BEYOND NODE TOKENS: MOTIFS AS STRUCTURAL CONTEXT FOR GRAPH TOKENIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph tokenization enables graph-level learning with discrete symbols drawn from a shared vocabulary. A common approach represents each graph as an unordered multiset of quantized node tokens. We analyze the capacity of this representation and empirically unveil the inter-graph token collapse issue, i.e., structurally different graphs can receive highly similar token representations. To remedy this problem, we propose `GMT`, a graph tokenization framework that uses recurring motifs as reusable structural context. `GMT` constructs a shared motif vocabulary and learns motif embeddings through soft node matching supervised by motif–graph occurrence, with collision regularization to discourage many-to-one assignments. These embeddings guide node representation learning before quantization, encouraging node tokens to retain both attribute and motif-level information. A Transformer then jointly processes node and motif tokens through token-type-aware attention and motif-conditioned reconstruction of masked node-token embeddings, followed by fine-tuning for graph-level tasks. Empirically, `GMT` achieves the best mean performance among evaluated methods on almost all classification and regression benchmarks, including a 17.1% reduction in *ZINC* mean absolute error over the strongest evaluated baseline. [1]

1 INTRODUCTION

The success of Transformers in sequence modeling has spurred growing interest in extending them to graphs (Ying et al., 2021; Kim et al., 2022). A key step is to convert the graph-structured data into token sequences that Transformers can process. A growing number of recent studies focus on tokenizing graph elements into discrete symbols to support node, link, and graph-level tasks (Yang et al., 2024; Wang et al., 2024; Guo et al., 2026). Among these efforts, *graph tokenization* aims to transform an entire graph into tokens drawn from a finite vocabulary shared across graphs, for downstream graph classification and regression. These tasks underpin various applications such as molecular property prediction (Wu et al., 2018), circuit performance estimation for logic synthesis (Chowdhury et al., 2021; Wu et al., 2022), and social network analysis (**?**), and many others (Gilmer et al., 2017; Li et al., 2025).

Several works pursue graph-level tokenization through *graph serialization*, using graph traversals (e.g., random walks, paths) to reversibly encode connectivity and node label as token sequences to uniquely represent graphs (Zhao et al., 2025; Guo et al., 2026; Wang et al., 2025a). Although these methods achieve competitive downstream performance, their explicit encodings can substantially expand the input. As reported in Appendix E.3, on *Dielectric* the token sequence of `GMT` is 17.80× shorter than the shortest BPE-compressed serialization. Since self-attention in Transformer models scales quadratically with sequence length, the long token sequences produced by graph serialization render both pretraining and downstream task training computationally expensive. Moreover, these serialization schemes cannot encode the attributes of nodes and edges. In contrast, a prominent alternative constructs graph tokens from *quantized node tokenization* (QNT). These methods first harness *graph neural networks* (GNNs) (Kipf & Welling, 2016) to encode node attributes and neighborhood structure into node representations, then map such embeddings into discrete codes from shared codebooks via *vector quantization* (VQ) (Xia et al., 2023; Yang et al., 2024; Wang et al., 2024; Luo et al., 2025). A graph-level representation is then formed by collecting the codes

[1] Our code is available at https://anonymous.4open.science/r/GMT-27

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

assigned to all its nodes into an unordered token multiset. With a fixed code budget per node, the number of tokens grows linearly with the number of nodes, offering compact representations for downstream learning. Despite this compactness, QNT methods considerably underperform graph serialization on benchmark datasets (e.g., *ZINC*), with MAEs of 0.394–0.572, versus 0.222-0.269 for `GraphGPT` and `GraphTokenizer` (Table 2).

Motivated by this performance gap, we examine the limitations of QNT through empirical diagnostics and theoretical analysis (§3.2), identifying two forms of *graph token collapse* (GTC). The message passing paradigm of GNNs can yield homogeneous representations for nodes within a graph, and VQ additionally erases their subtle differences by mapping them to shared codes, leading to *Intra GTC*. This further reduces graph-level discrimination, particularly when graphs are represented as unordered node-token multisets without explicit connectivity. Consequently, structurally different graphs may receive highly similar token-derived representations, a phenomenon we term *Inter GTC*. Together, these findings expose a limitation of using quantized node tokens solely: distinctions captured at the node level may not preserve those required to discriminate entire graphs, potentially impairing downstream prediction.

Inspired by these findings, our key idea is to explicitly encode distinctive subgraph patterns to better distinguish graphs with similar node tokens. We therefore propose `GMT`, which encodes recurring *motifs* into motif tokens as reusable structural context to guide node tokenization as well as augment graph representations. Particularly, our contributions can be summarized as follows:

- We introduce a carefully-crafted motif tokenizer that learns reusable embeddings from both template structure and motif occurrences across attributed graphs. It combines soft matching supervised by occurrence labels with collision regularization, discouraging many-to-one matches without requiring node-level correspondence supervision.
- We extend QNT with *motif supervision before quantization*, using learned motif token embeddings as structural targets shared across graphs. Joint graph–motif ranking and masked attribute reconstruction encourage node representations to capture motif context while retaining the attribute information needed to distinguish individual nodes.
- We introduce *motif-conditioned masked pretraining*, using visible motif tokens as structural context to reconstruct masked node tokens. We additionally design token-type-aware attention with learned type-pair biases to balance node and motif context during masked reconstruction.
- Experiments on 12 graph classification and regression benchmarks show that `GMT` achieves the best mean performance on eleven, including a 17.1% reduction in *ZINC* MAE over the strongest evaluated baseline. Ablations further demonstrate the benefits of learned motif representations and motif-conditioned pretraining.

2 Related Work

Graph tokenization has been used to describe both continuous graph embeddings (Tang et al., 2024) and discrete symbolic representations (Yang et al., 2024). Here, we focus on mapping graphs to token IDs drawn from a finite vocabulary shared across graphs. Within this scope, two closely related lines are *quantized node tokenization* (QNT) and *graph serialization*.

Quantized Node Tokenization. Node tokenization represents a graph as a collection of node-level tokens derived from attributes of nodes and their structural contexts. A common discrete approach is QNT, which maps GNN-encoded node representations to discrete codes from finite codebooks shared across graphs via vector quantization techniques, providing reusable node-level symbols for downstream graph learning. Single-code approaches include `Mole-BERT` (Xia et al., 2023), which learns contextual atom codes for molecular pretraining, and `VQGraph` (Yang et al., 2024), which transfers structural knowledge from GNNs to MLPs through code assignments for node classification. Multi-code approaches include `GQT` (Wang et al., 2024) and `NodeID` (Luo et al., 2025): both use residual vector quantization, with `GQT` learning hierarchical tokens through self-supervised objectives and `NodeID` combining codes across GNN layers. `HIGHT` (Chen et al., 2024) augments atom tokens with BRICS-derived motif tokens for molecule–language alignment, demonstrating the importance of preserving node–motif relations. However, this construction relies on BRICS providing motif membership for molecules and does not generalize to arbitrary graphs.

Graph Serialization. Graph serialization converts graph structure and attributes into ordered sequences for processing by sequence models. Its defining step is an explicit graph-to-sequence construction, which is not required by node-wise quantization. For explicit graph serialization, `GraphGPT` (Zhao et al., 2025) uses Eulerian traversals, whereas `G2PT` (Chen et al., 2025) repre-

sents graphs using node records followed by explicit edge records. `GraphTokenizer` (Guo et al., 2026) combines frequency-guided serialization with corpus-level *byte-pair encoding* (BPE) (Sennrich et al., 2015) to learn a shared vocabulary and compress the resulting sequences. Another group constructs representations from sampled walks: `GPM` (Wang et al., 2025a) encodes node attributes and repeated-node relationships through semantic and anonymous paths, while `G`^{2}`PM` (Wang et al., 2025b) learns walk-based substructure representations through masked pretraining. These walk-based methods use shared pattern encoders, and their sampled substructures can overlap. The above methods differ in structural coverage, ordering dependence, and sequence length; explicit connectivity records allow serialization to retain information beyond path adjacency.

GNNs and Graph Transformers. Message-passing GNNs (`GCN` (Kipf & Welling, 2016), `GIN` (Xu et al., 2018), `GAT` (Velickovic et al., 2017)) encode local neighborhoods and read out a graph vector. Transformers are designed for ordered token sequences (Vaswani et al., 2017). Graphs have no canonical order and are therefore non-sequential inputs. *Graph Transformers* inject structure into input representations and attention via positional/structural encodings (Rampášek et al., 2022; Ma et al., 2023; Shirzad et al., 2023), relative attention biases (Ying et al., 2021; Hussain et al., 2021; Jin et al., 2023), and structure-aware token identifiers (Kim et al., 2022; Liu et al., 2023; Chen et al., 2023). Both lines keep topology inside the architecture.

3 Preliminaries

3.1 Problem Formulation

Notations. We define $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \tau, \boldsymbol{X})$ as a *typed attributed graph*, where $\mathcal{V}$ denotes a set of $n = |\mathcal{V}|$ nodes and $\mathcal{E}$ represents a set of $e = |\mathcal{E}|$ edges. $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$ denotes the attribute matrix of $\mathcal{G}$, wherein $\tilde{\mathbf{x}}_v$ represents the attribute vector of node $v \in \mathcal{V}$. Each node v is mapped to a node type $\tau(v) \in \mathcal{T}$. Accordingly, we denote by $\mathcal{D} = (\mathcal{G}^{(i)}, y^{(i)})_{i=1}^{N}$ a labeled graph dataset, wherein each graph $\mathcal{G}^{(i)} = (\mathcal{V}^{(i)}, \mathcal{E}^{(i)}, \boldsymbol{X}^{(i)})$ is associated with a label vector $y^{(i)}$ that represents either categorical labels for classification or continuous values for regression.

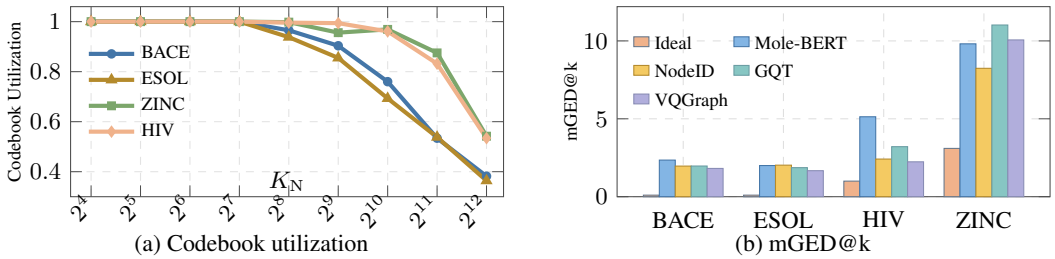
Structural Motifs. A graph $g = (\mathcal{V}_g, \mathcal{E}_g)$ is a *subgraph* of $\mathcal{G}$ ($g \subseteq \mathcal{G}$) if $\mathcal{V}_g \subseteq \mathcal{V}$ and $\mathcal{E}_g \subseteq \mathcal{E}$. A *structural motif* g_j is a connected subgraph pattern that occurs frequently in graph collection $\mathcal{D}$; we collect these patterns into a vocabulary $S = \{g_1, \ldots, g_{|S|}\}$. Particularly, such motifs can be discovered by *frequent subgraph mining* algorithms such as `gSpan` (Yan & Han, 2002). Finding a match between g_j and $\mathcal{G}$ is NP-complete when the motif size is part of the input; classical exact algorithms use backtracking and pruning, as in (Ullmann, 1976).

Vector Quantization. Vector quantization (VQ) maps continuous representations to a finite set of codewords (van den Oord et al., 2017). A shared codebook contains K learnable codeword vectors $\{\vec{\mathbf{u}}_i\}_{i=1}^{K} \subset \mathbb{R}^d$. Each embedding vector $\vec{\mathbf{h}}_v \in \mathbb{R}^d$ is assigned to its nearest codeword: $z_v = \arg\min_{i \in \{1,\ldots,K\}} \|\vec{\mathbf{h}}_v - \vec{\mathbf{u}}_i\|_2$, $\mathrm{VQ}(\vec{\mathbf{h}}_v) = \vec{\mathbf{u}}_{z_v}$. Here, z_v is the discrete token ID, whereas $\vec{\mathbf{u}}_{z_v}$ is its corresponding embedding. These assignments partition the representation space into *Voronoi regions*, within each of which all embeddings receive the same token ID. Over a given collection, a code is *active* if it is assigned to at least one object. We denote the number of active codes by $\widetilde{K} \leq K$ and define *codebook utilization* as $\widetilde{K}/K$.

Graph Tokenization. Graph tokenization aims to represents a graph using discrete symbols from a shared finite vocabulary $\mathcal{C} = \{c_1, \ldots, c_K\}$, wherein each code (i.e., token ID) is mapped into an embedding vector (i.e., token) via a mapping function $f : \mathcal{C} \to \mathbb{R}^d$. For the unordered tokenization considered here, the tokenizer T outputs a token multiset for each input graph $\mathcal{G}$: $T(\mathcal{G}) = (c_{i_1}, \ldots, c_{i_{p(\mathcal{G})}})$, $c_{i_\ell} \in \mathcal{C}$, where $p(G)$ denotes the number of tokens for $\mathcal{G}$. A permutation-invariant predictor then produces $\widehat{y} = \psi_\theta(\{\{f(c) : c \in T(\mathcal{G})\}\})$. The objective is to retain information useful for graph-level classification or regression on unseen graphs.

3.2 Intra- and Inter-Graph Token Collapse

QNT constructs graph tokens by quantizing GNN-based node representations (Xia et al., 2023; Yang et al., 2024; Wang et al., 2024; Luo et al., 2025). Let K_{N} denote the size of the shared node vocabulary, with codeword embeddings $\{\vec{\mathbf{u}}_i\}_{i=1}^{K_{\mathrm{N}}}$. For each node v, QNT computes $\vec{\mathbf{h}}_v = \mathrm{GNN}(\mathcal{G})_v$ and assigns it the nearest-codeword index z_v through VQ. The graph is then represented by the



(a) Codebook utilization (b) mGED@k

Figure 1: Token collapse analysis.

unordered token multiset $T_{\mathrm{N}}(\mathcal{G}) = \{\{z_v : v \in \mathcal{V}\}\}$. Its ability to distinguish graph structures therefore depends on the information retained through both message passing and quantization.

Intra-Graph Token Collapse. However, repeated neighborhood aggregation can suppress differences between node representations through *oversmoothing* and *feature correlation* (Oono & Suzuki, 2019; Jin et al., 2022). This is particularly relevant when a few propagation steps already cover much of a small graph, leading to substantially overlapping aggregation neighborhoods. VQ can further reduces distinctions between node representations: embeddings within the same Voronoi region are mapped to the same codeword. We refer to this loss of within-graph distinctions as *Intra-Graph Token Collapse* (Intra GTC) henceforth. Unfortunately, enlarging vocabulary size does not ensure that additional codes are used: in Fig. 1(a), codebook utilization is only 36.4%–54.2% at $K_{\mathrm{N}} = 4,096$, consistent with the codebook underutilization documented in prior VQ research (Takida et al., 2022).

Inter-Graph Token Collapse. An unordered token multiset $T_{\mathrm{N}}(\mathcal{G})$ retains code identities and multiplicities, but does not explicitly record how their occurrences are connected. By a counting argument, for an n-node graph, a vocabulary of size K_{N} admits at most $\mathcal{U}(n, K_{\mathrm{N}}) = \binom{n+K_{\mathrm{N}}-1}{n}$ distinct token multisets. If K_{N} is fixed or grows at most polynomially with n, then $\log \mathcal{U}(n, K_{\mathrm{N}}) = O(n \log n)$. In contrast, even simple graphs with identical node attributes have at least $2^{\binom{n}{2}}/n! = 2^{\Omega(n^2)}$ non-isomorphic configurations. Consequently, exact token-multiset collisions are unavoidable over unrestricted graph families for sufficiently large n (Appendix C.1). Notably, the intra GTC issue weakens graph-level discrimination, and thus, further exacerbates the token-multiset collisions. As an aftermath, structurally different graphs may share highly similar token representations, a phenomenon we describe as *Inter-Graph Token Collapse* (Inter GTC).

To examine structural distinguishability on real datasets, for each tokenizer, let $\mathcal{P}^{(k)}$ contain the k distinct unordered pairs of different graphs with the highest similarity between their graph representations derived from their respective tokens. The construction of $\mathcal{P}^{(k)}$ is given in Appendix D.5. We report their *mean GED@k*:

$$\mathrm{mGED}\,@k = \frac{1}{k} \sum_{(\mathcal{G}_i, \mathcal{G}_j) \in \mathcal{P}^{(k)}} \mathrm{nGED}(\mathcal{G}_i, \mathcal{G}_j), \tag{1}$$

where $\mathrm{nGED}(\mathcal{G}_i, \mathcal{G}_j)$ denotes the size-normalized *graph edit distance* (GED) (Sanfeliu & Fu, 1983) between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$. Intuitively, a larger mGED@k indicates that graphs considered highly similar in token space but differ substantially in structure. Fig. 1(b) shows consistently and conspicuously higher mGED@k for all four evaluated QNT methods across all four datasets, corroborating the presence of Inter GTC issue. On *ZINC*, mGED@k ranges from 8.24 to 11.02, compared with 3.1 for the ideal case where the k graph pairs are selected from the dataset by lowest GED.

The preceding analysis shows that quantized node-token multisets can fail to preserve structural differences between graphs. This motivates a natural solution that separately encodes structural elements or patterns and explicitly incorporates them for better discrimination of graphs. We therefore consider recurring *motifs* as reusable structural context: graphs with similar node tokens may contain different motifs that can that reflect their distinctive structural features, such as functional groups in molecular graphs. However, it remains unclear how to extract informative motifs, encode them as reusable tokens, and integrate them with node tokens to better preserve structural distinctions between graphs.

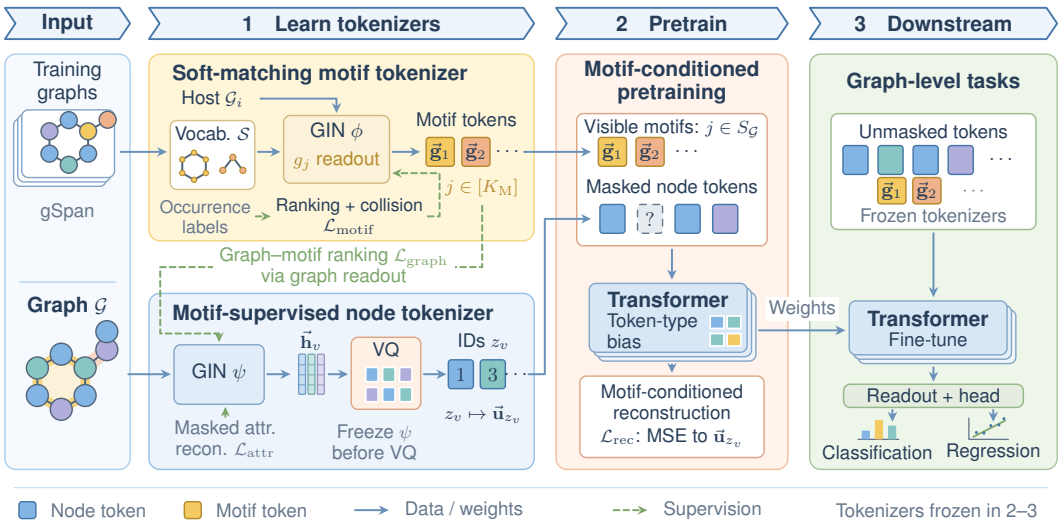


Figure 2: Framework Overview of GMT.

4 METHODOLOGY

Building on the above findings and inspirations, we propose GMT, which capitalizes on recurring motifs as shared structural context for graph tokenization. As illustrated in Figure 2, GMT first learns reusable motif embeddings through soft matching supervised by motif–graph occurrence labels (§4.1). These embeddings then guide node representation learning through graph–motif ranking, before the node representations are quantized into discrete tokens (§4.2). Finally, a Transformer with token-type-aware attention jointly processes node and motif tokens, learning to reconstruct masked node-token embeddings with visible motifs as context (§4.3). The pretrained Transformer is then fine-tuned for downstream graph-level tasks.

4.1 MOTIF TOKENIZATION VIA SOFT MATCHING

Motif Vocabulary Construction. We apply the well-known gSpan to the training graphs in $\mathcal{D}$ to obtain a vocabulary $\mathcal{S} = \{g_1, \ldots, g_{|K_M|}\}$ of K_M frequent connected subgraphs with the highest *support*. Unlike BRICS (Degen et al., 2008), gSpan does not rely on atom types or reaction rules, and hence, S applies to molecules, circuits, and other attributed graphs. Notice that vocabulary construction is performed offline, and the resulting patterns are reused throughout training.

Soft Matching with Collision Regularization. After extracting the motif vocabulary $\mathcal{S}$, we then learn a reusable token embedding $\vec{\mathbf{g}}_j$ for each motif $g_j \in \mathcal{S}$. Aside from encoding g_j's internal structural patterns, GMT additionally utilizes motif–graph occurrence labels to supervise matching between *template nodes* (without attributes but types) in motif g_j and attributed nodes in graph object $\mathcal{G}_i$. This encourages the encoder to learn which node-level structures support a motif's occurrence, providing reusable structural context for subsequent node tokenization and masked pretraining.

Specifically, we first encode each template node w in motif g_i and each node v in graph $\mathcal{G}_i$ via a shared *Graph Isomorphism Network* (GIN) (Xu et al., 2018) ϕ, i.e., $\vec{\mathbf{h}}_w = \phi(g_j)_w$ and $\vec{\mathbf{h}}_v = \phi(\mathcal{G}_i)_v$. As such, both motif g_j and graph $\mathcal{G}_i$ are then converted into a vector set $\{\vec{\mathbf{h}}_w\}_{w \in g_j}$ and $\{\vec{\mathbf{h}}_v\}_{v \in \mathcal{G}_i}$, respectively. To model the containment relation between g_j and $\mathcal{G}_i$, We compute a *soft matching cost* (SMC) between the motif and the host graph by averaging node-wise distances weighted by the soft assignment weights over template nodes in g_j:

$$\Delta(g_j, \mathcal{G}_i) = \frac{1}{|\mathcal{V}_{g_j}|} \sum_{w \in g_j} \sum_{v \in \mathcal{G}_i} B^j_{w,v} \cdot \left(1 - \cos(\vec{\mathbf{h}}_w, \vec{\mathbf{h}}_v)\right), \tag{2}$$

where soft assignment weight $B^j_{w,v}$ for node pair (w, v) is calculated by $\frac{\exp\left(\cos(\mathbf{h}_w, \mathbf{h}_v)/\tau\right)}{\sum_{v'} \exp\left(\cos(\vec{\mathbf{h}}_w, \vec{\mathbf{h}}'_v)/\tau\right)}$, which indicates how likely host-graph node v is to match template node w, relative to other nodes in $\mathcal{G}_i$. With SMC and a few motif-graph occurrence labels obtained by gSpan, we can formulate the ranking loss as follows: $\mathcal{L}_{\text{rank}} = \sum_{\mathcal{G}_i} \sum_{g_j^+} \sum_{g_j^-} \max\left(0, \gamma + \Delta(g_j^+, \mathcal{G}_i) - \Delta(g_j^-, \mathcal{G}_i)\right)$, which encourages positive motifs g_j^+ that occur in $\mathcal{G}_i$ to have SMC at least $\gamma > 0$ lower than those of

negative motifs g_j^- not in $\mathcal{G}_i$. Since subgraph isomorphism requires an injective mapping from template nodes to host nodes, GMT introduces a collision penalty to discourage distinct template nodes from assigning probability mass to the same host node:

$$\mathcal{L}_{\text{coll}} = \binom{|\mathcal{V}_{g_j}|}{2}^{-1} \sum_{w<w' \in g_j} \sum_{v \in \mathcal{G}_i} B^j_{wv} B^j_{w'v}. \tag{3}$$

Overall, the training loss for motif tokenization is then $\mathcal{L}_{\text{motif}} = \mathcal{L}_{\text{rank}} + \lambda_{\text{coll}}\mathcal{L}_{\text{coll}}$. Finally, we obtain the token embedding $\vec{\mathbf{g}}_j$ of motif g_j by applying a readout function to its learned template-node representations: $\vec{\mathbf{g}}_j = \text{READOUT}\big(\{\vec{\mathbf{h}}_w : w \in \mathcal{V}_{g_j}\}\big)$.

4.2 Motif-Supervised Node Tokenization

Next, we extend QNT by using the motif tokens learned in preceding section to supervise node representations before quantization, thereby encouraging the resulting node tokens to retain motif-level structural context that standard QNT may fail to preserve, and hence, improve the graph-level discriminative power of the resulting tokens.

Dual-Task Node Encoder. Unlike vanilla QNT that learns node representations by optimizing graph reconstruction or contrastive objectives (Xia et al., 2023; Yang et al., 2024; Wang et al., 2024; Luo et al., 2025), we train another GIN encoder ψ through two complementary tasks: *masked attribute reconstruction* and *graph–motif ranking*. Concretely, we mask the input attributes of a subset of nodes $\mathcal{V}^{(i)}_{\text{mask}}$ in each training graph $\mathcal{G}_i$. A decoder predicts their original attributes from the representations produced by ψ, yielding $\mathcal{L}_{\text{attr}}(\mathcal{G}_i) = \frac{1}{|\mathcal{V}^{(i)}_{\text{mask}}|} \sum_{v \in \mathcal{V}^{(i)}_{\text{mask}}} \left\| \vec{\mathbf{x}}_v - \text{MLP}(\vec{\mathbf{h}}_v) \right\|_2^2$. This objective encourages the encoder to retain information needed to recover individual node attributes. On the other hand, we supervise a graph readout from ψ, $\vec{\mathbf{h}}_{\mathcal{G}_i} = \text{READOUT}(\{\vec{\mathbf{h}}_v : v \in \mathcal{V}_{\mathcal{G}_i}\})$, using the motif-occurrence labels obtained during subgraph mining. Let $S_{\mathcal{G}_i}$ index the motifs occurring in $\mathcal{G}_i$. We encourage the graph representation to have higher similarity to embeddings of occurring motifs than to those of absent motifs:

$$\mathcal{L}_{\text{graph}}(\mathcal{G}_i) = \sum_{j \in S_{\mathcal{G}_i}} \sum_{k \notin S_{\mathcal{G}_i}} \max\Big(0,\, \gamma - \cos(\vec{\mathbf{h}}_{\mathcal{G}_i}, \vec{\mathbf{g}}_j) + \cos(\vec{\mathbf{h}}_{\mathcal{G}_i}, \vec{\mathbf{g}}_k)\Big), \tag{4}$$

where $\gamma > 0$ is the ranking margin. The same motif embeddings provide shared structural targets across graphs, and the ranking gradients guide node representation learning through the graph readout. Lastly, we jointly optimize $\mathcal{L}_{\text{node}} = \sum_{\mathcal{G}_i} \left[\mathcal{L}_{\text{attr}}(\mathcal{G}_i) + \lambda_{\text{g}}\mathcal{L}_{\text{graph}}(\mathcal{G}_i)\right]$.

Vector Quantization. After training, GMT freezes ψ and discretizes the frozen node embeddings with a codebook $\mathcal{C} = \{\vec{\mathbf{u}}_i\}_{i=1}^{K_{\text{N}}}$ shared across graphs, where $z_v = \arg\min_{i \in \{1,\ldots,K_{\text{N}}\}} \|\vec{\mathbf{h}}_v - \vec{\mathbf{u}}_i\|_2$. The codebook is updated by exponential moving averages of the assigned embeddings (van den Oord et al., 2017), not by backpropagation through $\mathcal{L}_{\text{VQ}}$. A reconstruction decoder D_ϕ is trained to recover $\vec{\mathbf{h}}_v$ from the stopped-gradient codeword by $\mathcal{L}_{\text{VQ}} = \frac{1}{n}\sum_{v=1}^{n} \left\| D_\phi\big(\text{sg}[\vec{\mathbf{u}}_{z_v}]\big) - \text{sg}[\vec{\mathbf{h}}_v] \right\|_2^2$, so the only parameters updated by this loss are those of D_ϕ. The discrete node token is the index z_v; subsequent pretraining uses the codebook vectors $\{\vec{\mathbf{u}}_{z_v}\}$.

4.3 Motif-Conditioned Masked Pretraining

Following pretrained language models (Devlin et al., 2019) and graph tokenizers that pretrain Transformers on discrete graph tokens (Zhao et al., 2025; Guo et al., 2026), we first pretrain a Transformer on unlabeled tokenized graphs and then fine-tune it for downstream tasks. we then pretrain a Transformer to learn interactions between the resulting tokens without relying on downstream task labels, offering an initialization for subsequent task-specific fine-tuning.

For each graph $\mathcal{G}$, we obtain two types of tokens: node tokens, with embeddings $\{\vec{\mathbf{u}}_{z_v}\}_{v \in \mathcal{V}}$, and motif tokens, with embeddings $\{\vec{\mathbf{g}}_j\}_{j \in S_{\mathcal{G}}}$. Node tokens encode node attributes and neighborhood information, whereas motif tokens describe recurring subgraph patterns present in the graph. Motivated by the structural limitations identified earlier, GMT leverages the latter as complementary context when learning from node tokens.

Token-Type-Aware Attention. Jointly processing these tokens requires learning both which individual tokens are relevant and how much information to draw from each type. To explicitly dis-

tinguish node context from motif context, we augment standard scaled dot-product attention with a learned bias for each query–key type pair:

$$\alpha_{ab}^{(h)} = \mathrm{softmax}_b\left(\frac{(\vec{\mathbf{q}}_a^{(h)})^\top \vec{\mathbf{k}}_b^{(h)}}{\sqrt{d_h}} + b_{t_a,t_b}^{(h)}\right), \tag{5}$$

where $t_a, t_b \in \{\mathrm{node}, \mathrm{motif}\}$, h indexes attention heads, and d_h is the dimension of each head. This bias allows each attention head to adjust the relative contributions of node and motif context, while query–key similarity scores determine which tokens to attend to within each type.

Motif-Conditioned Reconstruction. `GMT` corrupts a subset of node tokens $\mathcal{M}$ and retain all motif tokens as context. The Transformer predicts the original, frozen codebook embedding of each corrupted node:

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{|\mathcal{M}|}\sum_{v\in\mathcal{M}} \|\vec{\mathbf{u}}'_v - \vec{\mathbf{u}}_{z_v}\|_2^2 . \tag{6}$$

This objective trains the model to predict missing node information conditioned on the remaining node tokens and the graph's motif context. Using fixed codebook embeddings as targets connects pretraining directly to the learned node vocabulary. For downstream graph-level tasks, we keep the tokenizers fixed and fine-tune the Transformer and a task-specific prediction head.

5 Experiments

5.1 Experiment Setup

Datasets. We evaluate on twelve graph-level benchmarks spanning four task types. Single-task classification uses HIV, BACE, and BBBP (Hu et al., 2020), together with IMDB-B (Yanardag & Vishwanathan, 2015). Multi-task classification uses SIDER (27 side-effect tasks) and Tox21 (12 toxicity tasks) (Hu et al., 2020). Single-task regression uses ZINC, AQSOL (Dwivedi et al., 2023), ESOL (Hu et al., 2020), and Dielectric (Huan et al., 2016). Multi-task regression uses QM9 and QM8 (Ramakrishnan et al., 2014). Details of datasets in Appendix D.1.

Baselines. We compare against message passing GNNs (`GCN` (Kipf & Welling, 2016), `GIN` (Xu et al., 2018), `GAT` (Velickovic et al., 2017)), Graph Transformers (`GraphGPS` (Rampášek et al., 2022), `Exphormer` (Shirzad et al., 2023), `SimSGT` (Liu et al., 2023), `TokenGT` (Kim et al., 2022),`GraphTrans` (Ying et al., 2021)), serialization tokenizers (`GraphGPT` (Zhao et al., 2025), `GraphTokenizer` (Guo et al., 2026),`G`^{2}`PM` (Wang et al., 2025b)), quantized tokenizers (`GQT` (Wang et al., 2024), `NodeID` (Luo et al., 2025), `Mole-BERT`[2] (Xia et al., 2023), `VQGraph` (Yang et al., 2024)). A detailed account of each baseline is given in Appendix D.2.

Tasks and Evaluation Metrics. Single-task classification is scored by ROC-AUC (AUC), except IMDB-B which uses accuracy. Multi-task classification is scored by macro-ROC-AUC (mAUC), the unweighted mean of per-task ROC-AUC. Single-task regression is scored by MAE, except molesol which follows the OGB protocol and uses RMSE. Multi-task regression is scored by standard-macro (std-mMAE), the unweighted mean of per-target MAE. Subgraph matching retrieval is scored by mean Average Precision (mAP) and Hit@10; their formal definitions are given in Appendix D.3.

5.2 Overall Performance

Tables 1 and 2 evaluate four graph-level task types on twelve benchmarks spanning molecular, crystal, and social-network data, all under official splits. We evaluate `GMT` on a standard Transformer and do not scale the backbone to obtain the reported results. `GMT` is state-of-the-art on eleven of the twelve datasets and is second only on HIV, slightly behind `GraphGPT` (0.789 vs. 0.795). However, single-task classification labels are coarse and class-imbalanced, so AUC does not fully measure whether a tokenizer encodes graph structure: quantized tokenizers remain close to serialization methods on classification (e.g., `VQGraph` 0.625 vs. `GraphGPT` 0.622 on SIDER) but fall off sharply on regression. `GMT` is best on every regression benchmark, reducing the ZINC MAE from 0.181 (`Exphormer`) to 0.150, and surpasses the jointly trained, end-to-end Graph Transformers `GraphGPS` and `Exphormer` on multi-task QM9/QM8. The tokenizer is frozen after unsupervised pretraining, so the Transformer sees only tokens at finetuning time. That a frozen tokenizer plus a standard Transformer remains competitive with jointly trained Graph Transformers is consistent with the token sequence already serving as a usable structural interface.

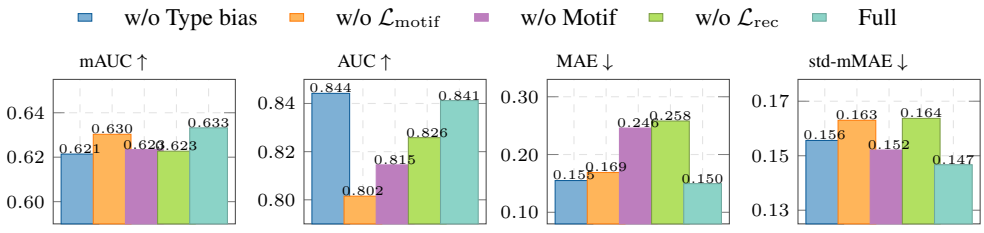
[2]Mole-BERT cannot be applied to non-molecular datasets.

Table 1: Overall performance on graph-level classification tasks. We report $\text{mean}_{\pm\text{std}}$. Best results **bolded** and runner-up underlined. "–" denotes a result that is not reported. Single-task classification uses ROC-AUC except IMDB-B (Accuracy); multi-task classification uses macro-ROC-AUC.

	Single-task classification ↑				Multi-task classification ↑	
Model	HIV	BACE	BBBP	IMDB-B	SIDER	Tox21
GCN	$0.756_{\pm 0.014}$	$0.745_{\pm 0.035}$	$0.640_{\pm 0.018}$	$0.570_{\pm 0.022}$	$0.602_{\pm 0.012}$	$0.743_{\pm 0.009}$
GIN	$0.747_{\pm 0.030}$	$0.751_{\pm 0.012}$	$0.608_{\pm 0.033}$	$0.625_{\pm 0.008}$	$0.586_{\pm 0.020}$	$0.742_{\pm 0.008}$
GAT	$0.751_{\pm 0.027}$	$0.753_{\pm 0.031}$	$0.664_{\pm 0.026}$	$0.539_{\pm 0.015}$	$0.605_{\pm 0.013}$	$0.755_{\pm 0.006}$
GraphGPS	$0.787_{\pm 0.005}$	$0.802_{\pm 0.002}$	$0.642_{\pm 0.003}$	$\underline{0.654}_{\pm 0.022}$	$0.564_{\pm 0.005}$	$0.734_{\pm 0.007}$
Exphormer	$0.763_{\pm 0.014}$	$0.796_{\pm 0.028}$	$0.681_{\pm 0.011}$	$0.619_{\pm 0.008}$	$0.577_{\pm 0.008}$	$0.749_{\pm 0.005}$
GraphTrans	$0.772_{\pm 0.012}$	$0.737_{\pm 0.020}$	$0.680_{\pm 0.007}$	$0.622_{\pm 0.009}$	$0.600_{\pm 0.007}$	$\underline{0.779}_{\pm 0.007}$
SimSGT	$0.770_{\pm 0.011}$	$\underline{0.822}_{\pm 0.009}$	$0.689_{\pm 0.006}$	$0.634_{\pm 0.014}$	$\underline{0.627}_{\pm 0.010}$	$0.763_{\pm 0.006}$
TokenGT	$0.768_{\pm 0.032}$	$0.720_{\pm 0.002}$	$\underline{0.707}_{\pm 0.002}$	$0.563_{\pm 0.029}$	$0.621_{\pm 0.001}$	$0.747_{\pm 0.001}$
GraphGPT	$\mathbf{0.795}_{\pm 0.014}$	$0.820_{\pm 0.018}$	$0.664_{\pm 0.012}$	$0.539_{\pm 0.009}$	$0.622_{\pm 0.022}$	$0.776_{\pm 0.006}$
GraphTokenizer	$0.789_{\pm 0.017}$	$0.773_{\pm 0.027}$	$0.623_{\pm 0.009}$	$0.652_{\pm 0.020}$	$0.608_{\pm 0.034}$	$0.717_{\pm 0.004}$
G^2PM	$0.745_{\pm 0.092}$	$0.706_{\pm 0.049}$	$0.588_{\pm 0.039}$	$0.495_{\pm 0.000}$	$0.603_{\pm 0.032}$	$0.703_{\pm 0.007}$
GQT	$0.699_{\pm 0.033}$	$0.724_{\pm 0.066}$	$0.602_{\pm 0.041}$	$0.568_{\pm 0.014}$	$0.574_{\pm 0.009}$	$0.724_{\pm 0.007}$
NodeID	$0.703_{\pm 0.030}$	$0.722_{\pm 0.044}$	$0.650_{\pm 0.028}$	$0.562_{\pm 0.013}$	$0.563_{\pm 0.021}$	$0.739_{\pm 0.012}$
Mole-BERT	$0.736_{\pm 0.014}$	$0.677_{\pm 0.044}$	$0.647_{\pm 0.020}$	–	$0.600_{\pm 0.023}$	$0.717_{\pm 0.004}$
VQGraph	$0.749_{\pm 0.033}$	$0.765_{\pm 0.044}$	$0.683_{\pm 0.012}$	$0.577_{\pm 0.009}$	$0.625_{\pm 0.023}$	$0.752_{\pm 0.011}$
GMT	$\underline{0.791}_{\pm 0.032}$	$\mathbf{0.845}_{\pm 0.004}$	$\mathbf{0.714}_{\pm 0.011}$	$\mathbf{0.664}_{\pm 0.014}$	$\mathbf{0.633}_{\pm 0.016}$	$\mathbf{0.782}_{\pm 0.003}$

Table 2: Overall performance on graph-level regression tasks. We report $\text{mean}_{\pm\text{std}}$. Best results are in **bold** and second-best are underlined. "–" denotes a result that is not reported. Single-task regression uses MAE except ESOL (RMSE); multi-task regression uses macro-MAE.

	Single-task regression ↓				Multi-task regression ↓	
Model	ZINC	ESOL	AQSOL	Dielectric	QM9	QM8
GCN	$0.459_{\pm 0.012}$	$\underline{0.865}_{\pm 0.061}$	$1.540_{\pm 0.021}$	$0.350_{\pm 0.076}$	$0.146_{\pm 0.003}$	$0.511_{\pm 0.004}$
GIN	$0.274_{\pm 0.007}$	$0.882_{\pm 0.026}$	$1.424_{\pm 0.025}$	$0.330_{\pm 0.080}$	$0.113_{\pm 0.002}$	$0.219_{\pm 0.001}$
GAT	$0.523_{\pm 0.022}$	$0.919_{\pm 0.066}$	$1.455_{\pm 0.028}$	$0.335_{\pm 0.066}$	$0.150_{\pm 0.004}$	$0.370_{\pm 0.010}$
GraphGPS	$0.195_{\pm 0.011}$	$0.917_{\pm 0.064}$	$\underline{0.944}_{\pm 0.020}$	$0.328_{\pm 0.083}$	$0.056_{\pm 0.004}$	$\underline{0.149}_{\pm 0.004}$
Exphormer	$\underline{0.181}_{\pm 0.005}$	$0.948_{\pm 0.037}$	$0.991_{\pm 0.006}$	$\underline{0.303}_{\pm 0.080}$	$\underline{0.054}_{\pm 0.002}$	$0.150_{\pm 0.002}$
GraphTrans	$0.219_{\pm 0.006}$	$0.912_{\pm 0.032}$	$1.106_{\pm 0.010}$	$0.519_{\pm 0.050}$	$0.092_{\pm 0.002}$	$0.219_{\pm 0.002}$
SimSGT	$0.274_{\pm 0.009}$	$0.888_{\pm 0.049}$	$1.107_{\pm 0.015}$	$0.359_{\pm 0.078}$	$0.411_{\pm 0.003}$	$0.215_{\pm 0.003}$
TokenGT	$0.321_{\pm 0.010}$	$0.872_{\pm 0.008}$	$1.116_{\pm 0.014}$	$0.305_{\pm 0.080}$	$0.085_{\pm 0.002}$	$0.152_{\pm 0.003}$
GraphGPT	$0.222_{\pm 0.029}$	$0.943_{\pm 0.037}$	$1.088_{\pm 0.023}$	$0.350_{\pm 0.085}$	$0.234_{\pm 0.048}$	$0.327_{\pm 0.006}$
GraphTokenizer	$0.269_{\pm 0.009}$	$0.947_{\pm 0.052}$	$1.138_{\pm 0.029}$	$0.427_{\pm 0.079}$	$0.114_{\pm 0.001}$	$0.547_{\pm 0.003}$
G^2PM	$0.459_{\pm 0.006}$	$1.246_{\pm 0.131}$	$1.574_{\pm 0.011}$	$0.513_{\pm 0.053}$	$0.348_{\pm 0.007}$	$0.441_{\pm 0.009}$
GQT	$0.511_{\pm 0.013}$	$1.709_{\pm 0.077}$	$1.091_{\pm 0.017}$	$0.555_{\pm 0.085}$	$0.183_{\pm 0.017}$	$0.204_{\pm 0.005}$
NodeID	$0.394_{\pm 0.017}$	$1.639_{\pm 0.206}$	$1.003_{\pm 0.017}$	$0.445_{\pm 0.077}$	$0.174_{\pm 0.016}$	$0.176_{\pm 0.002}$
Mole-BERT	$0.572_{\pm 0.009}$	$1.032_{\pm 0.075}$	$1.104_{\pm 0.012}$	–	$0.188_{\pm 0.009}$	$0.205_{\pm 0.004}$
VQGraph	$0.481_{\pm 0.011}$	$1.035_{\pm 0.051}$	$1.113_{\pm 0.025}$	$0.497_{\pm 0.084}$	$0.178_{\pm 0.004}$	$0.449_{\pm 0.005}$
GMT	$\mathbf{0.150}_{\pm 0.009}$	$\mathbf{0.844}_{\pm 0.028}$	$\mathbf{0.927}_{\pm 0.023}$	$\mathbf{0.293}_{\pm 0.003}$	$\mathbf{0.052}_{\pm 0.003}$	$\mathbf{0.147}_{\pm 0.003}$

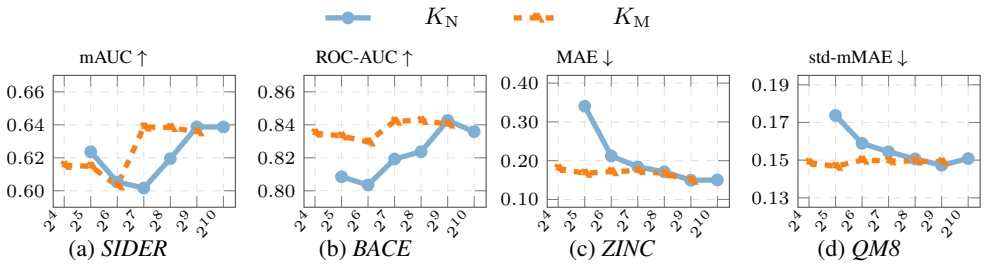


(a) *SIDER* (b) *BACE* (c) *ZINC* (d) *QM8*

Figure 3: Ablation study.

5.3 Ablation Study

Figure 3 ablates components under one shared, exactly labeled motif inventory. w/o Motif still applies masked reconstruction, but removes motif tokens. Accuracy drops relative to Full on every dataset; the ZINC MAE rises from 0.150 to 0.246, so motif tokens add information to the graph representation. w/o $\mathcal{L}_{\text{motif}}$ still appends $\{s_j\}$ according to $S_{\mathcal{G}}$, but the motif encoder is randomly initialized. It remains better than w/o Motif on ZINC and SIDER, so extra motif symbols can help even without a trained encoder. BACE, however, falls from 0.841 to 0.802: under the same inventory, a trained motif encoder uses those symbols more effectively than a random one. w/o $\mathcal{L}_{\text{rec}}$ keeps the same tokens but drops motif-conditioned masking. The ZINC MAE rises to 0.258, above w/o Motif, so pretraining must read motif symbols as context rather than merely insert them. w/o Type bias raises the BACE AUC from 0.841 to 0.844 and drops slightly elsewhere; the bias only marks the two token types and is not the source of the gains.



(a) *SIDER* (b) *BACE* (c) *ZINC* (d) *QM8*

Figure 4: Sensitivity to codebook size.

5.4 Hyperparameter Sensitivity

Figure 4 varies K_N and K_M under the full method to check two observations in Sec. 3.2: enlarging the allocated K_N does not further increase the number of active codes $\widetilde{K}_N$; simply enlarging the codebook, here by raising K_M, also does not improve performance without bound. When K_N is small the allocated alphabet itself is small, so Intra GTC is more severe: the ZINC MAE is 0.340 at K_N=32 and 0.150 at K_N=512, and the QM8 MAE is 0.174 and 0.147 at these two sizes. Raising K_N to 1024 no longer helps ZINC and slightly hurts BACE and QM8, so $\widetilde{K}_N$ does not grow without bound in K_N, as already seen in Fig. 1(a). The K_M curves are flatter. QM8 graphs have only about 8 nodes on average, so few distinct key motifs can occur, and K_M from 16 to 512 barely moves. BACE is already near its full setting at K_M=16. ZINC falls from 0.180 to 0.150, a much smaller swing than changing K_N. SIDER saturates after K_M=128. Some datasets are therefore insensitive to a larger K_M: the vocabulary need only cover the small set of key motifs at that graph scale, not grow further.

5.5 Token-Collapse Analysis

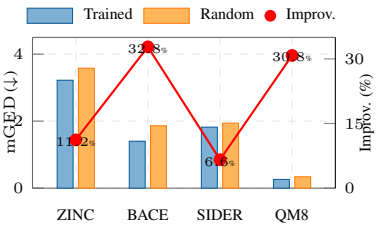


Figure 5: Trained vs. random motif tokens.

Introducing distinct motif tokens into each graph's token set has already partly relieved Intra GTC, so the absolute mGED @200 gap remains small on all datasets. Figure 5 shows the same direction on ZINC, BACE, SIDER, and QM8: motif tokens trained with Soft Matching with Collision Regularization attain a lower mGED @200 than a random initialization. Relative to the trained tokens, random embeddings raise mGED @200 by 6.6%–32.9%, with the largest increase on BACE (32.9%). As shown in §5.3, removing $\mathcal{L}_{\text{motif}}$ is the same control: the motif tokenizer no longer receives $\mathcal{L}_{\text{rank}}$ or $\mathcal{L}_{\text{coll}}$. On BACE this variant reaches an AUC of 0.802, below the 0.815 obtained when motif tokens are dropped entirely. Unsupervised s_j therefore widen the structural distance of token-space neighbors on BACE and weaken downstream discrimination. The ranking loss $\mathcal{L}_{\text{rank}}$ uses motif–graph occurrence labels so that s_j separates graphs that contain a template from those that do not, and the collision penalty keeps soft assignments injective across template nodes, writing reusable subgraph roles into the token space and lowering Inter GTC consistently.

5.6 Conclusion

We identified two structural weaknesses of quantized node-token sets for graph-level learning: intra-graph token collapse and inter-graph collisions caused by representing each graph only as an unordered multiset of local node codes, without explicitly encoding its distinctive local structural patterns. We introduced `GMT`, which learns reusable motif tokens from cross-graph occurrence supervision and uses them as context for masked node-token reconstruction. Across twelve benchmarks, `GMT` achieves the best result on eleven datasets and on every evaluated regression task. Ablation and token-collapse analyses show that the improvement comes from both the motif tokens and their learned association with graphs. These findings support motif tokens as a compact structural interface for graph Transformers, while leaving scalable motif discovery and adaptive vocabularies as important directions for future work.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, organization of some sections, and clarification of technical explanations. They were not used to generate experimental data, fabricate results, conduct the final statistical analysis, or make final decisions about the scientific claims. AI tools were also used to suggest code explanations and debugging ideas where appropriate; all such suggestions were inspected, modified, and tested by the authors.

The authors reviewed all AI-assisted content, verified the equations, proofs, implementation, experimental results, and citations, and take full responsibility for the final manuscript and all materials submitted with it.

ETHICS STATEMENT

This work uses publicly available graph datasets and does not involve human participants, personally identifiable information, private data, or human subject experiments. The datasets and software used in this study are employed for research purposes in accordance with their respective licenses. The proposed graph tokenization method is a general-purpose representation learning technique and is not designed for decisions about individuals or access to sensitive resources. We acknowledge that learned graph representations may inherit biases or artifacts from the underlying datasets; therefore, reported results should be interpreted within the scope of the evaluated datasets and tasks. The authors are responsible for the ethical use of the released code and materials.

REPRODUCIBILITY STATEMENT

We provide an anonymous code repository at https://anonymous.4open.science/r/GMT-27. The repository contains the implementation of the proposed graph tokenization framework, motif mining and matching procedures, training and fine-tuning scripts, dataset preprocessing code, configuration files, and evaluation scripts. The paper and appendix describe the model architecture, objective functions, experimental protocols, dataset splits, and hyperparameters needed to reproduce the reported results. The repository also includes instructions for reproducing the main tables and ablations. All code and materials have been checked to remove author names, institutional references, and other identifying metadata.

REFERENCES

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *International Conference on Learning Representations*, 2023. 2

Xiaohui Chen, Yinkai Wang, Jia-Xing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Liping Liu. Graph generative pre-trained transformer. *ArXiv*, abs/2501.01073, 2025. 2

Yongqiang Chen, Quan-Ming Yao, Juzheng Zhang, James Cheng, and Yatao Bian. Hierarchical graph tokenization for molecule-language alignment. In *International Conference on Machine Learning*, 2024. 2

Animesh Basak Chowdhury, Benjamin Tan, Ramesh Karri, and Siddharth Garg. Openabc-d: A large-scale dataset for machine learning guided integrated circuit synthesis. *arXiv preprint arXiv:2110.11292*, 2021. 1

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10): 1367–1372, 2004. D.2

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503–1507, 2008. 4.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. 4.3

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):1–48, 2023. 5.1, D.1

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, 2017. 1

Zeyuan Guo, Enmao Diao, Cheng Yang, and Chuan Shi. Graph tokenization for bridging graphs and transformers. In *International Conference on Learning Representations (ICLR)*, 2026. 1, 2, 4.3, 5.1, 5, D.2, E.3

C. Hierholzer and Christian Wiener. Ueber die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6:30–32, 1873. E.3

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020. 5.1, D.1

Tran Doan Huan, Arun Mannodi-Kanakkithodi, Chiho Kim, Vinit Sharma, Ghanshyam Pilania, and Rampi Ramprasad. A polymer dataset for accelerated property prediction and design. *Scientific Data*, 3:160012, 2016. doi: 10.1038/sdata.2016.12. 5.1, D.1

Md Shamim Hussain, Mohammed J. Zaki, and D. Subramanian. Global self-attention as a replacement for graph convolution. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021. 2

Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *International Conference on Learning Representations*, 2023. 2

Wei Jin, Xiaorui Liu, Yao Ma, Charu Aggarwal, and Jiliang Tang. Feature overcorrelation in deep graph neural networks: A new perspective. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 709–719, 2022. 3.2

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv*, abs/2207.02505, 2022. 1, 2, 5.1, 5, D.2

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016. 1, 2, 5.1, 5, D.2

Haoyang Li, Yuming Xu, Alexander Zhou, Yongqi Zhang, Jason Chen Zhang, Lei Chen, and Qing Li. Openglt: A comprehensive benchmark of graph neural networks for graph-level tasks. *arXiv preprint arXiv:2501.00773*, 2025. 1

Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Rethinking tokenizer and decoder in masked graph modeling for molecules. *ArXiv*, abs/2310.14753, 2023. 2, 5.1, 5, D.2

Zhouyang Liu, Ning Liu, Yixin Chen, Jiezhong He, Menghan Jia, and Dongsheng Li. Hierarchy-aware neural subgraph matching with enhanced similarity measure. *IEEE Transactions on Knowledge and Data Engineering*, 37:5298–5310, 2025. D.2

Yuankai Luo, Hongkang Li, Qijiong Liu, Lei Shi, and Xiao-Ming Wu. Node identifiers: Compact, discrete representations for efficient graph learning. In *ICLR*, 2025. 1, 2, 3.2, 4.2, 5.1, 5, D.2

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet Kumar Dokania, Mark Coates, Philip H. S. Torr, and Ser Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, 2023. 2

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv: Learning*, 2019. 3.2

Vaibhav Raj, Indradyumna Roy, Ashwin Ramachandran, Soumen Chakrabarti, and Abir De. Charting the design space of neural graph representations for subgraph matching, 2025. D.2

Ashwin Ramachandran, Vaibhav Raj, Indrayumna Roy, Soumen Chakrabarti, and Abir De. Iteratively refined early interaction alignment for subgraph matching based graph retrieval. NIPS '24, 2024. D.2

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. 5.1, D.1

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *ArXiv*, abs/2205.12454, 2022. 2, 5.1, 5, D.2

Indradyumna Roy, Venkata Velugoti, Soumen Chakrabarti, and Abir De. Interpretable neural subgraph matching for graph retrieval. In *AAAI Conference on Artificial Intelligence*, 2022. D.2

Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983. 3.2

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909, 2015. 2, E.3

Hamed Shirzad, Ameya Velingker, B. Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 2023. 2, 5.1, 5, D.2

Yuhta Takida, Takashi Shibuya, WeiHsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. SQ-VAE: Variational bayes on discrete representation with self-annealed stochastic quantization. In *International Conference on Machine Learning*, 2022. 3.2

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 491–500, 2024. 2

Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160, 1972. E.3

J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976. 3.1, D.2

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Neural Information Processing Systems*, 2017. 3.1, 4.2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. 2

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017. 2, 5.1, 5, D.2

Limei Wang, Kaveh Hassani, Si Zhang, Dong-Qi Fu, Baichuan Yuan, Wei-Lin Cong, Zhi-Gang Hua, Hao Wu, Ning Yao, and Bo Long. Learning graph quantized tokenizers. In *International Conference on Learning Representations*, 2024. 1, 2, 3.2, 4.2, 5.1, 5, D.2

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. Beyond message passing: Neural graph pattern machine. In *International Conference on Machine Learning*, 2025a. 1, 2

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Generative graph pattern machine. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. 2, 5.1, 5, D.2, E.3

Nan Wu, Jiwon Lee, Yuan Xie, and Cong Hao. Lostin: Logic optimization via spatio-temporal information with hybrid graph models. In *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 11–18. IEEE, 2022. 1

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 1

Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:259298531. 1, 2, 3.2, 4.2, 5.1, 5, D.2

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ArXiv*, abs/1810.00826, 2018. 2, 4.1, 5.1, 5, D.2

Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. ICDM '02, USA, 2002. 3.1

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015. 5.1, D.1

Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, Bin CUI, Muhan Zhang, and Jure Leskovec. Vqgraph: Rethinking graph representation space for bridging gnns and mlps. In *ICLR*, 2024. 1, 2, 3.2, 4.2, 5.1, 5, D.2

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *ArXiv*, abs/2106.05234, 2021. 1, 2, 5.1, 5, D.2

Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. *ArXiv*, abs/2007.03092, 2020. D.2

Qifang Zhao, Weidong Ren, Tianyu Li, Hong Liu, Xingsheng He, and Xiaoxiao Xu. GraphGPT: Generative pre-trained graph eulerian transformer. In *Forty-second International Conference on Machine Learning*, 2025. 1, 2, 4.3, 5.1, 5, D.2, E.3

A Appendix

B Algorithm and Complexity Analysis

The full pipeline is summarized in Algorithm 1. Motif mining and the true motif–graph matchings are computed once in a C++ preprocessing pass and then cached; subsequent training only reads these stored pairs and never reruns subgraph isomorphism.

Tokenizer Complexity. We analyze only the learnable tokenizers. The C++ `gSpan` pass that produces $\mathcal{S}$, the occurrence sets $\{S_{\mathcal{G}}\}$, and the matched pairs $\mathcal{P}$ is a one-time preprocessing step and is not included.

Consider a cached pair $(g_j, \mathcal{G}_i)$ with $s = |\mathcal{V}_{g_j}|$ template nodes, $n = |\mathcal{V}_{\mathcal{G}_i}|$ host nodes, e host edges, GIN depth L, and hidden size d. A GIN forward pass costs $O(Led^2)$. Given the node embeddings, the soft matching cost $\Delta(g_j, \mathcal{G}_i)$ evaluates $s \times n$ cosine similarities and a row-wise softmax, which is $O(snd)$. The collision term in Eq. equation 3 is $O(s^2 n)$. Thus one motif-tokenizer step on a precomputed pair is

$$O\big(Led^2 + snd\big), \tag{7}$$

and training ϕ simply repeats this over minibatches drawn from $\mathcal{P}$. No subgraph-isomorphism search is performed at training time.

The node tokenizer encodes $\mathcal{G}_i$ once with ψ in $O(Led^2)$ and ranks the graph readout against a sampled set of P motif tokens in $O(Pd)$. Vector quantization of n frozen embeddings against K_{N} codewords costs $O(nK_{\mathrm{N}}d)$. After both tokenizers are frozen, tokenizing a graph is one ψ pass, one VQ assignment, and a lookup of the $m = |S_{\mathcal{G}}|$ cached motif embeddings, i.e., $O(Led^2 + nK_{\mathrm{N}}d)$. The subsequent Transformer acts on a sequence of length $L_{\mathrm{seq}} = n+m$ and is the standard $O(L_{\mathrm{seq}}^2 d)$ attention cost; we do not attribute it to the tokenizer.

Algorithm 1: Training pipeline of `GMT`

Input: training graphs $\mathcal{D}$
Output: frozen $\phi, \psi, \mathcal{C}$; pretrained Transformer

`// One-time C++ preprocessing` $\mathcal{S}, \{S_{\mathcal{G}}\}, \mathcal{P} \leftarrow \texttt{gSpan}(\mathcal{D})$

`// Motif tokenizer` **foreach** *cached pair* $(g_j, \mathcal{G}_i) \in \mathcal{P}$ **do**
  $\vec{\mathbf{h}}_w \leftarrow \phi(g_j)_w, \quad \vec{\mathbf{h}}_v \leftarrow \phi(\mathcal{G}_i)_v$
  compute soft assignment B^j and matching cost $\Delta(g_j, \mathcal{G}_i)$
end
update ϕ with $\mathcal{L}_{\text{motif}}$
foreach $g_j \in \mathcal{S}$ **do**
  $\vec{\mathbf{g}}_j \leftarrow \text{READOUT}(\{\phi(g_j)_w\})$
end
freeze ϕ and $\{\vec{\mathbf{g}}_j\}$

`// Node tokenizer` **foreach** $\mathcal{G}_i \in \mathcal{D}$ **do**
  mask a subset of node attributes and encode $\vec{\mathbf{h}}_v \leftarrow \psi(\mathcal{G}_i)_v$
  reconstruct the masked attributes
  $\vec{\mathbf{h}}_{\mathcal{G}_i} \leftarrow \text{READOUT}(\{\vec{\mathbf{h}}_v\})$
  rank $\vec{\mathbf{h}}_{\mathcal{G}_i}$ against occurring vs. non-occurring motif tokens
end
update ψ with $\mathcal{L}_{\text{node}}$
freeze ψ; quantize $\vec{\mathbf{h}}_v$ to z_v with codebook $\mathcal{C}$

`// Pretraining and fine-tuning` $\mathcal{T}(\mathcal{G}) \leftarrow \{\vec{\mathbf{u}}_{z_v}\} \cup \{\vec{\mathbf{g}}_j\}_{j \in S_{\mathcal{G}}}$
pretrain with $\mathcal{L}_{\text{rec}}$; freeze tokenizers and fine-tune the Transformer

Table 3: Additional notation used in the paper.

Symbol	Meaning
$\mathcal{D} = \{\mathcal{G}_i\}$	training graph collection
$g = (\mathcal{V}_g, \mathcal{E}_g) \subseteq \mathcal{G}$	subgraph of $\mathcal{G}$
$\pi : g \hookrightarrow \mathcal{G}$	subgraph isomorphism
$S = \{g_1, \ldots, g_{\lvert S \rvert}\}$	motif codebook
g_j	j-th motif template
$S_{\mathcal{G}}$	indices of templates occurring in $\mathcal{G}$
$m = \lvert S_{\mathcal{G}} \rvert$	number of occurring motifs
$\mathcal{C} = \{\vec{\mathbf{u}}_i\}_{i=1}^{K_{\text{N}}}$	node VQ codebook
z_v	node token
s_j	motif token
$\mathcal{T}(\mathcal{G})$	graph tokenization
$\boldsymbol{B}(\mathcal{G})$, $\boldsymbol{B}_{vj}$	node–motif incidence
$\widehat{\boldsymbol{B}}$, $\widehat{\boldsymbol{B}}^{(j)}$	soft incidence; column of g_j
$\mathcal{R}_j$	soft support of template j
$\mathcal{U}(p, K)$	number of unordered p-token bags
$\mathcal{G}_{n,e,d_0}$	attributed graphs with n, e, d_0
$t \in \{\text{n}, \text{m}\}$	token type
$\mathcal{M} \subset \mathcal{V}$	masked nodes

B.1 Motif Mining

Let $\mathcal{D}_{\text{tr}}$, $\mathcal{D}_{\text{va}}$, and $\mathcal{D}_{\text{te}}$ denote the training, validation, and test splits. To avoid information leakage, motif mining and motif-vocabulary construction are performed exclusively on $\mathcal{D}_{\text{tr}}$.

Motif construction. We run `gSpan` once on the training graphs and retain the K_{M} frequent connected subgraphs with the highest support:

$$\mathcal{S}_{\mathrm{tr}} = \{g_1, \ldots, g_{K_{\mathrm{M}}}\}.$$

For each training graph $\mathcal{G}_i \in \mathcal{D}_{\mathrm{tr}}$, `gSpan` provides graph-level occurrence indicators

$$y_{ij} = \mathbb{1}[g_j \text{ occurs in } \mathcal{G}_i], \qquad S_{\mathcal{G}_i} = \{j : y_{ij} = 1\}.$$

The occurrence sets $S_{\mathcal{G}_i}$, together with the cached motif–graph matching pairs, are used to train the motif encoder and the node encoder. We do not run `gSpan` on validation or test graphs to expand the vocabulary, and no validation or test graph contributes a motif template or a training loss.

Occurrence-supervised reusable motif prototypes. For a fixed motif g_j, let

$$\mathcal{P}_j = \{\mathcal{G}_i \in \mathcal{D}_{\mathrm{tr}} : y_{ij} = 1\}, \qquad \mathcal{N}_j = \{\mathcal{G}_i \in \mathcal{D}_{\mathrm{tr}} : y_{ij} = 0\}.$$

The same motif encoder and motif embedding $\vec{\mathbf{g}}_j$ are shared across all host graphs. The ranking loss can be written as

$$\mathcal{L}_{\mathrm{rank}} = \sum_i \sum_{j:y_{ij}=1} \sum_{k:y_{ik}=0} \left[\gamma + \Delta_\phi(g_j, \mathcal{G}_i) - \Delta_\phi(g_k, \mathcal{G}_i)\right]_+ .$$

Thus, graphs in $\mathcal{P}_j$ repeatedly provide positive compatibility signals for g_j, while graphs in $\mathcal{N}_j$ provide negative contrast. Because the motif and host encoders are shared across graphs, $\vec{\mathbf{g}}_j$ is encouraged to represent local host environments that recur across graphs containing g_j, rather than a graph-specific identifier.

The soft assignment matrix B^j provides a differentiable compatibility profile over host nodes. In particular, we define

$$p_j(v \mid \mathcal{G}_i) = \frac{1}{|\mathcal{V}_{g_j}|} \sum_{w \in \mathcal{V}_{g_j}} B^j_{w,v}.$$

This quantity measures how strongly host node v is compatible with motif g_j. The collision regularizer discourages the degenerate solution in which multiple motif nodes assign their probability mass to the same host node. Nevertheless, $p_j(v \mid \mathcal{G}_i)$ is interpreted as soft compatibility evidence rather than an exact subgraph-isomorphism certificate, because the supervision specifies graph-level occurrence and does not uniquely identify a node correspondence.

Test-time motif selection. After training, $\mathcal{S}_{\mathrm{tr}}$, the motif encoder, and the motif embeddings are frozen. For an unseen graph $\mathcal{G}$, we encode the graph with the frozen motif encoder and compute a compatibility score for every candidate motif in the training-derived vocabulary:

$$r_j(\mathcal{G}) = -\Delta_\phi(g_j, \mathcal{G}), \qquad g_j \in \mathcal{S}_{\mathrm{tr}}.$$

We then select the motif set

$$\widehat{S}_{\mathcal{G}} = \{j : r_j(\mathcal{G}) \geq \tau\},$$

where τ is determined using the validation split only. If a fixed top-K rule is used instead, $\widehat{S}_{\mathcal{G}}$ contains the K motifs with the largest scores, with K fixed before test evaluation. The final token set is

$$T(\mathcal{G}) = \{\vec{\mathbf{u}}_{z_v} : v \in \mathcal{V}_{\mathcal{G}}\} \cup \{\vec{\mathbf{g}}_j : j \in \widehat{S}_{\mathcal{G}}\}.$$

Therefore, test-time motif selection uses the learned motif tokenizer and the fixed training-derived vocabulary; it does not use test occurrence labels, rerun motif mining, or solve an exact subgraph-isomorphism problem.

C Theoretical Proofs

C.1 Proof of $\mathcal{U}(n, \widetilde{K}_{\mathrm{N}}) \ll |\mathcal{G}_{n,d_0}|$

We interpret

$$\mathcal{U}(n, \widetilde{K}_{\mathrm{N}}) \ll |\mathcal{G}_{n,d_0}|$$

as

$$\frac{\mathcal{U}(n, \widetilde{K}_{\mathrm{N}})}{|\mathcal{G}_{n,d_0}|} \longrightarrow 0 \qquad \text{as } n \to \infty.$$

Because node attributes take values in $\mathbb{Z}^{d_0}$, let $\mathcal{G}^{(0)}_{n,d_0} \subseteq \mathcal{G}_{n,d_0}$ denote the subset of attributed graphs in which every node has the zero attribute vector. It is sufficient to establish the result for this subset.

An unordered multiset of n codes drawn from a codebook of size $\widetilde{K}_{\mathrm{N}}$ is determined by the multiplicity of each codeword. Hence, by the stars-and-bars formula,

$$\mathcal{U}(n, \widetilde{K}_{\mathrm{N}}) = \binom{n + \widetilde{K}_{\mathrm{N}} - 1}{n}.$$

Under the capacity bound $\widetilde{K}_{\mathrm{N}} = n^{O(1)}$, we have

$$\log \mathcal{U}(n, \widetilde{K}_{\mathrm{N}}) = O(n \log n).$$

There are $\binom{n}{2}$ possible edges in a labeled simple undirected graph on n nodes, so the number of labeled graph topologies is $2^{\binom{n}{2}}$. Since each isomorphism class contains at most $n!$ labeled graphs,

$$\left|\mathcal{G}^{(0)}_{n,d_0}\right| \geq \frac{2^{\binom{n}{2}}}{n!}.$$

Therefore,

$$\log \left|\mathcal{G}^{(0)}_{n,d_0}\right| \geq \binom{n}{2} - \log(n!) = \Omega(n^2).$$

Since $\mathcal{G}^{(0)}_{n,d_0} \subseteq \mathcal{G}_{n,d_0}$,

$$\log |\mathcal{G}_{n,d_0}| \geq \Omega(n^2).$$

Consequently,

$$\log \frac{\mathcal{U}(n, K_{\mathrm{N}})}{|\mathcal{G}_{n,d_0}|} = \log \mathcal{U}(n, K_{\mathrm{N}}) - \log |\mathcal{G}_{n,d_0}| = O(n \log n) - \Omega(n^2) = -\Omega(n^2).$$

and thus

$$\frac{\mathcal{U}(n, \widetilde{K}_{\mathrm{N}})}{|\mathcal{G}_{n,d_0}|} \leq 2^{-\Omega(n^2)} \longrightarrow 0.$$

Hence,

$$\mathcal{U}(n, \widetilde{K}_{\mathrm{N}}) \ll |\mathcal{G}_{n,d_0}|$$

D EXPERIMENTAL SETUP

D.1 DATASETS

Sizes, average order and size, number of targets, metrics, and splits are given in Table 4; we do not repeat those numbers here. Molecular graphs use atoms as nodes and bonds as edges, with the official node/edge features of the source benchmark.

Single-task classification. HIV, BACE, and BBBP are MoleculeNet molecular property tasks released through OGB (Hu et al., 2020). Each graph carries one binary label: inhibition of HIV replication, binding to human β-secretase 1, and blood–brain barrier penetration, respectively. The three tasks are class-imbalanced, so the official score is ROC-AUC under the OGB scaffold split (Bemis–Murcko scaffolds keep validation/test molecules structurally distant from training). IMDB-B (IMDB-BINARY) (Yanardag & Vishwanathan, 2015) is a TUDataset social-network benchmark. Each graph is an actor/actress ego-network: nodes are people and an edge marks co-appearance in a movie; the binary label distinguishes Action from Romance collaborations. There are no native node attributes; we use (one-hot) degree features, as is standard. The official score is classification accuracy.

Multi-task classification. SIDER annotates marketed drugs with 27 binary side-effect labels grouped by organ system. Tox21 annotates compounds with 12 nuclear-receptor and stress-response

Table 4: Summary of benchmark datasets. Average edge counts are reported as undirected edges.

Task	Dataset	#Graphs	Avg. Nodes	Avg. Edges	#Tasks	Metric	Split
Classification	HIV	41,127	25.5	27.5	1	ROC-AUC	Scaffold
	BACE	1,513	34.1	36.9	1	ROC-AUC	Scaffold
	BBBP	2,039	24.1	26.0	1	ROC-AUC	Scaffold
	IMDB-B	1,000	19.8	96.5	1	Accuracy	Fixed 60/20/20
Multi-task clf.	SIDER	1,427	33.6	35.4	27	Macro-ROC-AUC	Scaffold
	Tox21	7,831	18.6	19.3	12	Macro-ROC-AUC	Scaffold
Regression	ZINC	12,000	23.2	24.9	1	MAE	10K/1K/1K
	ESOL	1,128	13.3	13.7	1	RMSE	Scaffold
	AQSOL	9,833	17.6	35.8	1	MAE	Official
	Dielectric	4,764	16.9	109.0	1	MAE	Official 5-fold
Multi-task reg.	QM9	130,831	18.0	18.6	12	Macro-MAE	100K/17.7K/13.1K
	QM8	21,786	7.8	8.1	12	Macro-MAE	80/10/10

Table 5: Official implementations of the baselines.

Category	Method	Code
MP-GNN	`GCN` (Kipf & Welling, 2016)	https://github.com/tkipf/gcn
	`GIN` (Xu et al., 2018)	https://github.com/snap-stanford/pretrain-gnns
	`GAT` (Velickovic et al., 2017)	https://github.com/PetarV-/GAT
Graph Transformer	`GraphGPS` (Rampášek et al., 2022)	https://github.com/rampasek/GraphGPS
	`Exphormer` (Shirzad et al., 2023)	https://github.com/hamed1375/Exphormer
	`SimSGT` (Liu et al., 2023)	https://github.com/syr-cn/SimSGT
	`TokenGT` (Kim et al., 2022)	https://github.com/jw9730/tokengt
	`GraphTrans` (Ying et al., 2021)	https://github.com/ucbrise/graphtrans
Serialization tokenizer	`GraphGPT` (Zhao et al., 2025)	https://github.com/alibaba/graph-gpt
	`GraphTokenizer` (Guo et al., 2026)	https://github.com/BUPT-GAMMA/Graph-Tokenization-for-Bridging-Graphs-and-Transformers
	`G`^{2}`PM` (Wang et al., 2025b)	https://github.com/Zehong-Wang/G2PM
Quantized tokenizer	`GQT` (Wang et al., 2024)	https://github.com/limei0307/GQT
	`NodeID` (Luo et al., 2025)	https://github.com/LUOyk1999/NodeID
	`Mole-BERT` (Xia et al., 2023)	https://github.com/junxia97/Mole-BERT
	`VQGraph` (Yang et al., 2024)	https://github.com/YangLing0818/VQGraph

toxicity assays (Hu et al., 2020). A molecule may be positive on several tasks, and a label may be missing; missing labels are ignored by the OGB evaluator for that task only. Both datasets use the official scaffold split and are scored by macro-ROC-AUC (the unweighted mean of the per-task ROC-AUC values).

Single-task regression. ZINC is the 12K molecular subset of Benchmarking GNNs (Dwivedi et al., 2023); the target is constrained solubility and the official score is MAE. AQSOL is built from AqSolDB (Dwivedi et al., 2023) and uses experimentally measured aqueous solubility, also scored by MAE under the official scaffold split of that benchmark. ESOL is the OGB release of the Delaney water-solubility set (`ogbg-molesol`) (Hu et al., 2020); following OGB we report RMSE, not MAE, on the scaffold split. Dielectric is the Khazana polymer set of Huan *et al.*, (Huan et al., 2016): polymer repeating units / related crystal structures with DFT-computed dielectric constants (the original release also provides band gaps and atomization energies). The prediction target used here is the dielectric constant, scored by MAE.

Multi-task regression. QM9 (Ramakrishnan et al., 2014) is scored under the 12-target protocol used in the graph-Transformer literature: dipole moment μ, isotropic polarizability α, HOMO, LUMO, HOMO–LUMO gap, electronic spatial extent $\langle R^2 \rangle$, ZPVE, internal energies U_0 and U, enthalpy H, free energy G, and heat capacity C_v. QM8 (Ramakrishnan et al., 2014) uses 12 electronic-spectra targets: the first two excitation energies and the corresponding oscillator strengths under CC2, PBE0, and CAM-B3LYP (E1/E2 and f1/f2 for each functional).

D.2 Baselines

We compare against five families of graph-level models and, on the retrieval task, against neural subgraph-matching methods. Unless a method is inherently molecule-specific, the same architecture is trained on every benchmark under the official split. Tokenizer methods freeze the tokenizer after

unsupervised (or official) pretraining and train only the downstream Transformer / prediction head; message-passing GNNs and Graph Transformers are trained end-to-end on the labeled split.

Message-passing GNNs. `GCN` (Kipf & Welling, 2016) applies a first-order spectral convolution: each node aggregates a symmetrically normalized average of its neighbors. `GIN` (Xu et al., 2018) replaces the average by a sum followed by an MLP, which is injective on multisets and matches the 1-WL test in the limit; we use the Graph Isomorphism Network with the standard sum readout. `GAT` (Velickovic et al., 2017) computes attention coefficients over neighbors and aggregates with those weights. For all three models a permutation-invariant readout (mean or sum) produces a graph vector that is passed to a linear classifier or regressor. These models keep topology inside message passing and do not expose a discrete token vocabulary.

Graph Transformers. `GraphGPS` (Rampášek et al., 2022) is a modular Graph Transformer: a message-passing layer and a global attention layer are stacked, and positional / structural encodings (e.g., random-walk or Laplacian PE) are added to node features. `Exphormer` (Shirzad et al., 2023) sparsifies attention with expander graphs (and optional virtual nodes / nearest-neighbor edges), retaining global mixing at sub-quadratic cost. `GraphTrans` (Ying et al., 2021) uses a GNN to encode local neighborhoods and a Transformer over the resulting node sequence to capture long-range context; the GNN is essential—the Transformer is not applied to raw atoms. `SimSGT` (Liu et al., 2023) is a masked graph-modeling method for molecules: a subgraph-level tokenizer discretizes the graph and a Transformer decoder reconstructs masked structure. All four inject topology into the architecture (encodings, sparse attention, or a GNN stem) rather than compiling it into a reusable codebook. `TokenGT` (Kim et al., 2022) treats nodes and edges as tokens, augments them with orthonormal node identifiers and type embeddings, and shows that a pure Transformer can recover GNN-like expressivity.

Serialization tokenizers. These methods flatten a graph into a sequence and feed a standard Transformer. `GraphGPT` (Zhao et al., 2025) serializes the graph along an Eulerian path and pretrains a generative Transformer on the resulting token stream. `GraphTokenizer` (Guo et al., 2026) first applies a reversible serialization guided by global substructure statistics, then merges frequent segments with BPE so that a BERT-style Transformer can be used without graph-specific layers. In all three cases, incidence between substructures is stored only in path order; attention cost grows with the length of the chain. `G`^{2}`PM` (Wang et al., 2025b) tokenizes a graph as a sequence of walk-extracted substructures and pretrains a generative Transformer on those sequences. Because exact subgraph matching is expensive, the patterns are partitioned per graph by random walks and do not form a unified codebook across the dataset. This is the closest prior tokenizer that treats subgraphs as tokens; it still does not write node–motif incidence into the codes.

Quantized tokenizers. Quantized node tokenization encodes each node with a GNN and maps the embedding onto a shared codebook, so a graph becomes an unordered multiset of discrete node tokens. `GQT` (Wang et al., 2024) learns a graph quantized tokenizer with residual vector quantization and uses the codes as input to a Transformer. `NodeID` (Luo et al., 2025) trains compact discrete node identifiers that can be reused across graphs and tasks. `Mole-BERT` (Xia et al., 2023) is a molecule-specific VQ tokenizer: atom contexts are quantized and the codes serve as a vocabulary for masked graph pretraining. `VQGraph` (Yang et al., 2024) quantizes GNN node embeddings into a structural codebook originally introduced to distill GNNs into MLPs; pooling the codes yields a graph-level vector. These methods share a vocabulary across graphs, but the observation is an unordered bag: they do not record how local codes compose into motifs.

Subgraph-matching. Exact subgraph isomorphism is NP-complete (Ullmann, 1976; Cordella et al., 2004). The retrieval experiment therefore compares against neural matchers that encode a query g and a host $\mathcal{G}$ and score containment, without requiring an explicit 0-1 correspondence at test time. `NeuroMatch` (Ying et al., 2020) embeds query and target neighborhoods with a GNN into an order-embedding space whose geometry respects subgraph relations (transitivity, antisymmetry), and decides containment by a coordinate-wise inequality (trained with a max-margin loss). `IsoNet` (Roy et al., 2022) is a late-interaction matcher: each graph is encoded independently, a differentiable Gumbel–Sinkhorn layer produces an (edge) alignment, and an asymmetric hinge counts uncovered query edges so that extra host structure is not penalized. Supervision is distant—only graph-level relevance, not node maps. `IsoNet++` (Ramachandran et al., 2024) replaces late interaction by early interaction: messages pass within and across the two graphs, guided by an injective alignment that is refined lazily over several rounds, and node-pair (edge) partner signals are used to update the alignment. `NC-Iso` (Liu et al., 2025) is hierarchy-aware: it preserves neighborhood features across

GNN layers and ranks hosts with an enhanced similarity that combines a compliance term with similarity dominance. NGDS (Raj et al., 2025) is the strong configuration identified by a systematic design-space study of neural graph representations for subgraph matching, spanning interaction stage (early vs. late), injective vs. attention alignment, node vs. edge granularity, and the form of the relevance head (set alignment vs. aggregated scoring). All five methods treat the correspondence as an auxiliary pairwise prediction.

D.3 Evaluation metrics

We write N_{te} for the number of test graphs, y_i (or y_{it}) for the ground-truth label of graph i (on task t), and $\hat{y}_i$ (or $\hat{y}_{it}$) for the corresponding prediction. For multi-label classification, T is the number of tasks and $\mathcal{I}_t \subseteq \{1, \dots, N_{\text{te}}\}$ is the set of test indices whose label on task t is observed (OGB missing labels, encoded as NaN, are dropped from that task only).

Accuracy (IMDB-B). IMDB-B is a balanced binary classification problem and is scored by accuracy,

$$\text{Acc} = \frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \mathbb{1}\big[\hat{y}_i = y_i\big], \tag{8}$$

where $\hat{y}_i$ is the predicted class. Higher is better.

ROC-AUC (single-task classification). For a binary label $y_i \in \{0, 1\}$ and a real-valued score p_i (the positive-class logit or probability), let $\mathcal{I}_{\text{pos}} = \{i : y_i = 1\}$ and $\mathcal{I}_{\text{neg}} = \{i : y_i = 0\}$. ROC-AUC is the probability that a random positive is ranked above a random negative,

$$\text{ROC-AUC} = \frac{1}{|\mathcal{I}_{\text{pos}}|\,|\mathcal{I}_{\text{neg}}|} \sum_{i \in \mathcal{I}_{\text{pos}}} \sum_{j \in \mathcal{I}_{\text{neg}}} \Big(\mathbb{1}[p_i > p_j] + \tfrac{1}{2}\mathbb{1}[p_i = p_j] \Big). \tag{9}$$

Equivalently, it is the area under the ROC curve obtained by sweeping the decision threshold on p_i. We use the OGB evaluator. Higher is better.

Macro-ROC-AUC (SIDER, Tox21). Each of the T binary tasks is scored independently by ROC-AUC on its observed labels, then averaged with equal task weight:

$$\text{Macro-ROC-AUC} = \frac{1}{T} \sum_{t=1}^{T} \text{ROC-AUC}_t, \qquad T = \begin{cases} 27 & \text{SIDER}, \\ 12 & \text{Tox21}. \end{cases} \tag{10}$$

Tasks are not weighted by prevalence or by the number of observed labels. Higher is better.

MAE (ZINC, AQSOL, Dielectric). Single-task regression (except ESOL) is scored by mean absolute error

$$\text{MAE} = \frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \big|\hat{y}_i - y_i\big|. \tag{11}$$

Lower is better. The physical unit is that of the official target (constrained solubility, aqueous solubility, dielectric constant).

RMSE (ESOL / `ogbg-molesol`). Following the OGB protocol, ESOL is scored by root mean squared error

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \big(\hat{y}_i - y_i\big)^2}. \tag{12}$$

Lower is better.

Macro-MAE. For a T-task regression problem, the (unnormalized) macro-MAE is the unweighted mean of per-task MAE,

$$\text{Macro-MAE} = \frac{1}{T} \sum_{t=1}^{T} \text{MAE}_t, \qquad \text{MAE}_t = \frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \big|\hat{y}_{it} - y_{it}\big|. \tag{13}$$

"Macro" means equal task weight only; it does *not* standardize scales. On QM9 and QM8 this quantity is dominated by large-magnitude targets and is *not* the number we report.

Standardized Macro-MAE (QM9 and QM8). QM9 and QM8 are multi-target regressions; we write T for the number of targets on the dataset being scored. To avoid a few wide-range properties

dominating the average, we normalize each task by the *training-set* standard deviation of that target and then take the unweighted mean. Let N_{tr} be the number of training graphs. For each task $t = 1, \dots, T$,

$$\mu_t = \frac{1}{N_{\text{tr}}} \sum_{i \in \text{train}} y_{it}, \qquad \sigma_t = \sqrt{\frac{1}{N_{\text{tr}}} \sum_{i \in \text{train}} (y_{it} - \mu_t)^2}. \tag{14}$$

We use the population standard deviation (denominator N_{tr}). If an implementation uses the sample standard deviation, the denominator is $N_{\text{tr}} - 1$; reproduction should follow the released code consistently. Validation and test labels are never used to estimate μ_t or σ_t. The standardized per-task error and the reported metric are

$$\text{StdMAE}_t = \frac{\text{MAE}_t}{\sigma_t}, \qquad \text{Standard Macro-MAE} = \frac{1}{T} \sum_{t=1}^{T} \frac{\text{MAE}_t}{\sigma_t} = \frac{1}{T\, N_{\text{te}}} \sum_{t=1}^{T} \sum_{i \in \text{test}} \frac{|\hat{y}_{it} - y_{it}|}{\sigma_t}. \tag{15}$$

The same number is obtained by z-scoring both the labels and the predictions with (μ_t, σ_t) and then computing ordinary Macro-MAE in z-space: $\tilde{y}_{it} = (y_{it} - \mu_t)/\sigma_t$, $\hat{\tilde{y}}_{it} = (\hat{y}_{it} - \mu_t)/\sigma_t$. The metric is dimensionless; lower is better. If the model already emits predictions in the standardized label space, one must compute MAE directly and *must not* divide by σ_t a second time.

Subgraph matching: mAP and Hit@10. Let $\mathcal{Q} = \{q\}$ be the query set (motif templates g_j) and, for each query q, let $\mathcal{D}_q$ be the corpus of host graphs and $\mathcal{V}_q \subseteq \mathcal{D}_q$ the relevant hosts (q occurs in the host). A model returns a ranking $\pi_q(1), \dots, \pi_q(|\mathcal{D}_q|)$ of $\mathcal{D}_q$ in decreasing order of the containment score. Write $\text{rel}_q(k) = \mathbb{1}[\pi_q(k) \in \mathcal{V}_q]$ and $\text{P}\,@k(q) = \frac{1}{k} \sum_{\ell=1}^{k} \text{rel}_q(\ell)$. Average Precision and mean Average Precision are

$$\text{AP}(q) = \frac{1}{|\mathcal{V}_q|} \sum_{k=1}^{|\mathcal{D}_q|} \text{P}\,@k(q) \cdot \text{rel}_q(k), \qquad \text{mAP} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \text{AP}(q). \tag{16}$$

Hit@10 is the fraction of queries for which at least one relevant host appears in the top 10,

$$\text{Hit@10} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}\Big[\{\pi_q(1), \dots, \pi_q(10)\} \cap \mathcal{V}_q \neq \emptyset\Big]. \tag{17}$$

Both metrics are higher-is-better. Queries with $\mathcal{V}_q = \emptyset$ are skipped. For `GMT` the score is $\text{sim}(s_j, \mathcal{T}(\mathcal{G}))$; neural matchers use their published pairwise relevance score.

D.4 Implementation details

The tokenizer is trained once per dataset and then frozen. A standard Transformer is trained on the resulting token sequence; we do not scale the backbone to obtain the reported numbers. The downstream Transformer uses $d_{\text{model}} = 256$, 4 layers, 8 heads, and $d_{\text{ff}} = 512$. Motif mining (Table 6) and fine-tuning hyperparameters (Table 7) are set per dataset. All runs use AdamW. The learning-rate schedule is constant except on IMDB-B and QM8, which use cosine decay.

D.5 Construction of $\mathcal{P}^{(k)}$

We first tokenize the graphs with each of the four QNT tokenizers and feed the resulting tokens into a Transformer of a shared architecture, which is trained to produce a graph representation for every graph. We then compute cosine similarities among these representations: for N graphs this yields an $N \times N$ similarity matrix, and we take the k distinct graph pairs with the highest similarities as $\mathcal{P}^{(k)}$. We refer to pairs that are close in this representation space yet have a large GED as *spuriously similar graph token sets*; Inter GTC is the cause of this spurious similarity. To fairness comparison, all of them are selected on test set.

D.6 Training protocol

Table 8 records the training pipeline of every method in Tables 1 and 2. All tokenizers are offline and frozen before downstream training. Every method that has a pretraining stage is pretrained on the training split of the target dataset. In particular, `GraphGPT` is pretrained on that same training split; we do not load its official cross-dataset checkpoint.

Table 6: Motif mining configurations. The reported number of motifs is the number of retained query motifs used by the model.

Dataset	# Motifs	Motif size range	Minimum support
HIV	1024	4–10	512
BACE	256	4–8	64
BBBP	256	4–10	64
IMDB-B	32	4–8	64
SIDER	256	4–10	64
Tox21	512	4–10	128
ZINC	512	4–10	128
ESOL	128	4–8	64
AQSOL	512	4–10	128
Dielectric	512	4–6	291
QM9	1024	4–10	1024
QM8	32	4–10	256

Table 7: Hyperparameters settings for transformer.

Dataset	LR	Batch size	Weight decay	Dropout
HIV	6.0×10^{-4}	128	1.0×10^{-5}	0.05
BACE	8.0×10^{-4}	256	1.0×10^{-5}	0.05
BBBP	1.7×10^{-3}	128	1.0×10^{-5}	0.00
IMDB-B	4.0×10^{-4}	32	1.0×10^{-5}	0.05
SIDER	5.2×10^{-4}	128	1.0×10^{-5}	0.20
Tox21	8.0×10^{-4}	128	1.0×10^{-5}	0.10
ZINC	2.0×10^{-4}	128	1.0×10^{-5}	0.00
ESOL	2.5×10^{-3}	128	1.0×10^{-5}	0.25
AQSOL	2.0×10^{-4}	128	1.0×10^{-5}	0.10
Dielectric	1.0×10^{-3}	128	1.0×10^{-5}	0.15
QM9	1.4×10^{-3}	128	1.0×10^{-5}	0.05
QM8	9.0×10^{-4}	128	0	0.025

E EXPERIMENTAL RESULTS

E.1 SUBGRAPH MATCHING

Table 9 asks whether a frozen motif token already records the occurrence $S_{\mathcal{G}}$ defined in §3.1. Neural matchers still align each query–host pair at test time. `GMT` ranks hosts by $\mathrm{sim}(\vec{\mathbf{g}}_j, T(\mathcal{G}))$ and does not emit $\mathcal{B}$. It attains the best mAP on ten of eleven datasets and Hit@10 of at least 0.99 except on IMDB-B (0.93). Thus whether g_j occurs in $\mathcal{G}$ can be read from the reusable embeddings, without solving the NP-complete matching problem at test time. This is the precondition for using motif tokens as structural context against Inter GTC: graphs that collide in a node-token bag can still be separated if their motif inventories differ, and those inventories are already stored in $\{\vec{\mathbf{g}}_j\}$.

E.2 NESTED OBSERVATIONS ON QUANTIZED NODE TOKENIZERS

A quantized node tokenizer represents a graph by an unordered bag of local codes. We test whether appending motif tokens already improves that bag, without training a new tokenizer. The comparison is the empirical reason we move from QNT to a motif-aware observation.

The four QNTs are `Mole-BERT`, `NodeID`, `GQT`, and `VQGraph`. Each is trained with its official recipe and then frozen. For the same graph $\mathcal{G}$ we compare

$$\mathcal{T}_{\mathrm{N}}(\mathcal{G}) = \{z_v\}_{v\in\mathcal{V}}, \qquad \mathcal{T}_{\mathrm{M}}(\mathcal{G}) = \{z_v\}_{v\in\mathcal{V}} \cup \{s_j\}_{j\in S_{\mathcal{G}}}. \tag{18}$$

$\mathcal{T}_{\mathrm{N}}$ keeps only the QNT node tokens. $\mathcal{T}_{\mathrm{M}}$ appends a motif token s_j for every mined template that occurs in $\mathcal{G}$; g_j is encoded by the *same* frozen GNN checkpoint that produced $\{z_v\}$ and then pooled, so no extra encoder is trained.

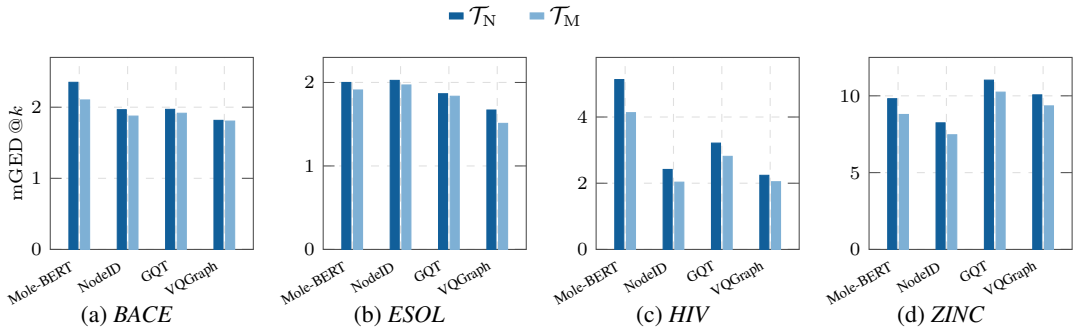
Table 8: Training protocol for all methods in the main tables. ✓ = yes, × = no, – = not applicable. "Unified ckpt" is defined only for methods with pretraining and asks whether an official or cross-dataset checkpoint is reused.

Method	Pretrain	End-to-end	Unified ckpt	Pretrain data	Fine-tune updates	Backbone
GCN	×	✓	–	–	all	GCN
GIN	×	✓	–	–	all	GIN
GAT	×	✓	–	–	all	GAT
GraphGPS	×	✓	–	–	all	GPS
Exphormer	×	✓	–	–	all	Exphormer
GraphTrans	×	✓	–	–	all	GNN+TF
TokenGT	×	✓	–	–	all	TokenGT
SimSGT	✓	×	×	target train	TF+head	TF 256/4/8
GraphGPT	✓	×	×	target train	BERT+head	BERT
GraphTokenizer	✓	×	×	target train	BERT+head	BERT
G^{2}PM	✓	×	×	target train	TF+head	TF 256/4/8
GQT	✓	×	×	target train	TF+head	TF 256/4/8
NodeID	✓	×	×	target train	MLP	MLP
Mole-BERT	✓	×	×	target train	MLP	MLP
VQGraph	✓	×	×	target train	MLP	MLP
GMT	✓	×	×	target train	TF+head	TF 256/4/8

Table 9: Subgraph matching retrieval. We report mAP and Hit@10 (↑). Best results are in **bold** and second-best are underlined. "–" denotes a result that is not reported.

	SIDER		BACE		ZINC		QM8		HIV		ESOL		QM9		Tox21		AQSOL		BBBP		IMDB-B	
Model	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10	mAP	Hit@10
NC-Iso	0.35	0.92	0.62	0.96	0.43	0.92	0.32	0.90	0.31	0.90	0.29	0.68	**0.79**	0.99	0.30	0.76	0.30	0.80	0.30	0.73	0.89	0.92
NeuroMatch	0.42	0.95	0.71	0.99	0.35	0.82	0.37	0.85	0.25	0.91	0.31	0.62	0.70	0.99	0.25	0.72	0.23	0.68	0.43	0.81	0.87	**0.93**
IsoNet	0.42	0.73	0.68	0.92	0.43	0.87	0.38	0.93	0.33	0.78	0.39	0.79	0.62	0.97	0.35	0.82	0.40	0.80	0.37	0.70	0.80	0.85
IsoNet++	0.44	0.81	0.79	0.98	0.47	0.91	0.39	0.94	0.37	0.86	0.48	0.90	–	–	0.38	0.87	0.42	0.78	0.48	0.85	0.81	0.83
GMT	**0.89**	**1.00**	**0.91**	**1.00**	**0.81**	**1.00**	**0.68**	**0.99**	**0.74**	**1.00**	**0.91**	**1.00**	0.73	**1.00**	**0.82**	**1.00**	**0.81**	**0.99**	**0.89**	**1.00**	**0.91**	**0.93**

Both observations are fed to a shared Transformer with the same architecture as GMT ($d_{\text{model}} = 256$, 4 layers, 8 heads, $d_{\text{ff}} = 512$). A graph embedding is obtained by sum pooling. Figure 6 and Table 10 report mGED @k and the downstream metrics already used in the main experiments.



(a) *BACE* (b) *ESOL* (c) *HIV* (d) *ZINC*

Figure 6: mGED @k ($k = 200$; lower is better) of four frozen QNTs under $\mathcal{T}_{\text{N}}$ and $\mathcal{T}_{\text{M}}$.

On all sixteen tokenizer–dataset pairs, mGED @$k(\mathcal{T}_{\text{N}})$ > mGED @$k(\mathcal{T}_{\text{M}})$: embedding neighbors become structurally closer once motif tokens are added. The downstream table moves in the same direction on fifteen of the sixteen cells (the exception is BACE with NodeID, 0.7331 → 0.7304). The gain is largest on structure-sensitive regression (ZINC with Mole-BERT: MAE 0.4867 → 0.3257, matching the mGED @k drop 9.812 → 8.786). Classification gains are smaller, which is consistent with the smaller mGED @k gaps on BACE relative to HIV and ZINC. A lower mGED @k is therefore not an isolated retrieval statistic: when nearest neighbors in token space are also nearer in GED, the same embedding yields a stronger graph-level predictor.

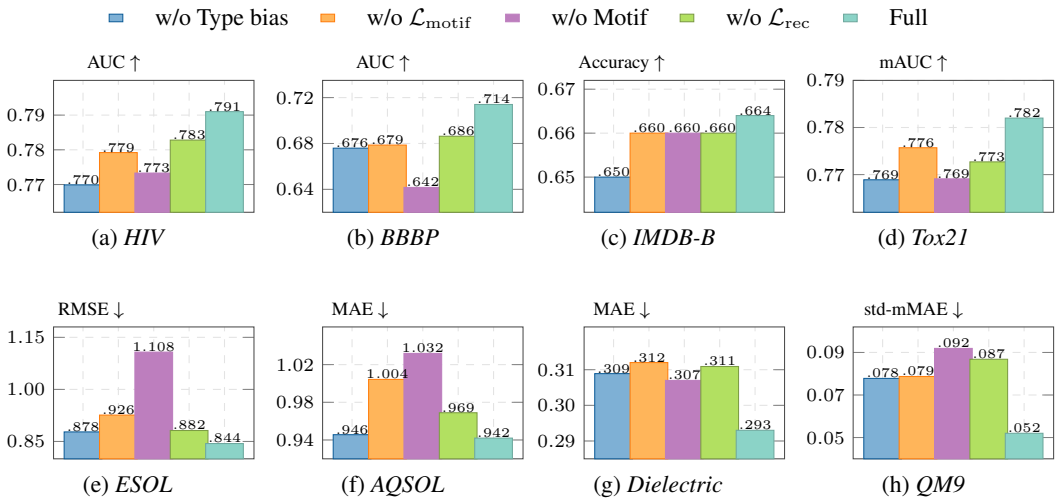
$\mathcal{T}_{\text{N}}$ is an unordered bag of local codes, so two graphs can share the same node-token multiset and still differ in which subgraphs they contain. $\mathcal{T}_{\text{M}}$ writes those recurring templates into the observation.

Table 10: Downstream results of four QNTs under $\mathcal{T}_{\mathrm{N}}/\mathcal{T}_{\mathrm{M}}$. Best of the two observations is in **bold**.

Tokenizer	BACE ↑	ESOL ↓	HIV ↑	ZINC ↓
`Mole-BERT`	0.7213/**0.7331**	1.0023/**0.9986**	0.7445/**0.7458**	0.4867/**0.3257**
`NodeID`	**0.7331**/0.7304	1.4887/**1.3340**	0.7228/**0.7324**	0.3884/**0.3145**
`GQT`	0.7235/**0.7345**	1.7089/**1.2234**	0.6989/**0.7145**	0.5114/**0.3215**
`VQGraph`	0.7724/**0.7845**	0.9758/**0.9235**	0.7558/**0.7601**	0.3882/**0.3215**

Table 11: Corpus token count of six serializations after BPE, versus the unordered token set of `GMT`. The shortest serialization in each row is underlined. The last column is that length divided by `GMT`.

Dataset	FEuler+BPE	Eulerian+BPE	CPP+BPE	FCPP+BPE	DFS+BPE	BFS+BPE	`GMT`	Shortest / `GMT`
QM8	607,363	630,088	533,426	532,301	651,078	765,812	**220,641**	**2.41**×
ZINC	2,571,032	2,653,570	2,198,317	2,198,816	2,448,087	3,368,581	**337,717**	**6.51**×
Dielectric	5,988,952	6,125,179	3,536,500	3,529,064	1,966,618	1,815,153	**101,973**	**17.80**×
BACE	509,252	524,990	458,282	458,282	565,454	881,985	**66,696**	**6.87**×



(a) *HIV* (b) *BBBP* (c) *IMDB-B* (d) *Tox21*

(e) *ESOL* (f) *AQSOL* (g) *Dielectric* (h) *QM9*

Figure 7: Ablation study on the remaining datasets. Variants follow Figure 3: w/o Type bias, w/o $\mathcal{L}_{\mathrm{motif}}$ (random motif GINE), w/o Motif, w/o $\mathcal{L}_{\mathrm{rec}}$, and Full.

The QNT stays frozen; only the token set changes. The consistent gain is therefore not an artifact of retraining the tokenizer, and it is the observation that motivates adding motif tokens in `GMT`.

E.3 Token length against graph serialization

Serialization keeps relative node positions by writing a walk (Hierholzer & Wiener, 1873; Tarjan, 1972). A walk that covers the edges must revisit vertices, so the string is already longer than n. The string also depends on the start: almost every start yields a different sequence, and a serialization tokenizer therefore stores several walks per graph. `GraphGPT` (Zhao et al., 2025) and `GraphTokenizer` (Guo et al., 2026) use Eulerian circuits; `G`^{2}`PM` (Wang et al., 2025b) serializes walk patterns by BFS. Mature serialization tokenizers already apply BPE (Sennrich et al., 2015) to shrink that walk and thereby address the length problem (Guo et al., 2026). For a fair comparison we therefore measure `GMT` against FEuler, Eulerian, CPP, FCPP, DFS, and BFS after BPE.

Table 11 reports a single walk per graph after BPE, which is already a lower bound on the serialization corpus. Even then the shortest walk is 2.41× longer than `GMT` on QM8 and 17.80× longer on Dielectric. This length gap is the basic advantage of a motif-aware token set over a serialized chain.

E.4 Ablation studies

Figure 7 repeats the five variants of Figure 3 on the eight datasets omitted from the main text. The top row is classification (ROC-AUC or accuracy) and the bottom row is regression (RMSE or MAE).

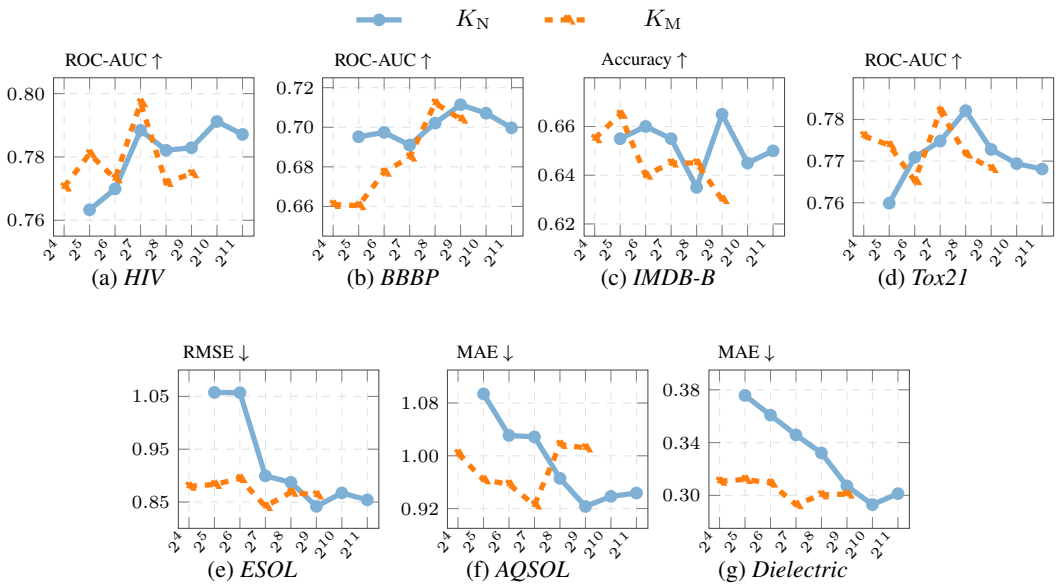


Figure 8: Sensitivity to codebook size on the remaining datasets.

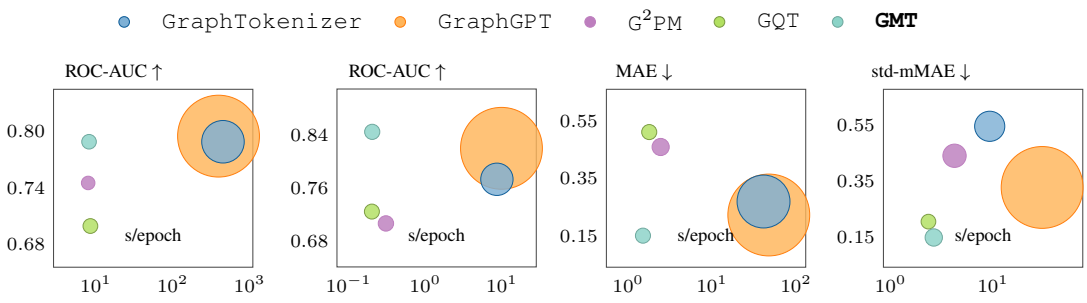
Full is best on every panel except IMDB-B, where several variants are nearly tied. Removing motif tokens hurts structure-sensitive regression most: the ESOL RMSE rises from 0.844 to 1.108 and the QM9 MAE from 0.052 to 0.092. Replacing the matching encoder by a random-motif GINE (w/o $\mathcal{L}_{\text{motif}}$) lands between Full and w/o Motif, so an untrained motif encoder is not enough. Dropping motif-conditioned pretraining (w/o $\mathcal{L}_{\text{rec}}$) and the type-pair bias each give smaller, dataset-dependent changes; the bias is again not the source of the gains. The eight panels therefore agree with the main-text conclusion: motif tokens help, and they help more when the motif–graph association is learned and then read in reconstruction.

E.5 Hyperparameter sensitivity

Figure 8 repeats the K_N / K_M sweep of Figure 4 on the seven datasets omitted from the main text, and the two observations in Sec. 3.2 still hold. A small allocated K_N tightens Intra GTC: the ESOL RMSE falls from 1.058 at K_N=32 to 0.842 at K_N=512, AQSOL from 1.094 to 0.923, and Dielectric from 0.376 to 0.307 (then 0.293 at K_N=1024). Classification moves less. HIV rises from 0.763 to 0.791 by K_N=1024, Tox21 peaks at 0.782 when K_N=256, and BBBP only from 0.695 to 0.711; IMDB-B stays inside a 0.03 band. Raising K_N to 2048 never improves over the best smaller setting and slightly hurts HIV, BBBP, Tox21, AQSOL, and Dielectric, so $\widetilde{K}_N$ does not grow without bound in the allocated size. The K_M curves are flatter, and a larger motif vocabulary is not always better. ESOL stays between 0.842 and 0.894; Dielectric between 0.293 and 0.312. HIV peaks at K_M=128 (0.797) and then drops; AQSOL is best at K_M=128 (0.925) and rises above 1.01 at $K_M \geq 256$; IMDB-B falls from 0.665 at K_M=32 to 0.630 at K_M=512 on only 1,000 graphs. BBBP is the exception that still needs some motif capacity (0.661 at K_M=16 vs. 0.712 at K_M=256), but it also saturates. The motif codebook therefore only has to cover the small set of key templates at that graph scale; extra codes do not relax Inter GTC further and can overfit.

E.6 Backbone Efficiency

Figure 9 reports downstream fine-tuning cost against the main-table scores on HIV, BACE, ZINC, and QM8. The comparison is restricted to the five baselines that are graph tokenizers and whose downstream backbone is a Transformer: the serialization methods `GraphGPT` and `GraphTokenizer`, the walk-pattern tokenizer `G`^{2}`PM`, the quantized-node tokenizer `GQT`, and `GMT`. Message-passing GNNs and Graph Transformers never compile the graph into a token sequence, and the remaining QNTs (`NodeID`, `Mole-BERT`, `VQGraph`) read pooled codes with an MLP rather than a Transformer, so their wall-clock and memory are not comparable on this axis. We time only



(a) *HIV* (b) *BACE* (c) *ZINC* (d) *QM8*

Figure 9: Downstream fine-tuning trade-off. The x-axis is seconds per epoch (log scale) and the y-axis is the main-table metric. Within each panel, marker diameter grows with peak GPU memory; sizes are not comparable across panels.

the downstream Transformer (one warmup epoch discarded, then the mean of three epochs); marker diameter is peak GPU memory in that window and is scaled independently in each panel.

Serialization sits on the far right: a long Eulerian or BPE chain makes attention quadratic in path length, so each epoch is slow and the memory footprint is large. `GQT` and `G²PM` stay on the left with short sequences, but the accuracy gap is large—especially on regression. `GMT` uses $L = n + m$ and therefore occupies the same cheap region, while matching or beating the serialization scores on every panel. On HIV, `GraphGPT` leads by only 0.006 AUC, yet one `GMT` epoch is about $48\times$ faster and uses about $1/38$ of its GPU memory (and is about $55\times$ faster and $16\times$ lighter than `GraphTokenizer` at the same AUC). On BACE, `GMT` is best and is still about $49\times$ faster and $29\times$ more memory-efficient than `GraphGPT`, and about $42\times$ faster and $8\times$ lighter than `GraphTokenizer`. On ZINC, `GMT` cuts the MAE of `GraphGPT` by about one third while being about $32\times$ faster and $30\times$ lighter; `GQT` matches that cost but its MAE is more than $3\times$ worse. QM8 repeats the pattern: `GMT` is best, about $11\times$ faster and $15\times$ lighter than `GraphGPT`, at a cost close to `GQT` whose MAE is about $1.4\times$ higher. Across all four datasets, `GMT` is the point that is both accurate and cheap: serialization buys little extra quality at a tens-of-times cost, and compact QNT or walk tokens stay cheap but leave a large quality gap.