Stability of Cascade-Rewired Auxiliary Graphs Against Supernode Perturbations

Using the same model configurations and training setup described in Section 6, 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.

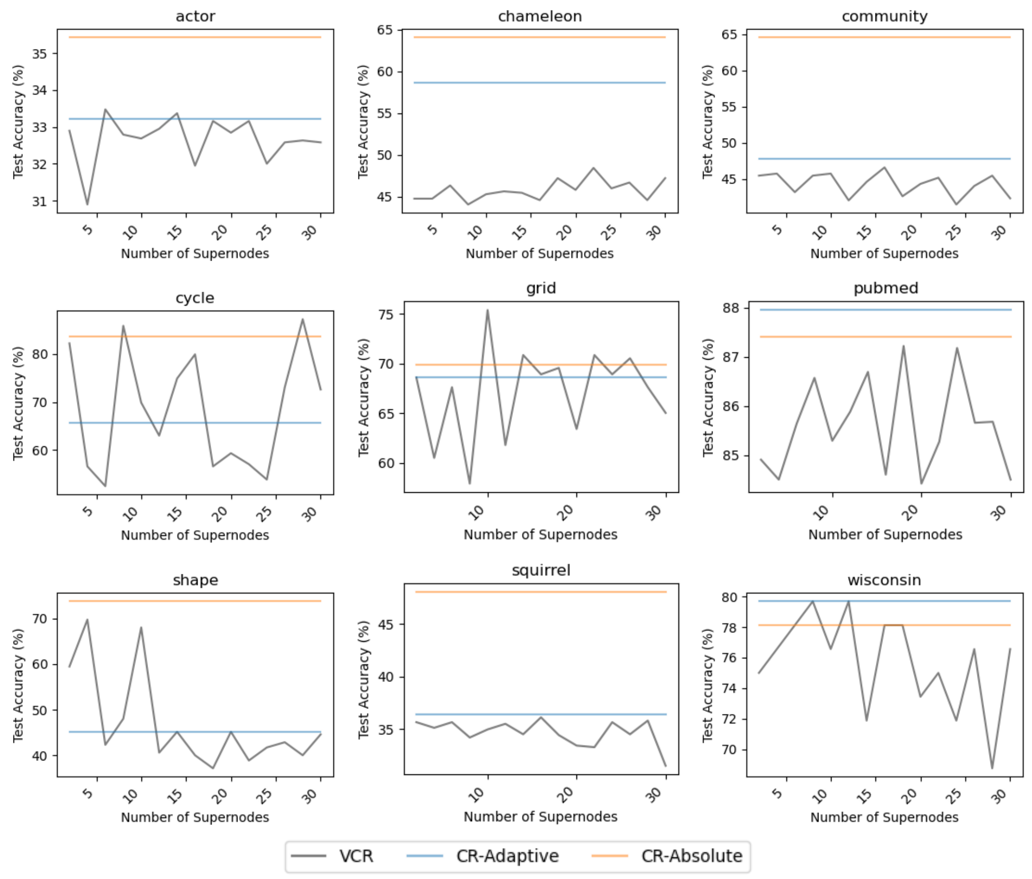

Figure 3: Performance comparison of VCR with our CR-Adaptive and CR-Absolute models across different numbers of supernodes.