Learning Graph Quantized Tokenizers for Transformers

Anonymous authors
Paper under double-blind review

Abstract

Transformers serve as the backbone architectures of Foundational Models, where a domain-specific tokenizer helps them adapt to various domains. Graph Transformers (GTs) have recently emerged as a leading model in geometric deep learning, outperforming Graph Neural Networks (GNNs) in various graph learning tasks. However, the development of tokenizers for graphs has lagged behind other modalities, with existing approaches relying on heuristics or GNNs co-trained with Transformers. To address this, we introduce GQT (**G**raph **Q**uantized **T**okenizer), which decouples tokenizer training from Transformer training by leveraging multi-task graph self-supervised learning, yielding robust and generalizable graph tokens. Furthermore, the GQT utilizes Residual Vector Quantization (RVQ) to learn hierarchical discrete tokens, resulting in significantly reduced memory requirements and improved generalization capabilities. By combining the GQT with token modulation, a Transformer encoder achieves state-of-the-art performance on 16 out of 18 benchmarks, including large-scale homophilic and heterophilic datasets.

1 Introduction

Following the success of Transformers (Vaswani et al., 2017) in natural language processing (Devlin et al., 2019; Brown et al., 2020) and computer vision (Dosovitskiy et al., 2021), Graph Transformers (GTs) (Dwivedi & Bresson, 2020; Ying et al., 2021a; Rampášek et al., 2022; Ma et al., 2023; Shirzad et al., 2023; Kong et al., 2023b; Chen et al., 2023; Wu et al., 2022b) have emerged as strong models in geometric deep learning. Unlike message-passing Graph Neural Networks (GNNs), which rely on strong locality inductive biases (Battaglia et al., 2018; Veličković et al., 2018; Hou et al., 2020; Hamilton et al., 2017a; Kipf & Welling, 2017), GTs are inherently more expressive due to their ability to capture long-range interactions between nodes (Ma et al., 2023). This is particularly beneficial in heterophilous settings where local alignment does not hold (Fu et al., 2024). GTs possess an expressive power at least equivalent to the 2-Weisfeiler-Lehman (WL) isomorphism test (Kim et al., 2022), which is sufficient for most real-world tasks (Zopf, 2022). This surpasses the expressive power of message-passing GNNs, which are limited to the 1-WL test (Ying et al., 2021a). Furthermore, a Transformer with sufficient attention heads can match or exceed the expressive power of a second-order invariant graph network, outperforming message-passing GNNs (Kim et al., 2022). However, both GNNs and Transformers are susceptible to over-smoothing (Li et al., 2018; Zhou et al., 2021; Dovonon et al., 2024).

GTs require consideration of both graph structure and features, as nodes with identical features will otherwise be projected into the same representation regardless of their surrounding structures (Hoang et al., 2024). There are three general approaches to address this limitation (Hoang et al., 2024): (1) node feature modulation, which involves injecting structural information into the node features; (2) context node sampling, where a sampling strategy is used to construct a sequence over the neighbor nodes; and (3) modifying the architecture of a vanilla Transformer to directly incorporate structural biases. Given that Transformers are universal approximators of sequence-to-sequence functions (Yun et al., 2020) and considering the rapid developments in efficient implementation of multi-head attention (MHA) module (Dao et al., 2022a; Liu et al., 2024), which enables longer context sizes of up to million-scale tokens (Reid et al., 2024), a well-designed graph tokenizer can allow a vanilla Transformer model to efficiently process even large-scale graphs. Recent studies on applying Large Language Models (LLMs) to graph-related tasks have found that representing graphs through textual descriptions can lead to surprisingly strong performance gains that surpass those of GNNs, suggesting that vanilla Transformers are indeed capable of effectively learning graph structures (Ye et al., 2024; He et al., 2024). Nonetheless, LLMs are not inference-efficient, and hence our goal in this paper is to devise a lightweight and efficient graph tokenization strategy that enables vanilla Transformer encoders to learn graph structures effectively, without relying on LLMs.

Tokenizers typically employ self-supervised objectives to abstract data into a sequence of discrete tokens, enabling Transformers to learn representations across various modalities as a unified stream of data. This discretization is achieved through vector quantization techniques (Van Den Oord et al., 2017; Lee et al., 2022), which offer several benefits, including: (1) significantly reduced memory requirements, (2) improved inference efficiency, (3) allowing Transformers to focus on long-range dependencies rather than local information, and (4) the capacity to learn more high-level representations due to a compact latent space (Yuan et al., 2021; Yu et al., 2022). These advantages are particularly important in auto-regressive generative modeling, where quantized tokens allow Transformers to generate high-quality outputs in multiple modalities (Dubey et al., 2024; Lee et al., 2022; Dhariwal et al., 2020; Ramesh et al., 2021; Team, 2024). Despite its importance in other domains, tokenization remains under-explored for graph-structured data. To address this limitation, we propose the **Graph Quantized Tokenizer (GQT)**, a novel approach that learns a hierarchical sequence of tokens over graphs using self-supervised objectives tailored to graph-structured data. More specifically, our contributions are as follows:

- We propose a graph tokenizer that utilizes multi-task graph self-supervised objectives to train a graph encoder, enabling it to fully capture local interactions and allowing the Transformer to focus on long-range dependencies.
- Our approach adapts Residual Vector Quantization (RVQ) within the graph tokenizer to learn hierarchical discrete tokens, resulting in significantly reduced memory requirements and improved generalization capabilities.
- We introduce a novel combination of semantic edges and random walks to facilitate the Transformer's access to long-range interactions, and employ hierarchical encoding and gating mechanisms to modulate the tokens and provide informative representations to the Transformer.
- Through extensive experiments on both homophilic and heterophilic datasets, including large-scale benchmarks, we demonstrate that our proposed tokenizer enables a Transformer encoder to achieve state-of-the-art performance on 16 out of 18 benchmarks while substantially reducing the memory footprint of the embeddings.

2 Related Works

Graph Transformers (GTs) have shown promising performance on various graph learning tasks, surpassing GNNs on many benchmarks. GTs can be broadly categorized into two directions (Hoang et al., 2024; Müller et al., 2024): (1) modifying the vanilla Transformer architecture to incorporate structural inductive biases, or (2) encoding the input graph to make it compatible with the vanilla Transformer design. Early examples of the first approach include Graph Attention Network (Veličković et al., 2018), which uses an attention module to compute pairwise node attentions and masks the attention matrix based on connectivity information. Subsequent works have replaced the scaled-dot attention module with various structure-aware sparse attention modules (Rampášek et al., 2022; Bo et al., 2023; Ying et al., 2021a; Deng et al., 2024; Wu et al., 2023b; Liu et al., 2023a; Chen et al., 2022; Dwivedi & Bresson, 2020; Shirzad et al., 2023; Ma et al., 2023). Examples of the second approach include Graph Memory Network (Khasahmadi et al., 2020), which passes non-linear projections of node features and structural encoding to a Transformer-like model. Structural encoding methods, such as Laplacian eigenvectors or Random walk-based encoding (Dwivedi et al., 2022; Ma et al., 2023; Cantürk et al., 2024), allow injecting structural information directly into the node features. Another approach involves using GNNs to encode local structure along with node features, followed by passing the representation to vanilla Transformers to capture long-range dependencies. (Rong et al., 2020; Wu et al., 2021; Chen et al., 2023; 2022). Recent studies leverage LLMs, where graphs are represented through natural language expressions, and an LLM performs graph-related tasks through in-context learning, instruction tuning, or soft prompts (Fatemi et al., 2024; Ye et al., 2024; He et al., 2024). For a detailed survey on GTs, see (Müller et al., 2024; Hoang et al., 2024).

Graph Tokenization provides GTs with rich node tokens that encapsulate both structural and semantic information. Various approaches have been proposed to define these node tokens. TokenGT (Kim et al., 2022) treats nodes and edges as independent tokens defined by their features, type identifiers, and structural encodings. NAGphormer (Chen et al., 2023) represents each node as a set of L tokens, where the l^{th} token is the representation of the node from l^{th} hop aggregation. In contrast, GraphiT (Mialon et al., 2021) defines a node token as the concatenation of its feature and representation from a graph convolutional kernel network (GCKN). VCR-Graphormer (Fu et al., 2024) expands the notion of node tokens to include sequences comprising the node feature and features of semantically and community-related neighboring nodes. SGT (Liu et al., 2023b) is a non-parametric tokenizer designed for molecular tasks, which simplifies the tokenization process to a non-parametric graph operator without non-linearity, unlike motif-based tokenizers (Zhang et al., 2021; Jin et al., 2018) or GNN pre-training methods (Xia et al., 2023). NodePiece (Galkin et al., 2022) is a knowledge-graph tokenizer that represents a target node as a hash of its top-k closest anchors, their distances, and relational context. For a more detailed review see Müller et al. (2024). While vector quantization (VQ) (Van Den Oord et al., 2017; Lee et al., 2022) has been explored in other modalities, its application in graph learning is limited. Notable exceptions include VQGraph (Yang et al., 2024), which employs VQ for graph distillation, and NID (Luo et al., 2024a), which uses VQ to learn discrete node IDs for downstream prediction tasks.

3 Preliminaries

Messag-Passing GNNs. Let $\mathcal{G}$ denote the space of graphs. A graph g within this space is defined as $g = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_x}$ represents the node features of dimension d_x, and $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d_e}$ represents the edge features of dimension d_e. A message-passing GNN takes g as input and learns representations h_v^l for $v \in \mathcal{V}$ ($h_v^0 = x_v$) in each layer l as follows (Gilmer et al., 2017):

$$h_v^l = f_\theta^l \left(h_v^{l-1}, g_\phi^l \left(\left\{ \left(h_v^{l-1}, h_u^{l-1}, e_{uv} \right) | u \in \mathcal{N}_i(v) \right\} \right) \right) \tag{1}$$

where f_θ and g_ϕ are known as update (combine) and message (aggregate) functions, respectively, and $\mathcal{N}_i(v)$ denotes the set of immediate neighbors of the node v. With this representation, we can perform various tasks, including node classification as $\text{MLP}\left(h_v\right)$, edge prediction as $\text{MLP}\left(h_u \odot h_v\right)$, or graph classification as $\text{MLP}\left(\mathcal{R}\left(\{h_u | u \in \mathcal{V}\}\right)\right)$ where $\mathcal{R}$ is a pooling (readout) function.

Graph Transformers, on the other hand, first use a tokenizer $T_v = \mathcal{T}_\psi\left(\mathcal{N}(v)\right)$ to map each node $v \in \mathcal{V}$ into a sequence of tokens T_v by considering some notion of neighborhood $\mathcal{N}$. The simplest design is when $\mathcal{N}$ is zero-hop neighborhood (i.e., the node itself) and $\mathcal{T}_\psi$ is a node feature lookup function. The neighborhood $\mathcal{N}$ can be extended to include an ego network (Zhao et al., 2021) or top-k Random Walk based neighbors (Fu et al., 2024), and $\mathcal{T}_\psi$ can be enhanced to representations from a GNN (Chen et al., 2023). Once the tokens are computed, along with a node positional encoding function (PE), we can define the input to a Transformer as $h_v^0 = [T_v || \text{PE}\left(v\right)]$ and compute the representation in each layer l of a vanilla Transformer encoder as follows:

$$h_v^l = \text{LN}\left(\text{MHA}\left(\text{LN}\left(h_v^{l-1}\right)\right) + h_v^{l-1}\right) \tag{2}$$

$$h_v^l = h_v^l + \text{MLP}\left(h_v^l\right) \tag{3}$$

where LN and MHA are Layer Normalization and Multi-Head Attention modules, respectively. Similar to Transformer encoders in other modalities (Devlin et al., 2019; Dosovitskiy et al., 2021), we can append a special classification token, denoted as [CLS], to the input and use its representation to perform various classification tasks on the graph: $\text{MLP}\left(h_{[\text{CLS}]}\right)$. In this setting, the input for node classification is T_v, for link prediction is $[T_v || T_u]$, and for graph classification is $\left[T_v ||_{v \in \mathcal{V}}\right]$.

Vector Quantization projects embeddings $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ into a more compact space of codebooks $\mathbf{C} \in \mathbb{R}^{k \times d_c}$, where $k \ll n$. The codebooks can be learned by minimizing various objectives such as K-means clustering. The new representation of x_i is then computed as follows (Van Den Oord et al., 2017):

$$z(x_i) = c_k \quad \text{where} \quad k = \arg\min_j \|x_i - c_j\|_2^2 \tag{4}$$

Building upon this concept, RQ-VAE (Lee et al., 2022) extends VQ to a sequence of codebooks, where each consecutive codebook quantizes the residual error from the previous codebook, i.e.,

$r_i = z_i - c_k$. This hierarchical approach constructs a multi-level quantized representation, enhancing the overall quantization quality.

4 Self-Supervised Graph Tokenization

4.1 Tokenizer Properties

Our goal is to design a graph tokenizer that can learn to generate tokens that exhibit three key characteristics, which are essential for effective graph representation learning. These characteristics are as follows.

Local Interactions. The learned tokens should encapsulate local interactions, allowing the Transformer to focus on global dependencies. This is analogous to Vision Transformers (ViTs), where the Transformer attends to image patches instead of pixels, enabling efficient learning on abstract tokens (Dosovitskiy et al., 2021; Liu et al., 2021). To achieve a similar effect on graph-structured data, we leverage message-passing GNNs as the foundation of the tokenizer's encoder, capitalizing on their strong locality inductive bias to effectively capture local interactions in the representation space (Battaglia et al., 2018). Our design accommodates various GNN layer choices without constraints; for simplicity, we opt for the widely used Graph Attention Network (GAT) (Veličković et al., 2018) as our base graph encoder. The representation of node i in layer l is computed as:

$$h_i^l = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W} h_j^{(l-1)}\right), \quad \alpha_{ij} = \frac{\exp\left(\sigma\left(\mathbf{W}_2\left[\mathbf{W}_1 h_i^{(l-1)} \| \mathbf{W}_1 h_j^{(l-1)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\sigma\left(\mathbf{W}_2\left[\mathbf{W}_1 h_i^{(l-1)} \| \mathbf{W}_1 h_k^{(l-1)}\right]\right)\right)} \tag{5}$$

where σ is a non-linearity, and α_{ij} is the normalized attention score between two connected nodes i and j.

Memory Efficiency. The tokens should be compact to facilitate efficient memory usage, enabling the Transformer to perform efficient inference. To achieve this, we introduce a Residual-VQ (RVQ) (Lee et al., 2022) layer to quantize the GNN representations into a sequence of discrete tokens. Quantization not only helps with generalization due to its regularization effect but also significantly reduces memory usage. Using an RVQ with c codebooks (typically $c = \{2, \cdots, 8\}$), a graph with feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d_x}$ can be represented as $\mathbf{X}_Q \in \mathbb{N}^{N \times c}$ and codebook representation of $\mathbf{C} \in \mathbb{R}^{c \times K \times d_c}$, where c is the number of codebooks (i.e., levels of quantization), K is the codebook size, and d_c is the code dimension. To illustrate the benefits of this approach, consider a graph with 10^6 nodes and a feature dimension of 1024 ($\mathbf{X} \in \mathbb{R}^{10^6 \times 1024}$). Using an RVQ with 3 codebooks and a codebook size of 256, this graph can be represented as $\mathbf{X}_Q \in \mathbb{N}^{10^6 \times 3}$ plus $\mathbf{C} \in \mathbb{R}^{3 \times 256 \times 1024}$, resulting in a 270-fold reduction in required memory.

Robustness and Generalization. The tokens should be robust and generalizable. To achieve this, we rely on graph self-supervised learning. Self-supervised representations have been shown to be more robust to class imbalance (Liu et al., 2022) and distribution shift (Shi et al., 2023), while also capturing better semantic information (Assran et al., 2023) compared to representations learned through supervised objectives. Moreover, self-supervised graph representations have demonstrated superior performance on downstream tasks compared to representations learned in a fully supervised manner, indicating better generalization capabilities (Hu et al., 2020b; Sun et al., 2020; You et al., 2020; 2021; Hassani & Khasahmadi, 2020; Hou et al., 2022; Veličković et al., 2019; Zhu et al., 2020b; Thakoor et al., 2022). Additionally, multi-task learning with self-supervised objectives has been shown to achieve better performance on downstream tasks (Doersch & Zisserman, 2017; Ghiasi et al., 2021). To leverage these benefits, we propose training the GNN encoder with three self-supervised objectives. Unlike RQ-VAE (Lee et al., 2022), which uses reconstruction as its primary objective, we employ graph-specific objectives to capture the nuances of both structure and features within the tokens. Specifically, we use Deep Graph Infomax (DGI) (Veličković et al., 2019) and Graph Masked Auto-Encoder 2 (GMAE2) (Hou et al., 2023). DGI is a contrastive method that contrasts local (node) encoding with global (graph or sub-graph) encoding, whereas GMAE2 combines generative and distillation objectives to jointly reconstruct masked features and track teacher representations.

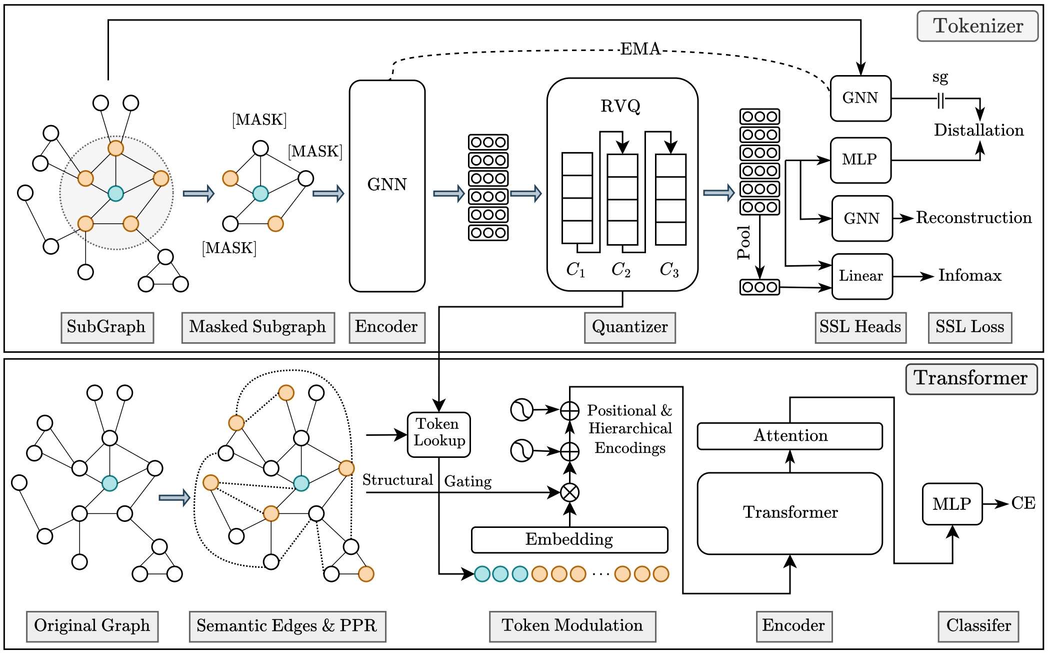

Figure 1: Overview of our proposed Graph Quantization Transformer (GQT) consisting of three main components: (1) a GNN to encode local interactions, (2) vector quantization for compact representation, and (3) generative and contrastive heads for robust representation learning. We also utilize a Transformers encoder to model long-range interactions. We augment the graph with semantic edges (dashed lines) and generate a sequence for each node based on Personalized PageRank scores. We then modulate the tokens through hierarchical encoding and structural gating, and feed them into the Transformer and aggregate the learned representations through an attention module before passing it to the classification head.

4.2 Training

The GNN encoder is trained through gradient descent to minimize a loss function comprising three terms, where β is the loss weight:

$$\mathcal{L} = \mathcal{L}_{\text{dgi}} + \mathcal{L}_{\text{gmae2}} + \beta\mathcal{L}_{\text{commit}} \tag{6}$$

The first term is the DGI objective, which maximizes mutual information (MI) between node representations and graph (sub-graph) representations, based on the Jensen-Shannon divergence between the joint and product of marginals as follows (Veličković et al., 2019):

$$\mathcal{L}_{\text{dgi}} = \mathbb{E}\left(\sum_{v\in g}\log\left(\mathcal{D}\left(h_v,h_g\right)\right) + \sum_{u\in\tilde{g}}\log\left(1-\mathcal{D}\left(\tilde{h}_u,h_g\right)\right)\right) \tag{7}$$

where h_u is the representation of node u, h_g is the patch (graph/sub-graph) representation that the node belongs to, $\mathcal{D}\left(.,.\right)$ is a discriminator computing the probability scores between local and global information, and $\tilde{g}$ is the corrupted version of the original graph providing negative examples. Following (Veličković et al., 2019), we define the discriminator as a bilinear layer $\mathcal{D}\left(h_u,h_g\right) = \sigma\left(h_u^T\mathbf{W}h_g\right)$, compute the global representation as a mean of node representations: $h_g = \frac{1}{|\mathcal{V}|}\sum\limits_{v\in g} h_v$, and define $\tilde{g}$ as a graph with the same structure but randomly shuffled features.

The second term is the GraphMAE2 objective (Hou et al., 2023), which combines the generative loss of GraphMAE (Hou et al., 2022) with the teacher-(noisy)student distillation loss of BGRL (Thakoor et al., 2022). This combination enables the model to avoid overfitting and learn more semantic representations. The GraphMAE2 loss is computed as follows:

$$\mathcal{L}_{\text{gmae2}} = \sum_{v\in\tilde{g}}\left(1-\frac{x_v^T.\tilde{h}_v}{\|x_v^T\|.\|\tilde{h}_v\|}\right)^{\gamma} + \lambda\sum_{v\in g}\left(1-\frac{h_v^T.\tilde{h}_v}{\|h_v^T\|.\|\tilde{h}_v\|}\right)^{\gamma} \tag{8}$$

where $\tilde{g}$ is the masked graph, $\tilde{h}_v$ is the node representation of a masked node learned by the noisy student, h_v is the corresponding node representation learned by the teacher over the original graph, and $\gamma \geq 1$ is a scaling factor. Note that the teacher's parameters are updated using an exponential moving average (EMA) of the noisy student's parameters.

The third term is the commitment loss, which encourages the representations to get close to their corresponding codebook embeddings within the RVQ layer. This loss is computed as:

$$\mathcal{L}_{\text{commit}} = \frac{1}{|\mathcal{V}|} \sum_{v \in g} ||h_v - \text{sg}\left[c_k\right]||_2 \tag{9}$$

where sg is the stop-gradient operator, and c_k is the representation of the codebook that h_v is assigned to (i.e., the centroid or prototype vector). Note that this loss only affects the node representations and does not update the codebooks.

To initialize and update the codebooks, we employ K-Means clustering and EMA with weight decay $\tau \in [0, 1]$, respectively. Specifically, the codebooks are updated as follows:

$$c_k^t = \tau c_k^{t-1} + (1-\tau) \frac{1}{|\mathcal{V}_k|} \sum_{v \in \mathcal{V}_k} h_v \tag{10}$$

where $\mathcal{V}_k$ is the set of nodes assigned to codebook c_k. This update rule allows the codebooks to adapt to the changing node representations while maintaining stability.

5 Graph Transformer

5.1 Sequence Generation

Once the tokenizer is trained, each node $v \in \mathcal{V}$ is mapped to a set of c tokens: $T_v = [t_1^v, \cdots, t_c^v] \in \mathbb{N}^c$, which compress information about local interactions. To enable the Transformer to capture long-range interactions, the input should consist of a sequence of tokens from nodes that are likely to have long-range dependencies. To facilitate this, we first augment the graph with *semantic edges* denoted as $\mathcal{E}_s$, which are computed as follows:

$$\mathcal{E}_s = \left\{ e_{u,v} \mid \underset{u \in \mathcal{V}}{\arg \operatorname{topk}} \operatorname{sim}\left(f\left(x_u\right), f\left(x_v\right)\right) \forall v \in \mathcal{V} \right\} \tag{11}$$

where $\text{sim}(\cdot, \cdot)$ denotes the similarity function, x_u is the feature vector of node u, and f is a projection function. We use cosine similarity as the similarity function and principal component analysis (PCA) as the projection function. This semantic edge augmentation effectively creates sparse edges between each node and its k-nearest neighbors in the feature space, enhancing the model's ability to recognize and utilize significant long-range dependencies.

We then merge the semantic edges with the original graph edges and use Personalized PageRank (PPR) to generate a sequence per node. A PPR vector for a node u captures the relative importance of other nodes with respect to node u by exploring the graph structure through iterative random walks:

$$r = \alpha \mathbf{P} r + (1-\alpha) q \tag{12}$$

where $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{n \times n}$, q is a stochastic personalized vector, r is the stationary distribution of random walks, and α is a damping factor.

Using PPR enriches the sequence with information beyond local interactions, allowing the Transformer to access potential long-range dependencies. We construct the sequence S_v for each node v as follows:

$$S_v = \left[T_v \| T_u \|_{u \in \arg \operatorname{topk} \text{PPR}(v, \mathcal{E} \cup \mathcal{E}_s)}\right] \tag{13}$$

where $S_v = [t_1^v \cdots t_c^v \mid t_1^{u_1} \cdots t_c^{u_1} \mid \cdots \mid t_1^{u_k} \cdots t_c^{u_k}]$ is the sequence of sorted integer tokens with length $c \times (k+1)$, based on the PPR scores for node u. Note that the computation of semantic edges and PPR sequences is performed only once as a pre-processing step, which reduces computational overhead during training.

5.2 Token Modulation

There are $c \times K$ possible integer tokens in total, where c is the number of codebooks and K is the codebook size. We randomly initialize an embedding matrix $\mathbf{X}_T \in \mathbb{R}^{c \times K \times d_x}$, which is trained end-to-end with the Transformer. To further enrich the token representation, we introduce an additional token to each node by aggregating the embeddings of its assigned codebooks:

$$h_c^v = \sum_{i=1}^{c} \mathbf{C}[i, t_i^v] \tag{14}$$

We found that adding this explicit aggregated token leads to better performance compared to initializing $\mathbf{X}_T$ with $\mathbf{C}$. The input representation of the sequence for node v is then defined as:

$$S_v = \left[\mathbf{X}_T[i, t_i^v] \overset{c}{\underset{i=1}{\Big\|}} h_c^v \,\|\, \mathbf{X}_T[i, t_i^{u_1}] \overset{c}{\underset{i=1}{\Big\|}} h_c^{u_1} \,\| \cdots \| \mathbf{X}_T[i, t_i^{u_k}] \overset{c}{\underset{i=1}{\Big\|}} h_c^{u_k} \right] \tag{15}$$

This representation combines the individual token embeddings with the aggregated codebook embeddings, providing a more comprehensive and nuanced input to the Transformer.

In order to provide the Transformer with the global structural importance scores of the nodes within the sequence with respect to the target node, we introduce a gating mechanism over the input token embeddings as follows:

$$S_v = S_v \odot \text{Softmax}\left(\text{topk}\,\text{PPR}\left(v, \mathcal{E} \cup \mathcal{E}_s\right)\right) \tag{16}$$

where we first apply a softmax function with temperature $\tau = 1$ to normalize the PPR scores, and then multiply each node token's representation by its corresponding normalized score.

We also introduce two trainable positional encodings to the input tokens. The first positional encoding enables the Transformer to distinguish between tokens from different nodes, while the second encoding, referred to as hierarchical encoding, allows the Transformer to recognize the hierarchy level of each token within the codebooks. We randomly initialize the positional encodings $\mathbf{PE} \in \mathbb{R}^{(k+1) \times d_x}$ and $\mathbf{HE} \in \mathbb{R}^{c \times d_x}$ and sum them with the encoding of their corresponding token. For example, the final encoding of the token j of the node i within the sequence is computed as: $x = \mathbf{X}_T[j, t_j^{u_i}] + \mathbf{PE}[i] + \mathbf{HE}[j]$. Note that we did not use any structural encoding, such as Laplacian eigenvectors, as our experiments did not show any significant benefits from including them.

5.3 Transformer Encoder & Classification Head

We use l layers of standard Transformer encoder with flash attention (Dao et al., 2022b) to generate contextual representations per token in the sequence: $\mathbf{H}^{(l)} \in \mathbb{R}^{(c+1) \times (k+1) \times d_h}$. We then aggregate the token representations for j-th node in the sequence by summing along the token dimension:

$$\mathbf{H}_{v_j} = \sum_{i=1}^{c+1} \mathbf{H}^{(l)}[i, j] \in \mathbb{R}^{(k+1) \times d_h} \tag{17}$$

To obtain a single representation for the entire sequence, We further aggregate the representation using a linear attention layer:

$$h = \sum_{i=1}^{k+1} \alpha_i h_i \quad \text{where} \quad \alpha_i = \frac{\exp(\mathbf{W} h_i)}{\sum_j \exp(\mathbf{W} h_j)} \tag{18}$$

We feed the resulting representation into a fully-connected classifier and train the model end-to-end with cross-entropy loss. Note that during inference, only the Transformer and classifier are utilized, as the tokenizer is pretrained and the sequences are pre-computed. Furthermore, since we only require discrete tokens and codebook embeddings, our approach allows for efficient memory usage, regardless of graph size enable efficient training and inference on large-scale graphs.

6 Experiments

We comprehensively evaluate GQT on both medium-scale and large-scale node classification tasks, encompassing both homophilous and heterophilous settings across 18 datasets. Homophilous graphs

Table 1: Mean node classification performance on medium-scale homophilous datasets over five runs.

		CoraFull	**CiteSeer**	**PubMed**	**Computer**	**Photo**	**CS**	**Physics**	**WikiCS**
Dataset	#Nodes	19,793	3,327	19,717	13,752	7,650	18,333	34,493	11,701
	#Edges	126,842	4,522	88,651	491,722	238,163	163,788	495,924	216,123
	#Features	8,710	3,703	500	767	745	6,805	8,415	300
	#Classes	70	6	3	10	8	15	5	10
	Measure	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑
GNN	GCN	61.76±0.14	76.50±1.36	86.54±0.12	89.65±0.52	92.70±0.20	92.92±0.12	96.18±0.07	77.47±0.85
	GAT	64.47±0.18	76.55±1.23	86.32±0.16	90.78±0.13	93.87±0.11	93.61±0.14	96.17±0.08	76.91±0.82
	APPNP	65.16±0.28	76.53±1.16	88.43±0.15	90.18±0.17	94.32±0.14	94.49±0.07	96.54±0.07	78.87±0.11
	GPRGNN	67.12±0.31	77.13±1.67	89.34±0.25	89.32±0.29	94.49±0.14	95.13±0.09	96.85±0.08	78.12±0.23
	GraphSAINT	67.85±0.21	–	88.96±0.16	90.22±0.15	91.72±0.13	94.41±0.09	96.43±0.05	–
	GraphSAGE	–	75.58±1.33	87.48±0.38	91.20±0.29	94.59±0.14	93.91±0.13	96.49±0.06	74.77±0.95
	PPRGo	63.54±0.25	–	87.38±0.11	88.69±0.21	93.61±0.12	92.52±0.15	95.51±0.08	78.12±0.23
	GRAND+	71.37±0.11	–	88.64±0.09	88.74±0.11	94.75±0.12	93.92±0.08	96.47±0.04	–
GT	GT	61.05±0.38	–	88.79±0.12	91.18±0.17	94.74±0.13	94.64±0.13	97.05±0.05	–
	Graphormer	OOM	–	OOM	OOM	92.74±0.14	94.64±0.13	OOM	–
	SAN	59.01±0.34	–	88.22±0.15	89.93±0.16	94.86±0.10	94.51±0.15	OOM	–
	GraphGPS	55.76±0.23	76.99±1.12	88.94±0.16	OOM	95.06±0.13	93.93±0.15	OOM	78.66±0.49
	GOAT	–	76.89±1.19	86.87±0.24	90.96±0.90	92.96±1.48	94.21±0.38	96.24±0.24	77.00±0.77
	NodeFormer	–	76.33±0.59	89.32±0.25	86.98±0.62	93.46±0.35	95.64±0.22	96.45±0.28	74.73±0.94
	DIFFormer	–	76.72±0.68	89.51±0.67	91.99±0.76	95.10±0.47	94.78±0.20	96.60±0.18	73.46±0.56
	NAGphormer	71.51±0.13	77.42±1.41	89.70±0.19	91.22±0.14	95.49±0.11	95.75±0.09	97.34±0.03	77.16±0.72
	Exphormer	69.09±0.72	76.83±1.24	89.52±0.54	91.59±0.31	95.27±0.42	95.77±0.15	97.16±0.13	78.54±0.49
	VCR-Graphormer	71.67±0.10	–	89.77±0.15	91.75±0.15	**95.53±0.14**	95.37±0.04	97.34±0.04	–
	GQT (ours)	**71.81±0.21**	**77.84±0.94**	**90.14±0.16**	**92.05±0.16**	95.35±0.18	**96.11±0.09**	**97.53±0.06**	**79.65±0.52**

are characterized by nodes with similar classes being connected to each other, whereas heterophilous graphs exhibit connections between nodes with different classes. Following the convention of most existing works on GTs, we focus on node classification tasks in our experiments. However, as discussed in Section 3, our model can be easily extended to graph classification and link prediction tasks. For each evaluation scenario, we adhere to the established experimental protocols from previous works to ensure fair comparisons. Detailed descriptions of the datasets are provided in Appendix A and detailed experimental setup and hyperparameters are provided in Appendix B.

6.1 Comparison with State-of-the-Art

Homophilous Node Classification. To evaluate the performance on medium-scale homophilous graphs, we use eight benchmark datasets including CoraFull (Bojchevski & Günnemann, 2017), CiteSeer, and PubMed (Yang et al., 2016), Amazon Computers, Amazon Photos, Co-author CS, and Co-author Physics (Shchur et al., 2018), as well as WikiCS (Mernyei & Cangea, 2020). We compare our results with a comprehensive set of baselines, including four traditional GNNs: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2019), APPNP (Gasteiger et al., 2018), and GPRGNN (Chien et al., 2020); four scalable GNN variants including GraphSAINT (Zeng et al., 2019), GraphSAGE (Hamilton et al., 2017b), PPRGo (Bojchevski et al., 2020), and GTAND+(Feng et al., 2022); four standard GTs including GT (Dwivedi & Bresson, 2020), Graphormer (Ying et al., 2021b), SAN (Kreuzer et al., 2021), and GraphGPS (Rampášek et al., 2022); and six state-of-the-art scalable GTs including GOAT (Kong et al., 2023a), NodeFormer (Wu et al., 2022a), DiffFormer (Wu et al., 2023a), NAGphormer (Chen et al., 2023), Exphormer (Shirzad et al., 2023), and VCR-Graphormer (Fu et al., 2024). The baseline performance is reported from existing works (Wu et al., 2023b; Luo et al., 2024a; Fu et al., 2024). As shown in Table 1, GQT outperforms the baseline GNN and GT models on 7 out of 8 benchmarks. Notably, this achievement comes with a significant reduction in memory requirement for node features during Transformer training and inference. For example, on the Physics dataset with 34,493 nodes, we only use 256×6 tokens, i.e., 23-fold memory reduction.

Heterophilous Node Classification. Furthermore, we evaluate GQT on six small or medium-scale heterophilous datasets: Squirrel and Chameleon (Rozemberczki et al., 2021), Questions, Roman-Empire, Amazon-Ratings, and Minesweeper (Platonov et al., 2023b). We compare the performance with seven variants of GNNs including GCN, GraphSAGE, GAT, GPRGNN, H2GCN (Zhu et al., 2020a), CPGNN (Zhu et al., 2021), and GloGNN (Li et al., 2022), and six variants of GTs, including GraphGPS, GOAT, NodeFormer, SGFormer, NAGphormer, and Exphormer. The baseline performance is reported from existing works (Wu et al., 2023b; Luo et al., 2024b; Platonov et al.,

Table 2: Mean node classification performance on heterophilous graphs over five runs.

		Squirrel	**Chameleon**	**Amazon-Ratings**	**Roman-Empire**	**Minesweeper**	**Questions**
Dataset	#Nodes	5,201	2,277	22,662	24,492	10,000	48,921
	#Edges	216,933	36,101	32,927	93,050	39,402	153,540
	#Features	2,089	2,325	300	300	7	301
	#Classes	5	5	18	5	2	2
	Measure	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	ROC-AUC↑	ROC-AUC↑
GNN	GCN	38.67±1.84	41.31±3.05	48.70±0.63	73.69±0.74	89.75±0.52	76.09±1.27
	GraphSAGE	36.09±1.99	37.77±4.14	53.63±0.39	85.74±0.67	93.51±0.57	76.44±0.62
	GAT	35.62±2.06	39.21±3.08	52.70±0.62	88.75±0.41	93.91±0.35	76.79±0.71
	H2GCN	35.10±1.15	26.75±3.64	36.47±0.23	60.11±0.52	89.71±0.31	63.59±1.46
	CPGNN	30.04±2.03	33.00±3.15	39.79±0.77	63.96±0.62	52.03±5.46	65.96±1.95
	GPRGNN	38.95±1.99	39.93±3.30	44.88±0.34	64.85±0.27	86.24±0.61	55.48±0.91
	GloGNN	35.11±1.24	25.90±3.58	36.89±0.14	59.63±0.69	51.08±1.23	65.74±1.19
GT	GraphGPS	39.67±2.84	40.79±4.03	53.10±0.42	82.00±0.61	90.63±0.67	71.73±1.47
	NodeFormer	38.52±1.57	34.73±4.14	43.86±0.35	64.49±0.73	86.71±0.88	74.27±1.46
	SGFormer	41.80±2.27	**44.93±3.91**	48.01±0.49	79.10±0.32	90.89±0.58	72.15±1.31
	NAGphormer	35.80±1.33	–	51.26±0.72	74.34±0.77	84.19±0.66	–
	Exphormer	36.04±1.45	–	53.51±0.46	89.03±0.37	90.74±0.53	–
	GQT(ours)	**42.54±1.37**	44.23±3.05	**53.89±0.36**	**89.21±0.43**	**95.28±0.44**	**77.28±1.36**

Table 3: Mean node classification performance on large-scale datasets over five runs.

		ogbn-proteins	**ogbn-arxiv**	**ogbn-products**	**pokec**
Dataset	#Nodes	132,534	169,343	2,449,029	1,632,803
	#Edges	39,561,252	1,166,243	61,859,140	30,622,564
	#Features	128	8	100	65
	#Classes	40	2	47	2
	Measure	ROC-AUC↑	Accuracy ↑	Accuracy ↑	Accuracy ↑
GNN	GCN	72.51±0.35	71.74±0.29	75.64±0.21	75.45±0.17
	GAT	72.02±0.44	71.95±0.36	79.45±0.59	72.23±0.18
	GPRGNN	75.68±0.49	71.10±0.12	79.76±0.59	72.23±0.18
	LINKX	71.37±0.58	66.18±0.33	71.59±0.71	82.04±0.07
	GraphSAGE	77.68±0.20	71.49±0.27	78.29±0.16	75.63±0.38
	SIGN	–	71.95±0.11	80.52±0.16	–
GT	GraphGPS	76.83±0.26	70.97±0.41	OOM	OOM
	GOAT	74.18±0.37	72.41±0.40	82.00±0.43	66.37±0.94
	NodeFormer	77.45±1.15	59.90±0.42	72.93±0.13	71.00±1.30
	SGFormer	79.53±0.38	72.63±0.13	74.16±0.31	73.76±0.24
	NAGphormer	73.61±0.33	70.13±0.55	73.55±0.21	76.59±0.25
	Exphormer	74.58±0.26	72.44±0.28	OOM	OOM
	GQT(ours)	**82.13±0.34**	**73.14±0.16**	**82.46±0.17**	**83.54±0.26**

2023a; Behrouz & Hashemi, 2024). As shown in Table 2, GQT outperforms the baselines on five out of six datasets. We observe that introducing semantic edges and structural gating mechanisms specifically benefits the heterophilous setting (see Appendix C), as they enable the Transformer to capture long-range dependencies that are not easily accessible through the original graph structure.

Large-scale Node Classification We also evaluate the performance of GQT on four large-scale datasets: ogbn-proteins, ogbn-arxiv, ogbn-products (Hu et al., 2020a), and pokec (Leskovec & Krevl, 2014), the last of which is a heterogeneous dataset. We compare the performance against six GNN variants: LINKX (Lim et al., 2021), SIGN (Frasca et al., 2020), GCN, GAT, GraphSAGE, and GPRGNN; and six GT variants: GraphGPS, GOAT, NodeFormer, NAGphormer, Exphormer, and SGFormer (Wu et al., 2024). We report the baseline performance from existing works (Wu et al., 2023b; Luo et al., 2024a). The results (Table 3) show that GQT outperforms the baseline models on all large-scale benchmarks. This is achieved while significantly reducing the required memory. For instance, on the ogbn-products dataset with 2,449,029 nodes and 100-dimensional node features, GQT requires only 3 codebooks of size 4096 each to represent the tokens, resulting in a remarkable 30-fold reduction in memory usage.

6.2 Ablation Study

Effect of Tokenization. We examine the performance of the tokenizer by training a linear model on the representations of the learned tokens without modulation, augmentation, or Transformer (1).

Table 4: Ablation study on effect of proposed components on the ogbn-arxiv dataset.

	Graph Tokenizer			**Token Modulation**			**Augmentation**		**Model**	**Performance**
	RVQ	GMAE2	DGI	Codebook Embeddings	Positional Encoding	Structural Gating	Semantic Edges	PPR Sequence		Accuracy↑
(1)	✓	✓	✓	✓					Linear	71.97
(2)					✓			✓	Transformer	70.50
(3)		✓	✓	✓	✓	✓	✓	✓	Transformer	72.84
(4)	✓		✓	✓	✓	✓	✓	✓	Transformer	71.79
(5)	✓	✓		✓	✓	✓	✓	✓	Transformer	72.71
(6)	✓	✓	✓		✓	✓	✓	✓	Transformer	71.28
(7)	✓	✓	✓	✓		✓	✓	✓	Transformer	72.69
(8)	✓	✓	✓	✓	✓		✓	✓	Transformer	73.02
(9)	✓	✓	✓	✓	✓	✓		✓	Transformer	72.61
(10)	✓	✓	✓	✓	✓	✓	✓	✓	Transformer	73.14

As shown in Table 4, within the linear evaluation protocol, the tokenizer shows strong performance, surpassing that of GTs such as GraphGPS and NAGphormer, as well as GNNs like GAT and SIGN (Table 3). This implies that the tokenizer is capable of learning effective token representations. To further investigate the importance of the tokenizer, we exclude it and train the Transformer directly on the original node features (2). As expected, this results in significant degradation in performance, highlighting the crucial role of the tokenizer. Additionally, to study the effects of vector quantization, GraphMAE2, and DGI objectives, we train the model by excluding each component (3-5). The results suggest that the SSL objectives contribute more significantly to the performance compared to vector quantization. This is because the primary purpose of vector quantization is to compress information into discrete tokens, reducing memory requirements. Between GraphMAE2 and DGI, GraphMAE2 yields the highest gain. The is due to its composition of two objectives: masked reconstruction and teacher-(noisy)student distillation. Both of these objectives have been shown to outperform InfoMax objectives on downstream tasks (Hou et al., 2022; Thakoor et al., 2022).

Effect of Modulation. We also investigate the impact of codebook embeddings, positional encoding, and structural gating on the model's performance (6-8). As shown in Table 4, introducing aggregated codebook embeddings leads to improved downstream performance due to the fact that it provides the Transformer with richer representations of each token. Positional encoding, as observed in other domains, contributes moderately to downstream performance. We also note that introducing structural gating yields moderate improvements in homophilous settings, whereas the gains are significant in heterophilous benchmarks (C). This disparity can be attributed to the ability of structural gating to provide the Transformer with importance scores computed over the global graph structure, which is particularly beneficial in heterophilous scenarios.

Effect of Augmentation. We study the effect of semantic edges on downstream performance (9). The results suggest that augmenting the graph structure with semantic edges yields significant gains. This is because introducing semantic edges allows the Transformer to access semantic information that may not be captured by the original graph structure. Furthermore, when combined with random walks, this also enables the Transformer to attend to long-range dependencies. This is especially important in heterophilous benchmarks, where semantic relationships between nodes are more nuanced.

7 Conclusion

We introduced GQT (**G**raph **Q**uantized **T**okenizer) to decouple graph tokenization from Transformer using multi-task graph self-supervised learning. The GQT uses vector quantization to learn hierarchical tokens, resulting in significantly reduced memory requirements and improved generalization. We also introduced structural gating, hierarchical encoding, and semantic edges to further improve the performance. We achieved state-of-the-art performance on 16 out of 18 datasets, including large-scale homophilic and heterophilic datasets, while significantly reducing memory requirements. As future directions, we plan to explore the effectiveness of the GQT in graph generative learning by transitioning to a Transformer decoder. Our research lays the groundwork for further investigation into Graph Foundational Models, where LLMs can project heterogeneous features from diverse datasets into a unified textual representation. Building on this foundation, our GQT model can then convert a large number of nodes across different datasets into an efficient set of tokens.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2464–2473, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pp. 1877–1901, 2020.

Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 5533–5566, 2024.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 3469–3489, 2022.

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

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, pp. 16344–16359, 2022a.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022b.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *The Twelfth International Conference on Learning Representations*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, June 2019.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2051–2060, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Gbetondji JS Dovonon, Michael M Bronstein, and Matt J Kusner. Setting the record straight on transformer oversmoothing. *arXiv preprint arXiv:2401.04301*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. VCR-graphormer: A mini-batch graph transformer via virtual connections. In *The Twelfth International Conference on Learning Representations*, 2024.

Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *International Conference on Learning Representations*, 2022.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Golnaz Ghiasi, Barret Zoph, Ekin D Cubuk, Quoc V Le, and Tsung-Yi Lin. Multi-task self-training for learning general representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8856–8865, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2017a.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017b.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126, 2020.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Van Thuy Hoang, O Lee, et al. A survey on structure-preserving graph transformers. *arXiv preprint arXiv:2401.16176*, 2024.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*, 2020.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*, pp. 737–746, 2023.

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, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332, 2018.

Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. In *International Conference on Learning Representations*, 2020.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, pp. 14582–14595, 2022.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C Bayan Bruss, and Tom Goldstein. Goat: A global transformer on large-scale graphs. In *International Conference on Machine Learning*, pp. 17375–17390, 2023a.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. GOAT: A global transformer on large-scale graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 17375–17390, 2023b.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, pp. 13242–13256, 2022.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.

Chuang Liu, Yibing Zhan, Xueqi Ma, Liang Ding, Dapeng Tao, Jia Wu, and Wenbin Hu. Gapformer: graph transformer with graph pooling for node classification. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023a.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024.

Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. In *International Conference on Learning Representations*, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Yuankai Luo, Qijiong Liu, Lei Shi, and Xiao-Ming Wu. Structure-aware semantic node identifiers for learning on graphs. *arXiv preprint arXiv:2405.16435*, 2024a.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *arXiv preprint arXiv:2406.08993*, 2024b.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23321–23337, 2023.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, pp. 1, 2012.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023a.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023b.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8821–8831, 2021.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, pp. 14501–14515, 2022.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, pp. 12559–12571, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Yuge Shi, Imant Daunhawer, Julia E Vogt, Philip Torr, and Amartya Sanyal. How robust is unsupervised representation learning to distribution shift? In *The Eleventh International Conference on Learning Representations*, 2023.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 31613–31632, 2023.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1): 396–413, 2020.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022a.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, pp. 27387–27401, 2022b.

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. DIFFormer: Scalable (graph) transformers induced by energy constrained diffusion. In *The Eleventh International Conference on Learning Representations*, 2023a.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, pp. 13266–13279, 2021.

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 *The Eleventh International Conference on Learning Representations*, 2023.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1955–1973, 2024.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, pp. 28877–28888, 2021a.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021b.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 2020.

Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pp. 12121–12132, 2021.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 558–567, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, 34:15870–15882, 2021.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020a.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11168–11176, 2021.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020b.

Markus Zopf. 1-wl expressiveness is (almost) all you need. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2022.

Appendix

A Datasets

Here we provide a detailed description of the datasets we used. All datasets are publicly available.

- **CoraFull** (Bojchevski & Günnemann, 2017), **CiteSeer**, and **Pubmed** (Namata et al., 2012) are citation datasets, where nodes represents documents and edges represent citation link. Labels indicates paper category.
- **Computer** and **Photo** (Shchur et al., 2018) are from Amazon co-purchase graph (McAuley et al., 2015), where nodes represent goods and edges indicate that two goods are frequently bought together. Node features are bag-of-words encoded product reviews, and class labels are given by the product category.
- **CS** and **Physics** (Shchur et al., 2018) are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenges. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author's papers, and class labels indicate most active fields of study for each author.
- **WikiCS** (Mernyei & Cangea, 2020) is derived from Wikipedia, where nodes are Computer Science articles, and edges are based on hyperlinks. Nodes are classified into 10 classes representing different branches of the field.
- **Squirrel** and **Chameleon** (Rozemberczki et al., 2021; Pei et al., 2020) are Wikipedia page-page networks, where nodes represent articles from the English Wikipedia, and edges reflect mutual links between them. The nodes were classified into 5 classes in terms of their average monthly traffic.
- **Amazon-Ratings** (Platonov et al., 2023b) is based on the Amazon product co-purchasing data. Nodes are products (books, music CDs, DVDs, VHS video tapes), and edges connect products that are frequently bought together. The task is to predict the average rating given to a product by reviewers.
- **Roman-Empire** (Platonov et al., 2023b) is based on the Roman Empire article from English Wikipedia. Each node in the graph corresponds to one (non-unique) word in the text. Thus, the number of nodes in the graph is equal to the length of the article. Two words are connected if these words follow each other in the text, or these words are connected in the dependency tree of the sentence. The class of a node is its syntactic role.
- **Minesweeper** (Platonov et al., 2023b) is inspired by the Minesweeper game. The graph is a regular 100x100 grid where each node (cell) is connected to eight neighboring nodes (with the exception of nodes at the edge of the grid, which have fewer neighbors). 20% of the nodes are randomly selected as mines. The task is to predict which nodes are mines. The node features are one-hot-encoded numbers of neighboring mines. However, for randomly selected 50% of the nodes, the features are unknown, which is indicated by a separate binary feature.
- **Questions** (Platonov et al., 2023b) is based on data from the question-answering website Yandex Q, where nodes are users, and an edge connects two nodes if one user answered the other user's question during a one-year time interval (from September 2021 to August 2022). The task is to predict which users remained active on the website, forming a binary classification task.
- **ogbn-proteins** (Hu et al., 2020a) is a protein-protein assiciation network, where nodes represent proteins, and edges indicate different types of biologically meaningful associations between proteins, e.g., physical interactions, co-expression or homology. The task is to predict the presence of protein functions in a multi-label binary classification setup.
- **ogbn-arxiv** (Hu et al., 2020a) is a citation network between all Computer Science (CS) arXiv papers indexed by MAG (Wang et al., 2020). Each node is an arXiv paper and each directed edge indicates that one paper cites another one. The task is to predict the 40 subject areas of arXiv CS papers, e.g., cs.AI, cs.LG, and cs.OS.

- **ogbn-products** (Hu et al., 2020a) is an Amazon product co-purchasing network[1] of 2M products. Edges indicate that the products are purchased together. The task is to predict the category of a product.
- **pokec** (Leskovec & Krevl, 2014; Lim et al., 2021) is a social network, where nodes are users, and edges represent friendships. The task is to predict the gender of users.

For CoraFull, Pubmed, PubMed, Computer, Photo, CS, and Physics, we follow previous work and use 60%/20%/20% train/valid/test split. For WiKiCS, we follow the official split in Mernyei & Cangea (2020). For Squirrel, Chameleon, Amazon-Ratings, Roman-Empire, Minesweeper, and Questions, we follow the splits in Platonov et al. (2023b). For ogbn-proteins, ogbn-arxiv, and ogbn-papers, we follow the splits in Hu et al. (2020a). And for pokec, we follow the split used in Lim et al. (2021).

B Experimental Setup

Software and hardware. The implementation of our method is based on PyTorch[2], PyG[3], DGL[4], and vector-quantize-pytorch package[5]. Most of the datasets can be accessed from PyG and DGL. All the experiments are conducted on one Nvidia A100 GPU.

Hyperparameters and experimental details. As illustrated in Figure 1, our method includes two parts: tokenizer and Transformer. We provide the hyperparameters and experimental details for each parts below.

During the training of graph tokenizer, we use full-graph training for small and medium-scale datasets, and apply sampling for large-scale graphs. We consider different sampling methods including random partitioning which randomly samples nodes within a graph and returns their induced subgraph, neighbor sampling (Hamilton et al., 2017b), GraphSAINT (Zeng et al., 2019), and local clustering used in Hou et al. (2023). For the GNN encoder and decoder, we use GCN or GAT as our backbone and tune the number of layers from {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and hidden dimensions from {128, 256, 512, 1024}. For the quantizer, we use residual-VQ (RVQ) (Lee et al., 2022) and tune the number of codebooks from {1, 2, 3, 6, 9} and codebook size from {128, 256, 512, 1024}. We set the code dimension to be the hidden dimension of the GNN encoder.

During the training of Transformer, we use KNN to add semantic edges and tune the number of semantic neighbors from {0, 5, 10, 15, 20}. Then we use PPR to generate a sequence of nodes for each target node. We tune the number of PPR neighbors from {0, 5, 10, 20, 30, 50}. For the Transformer model, we use the TransformerEncoder module in PyTorch ad our backbone, and tune the number of layers from{1, 2, 3, 4, 5, 6}, number of heads from {4, 8}, and feedforward dimension from {512, 1024, 2048}.

C Further Ablation Study

Additionally, we provide ablation study on one of the heterophilous dataset. Results are shown in Table 5. Results show that introducing semantic edges and structural gating mechanisms specifically benefits the heterophilous setting.

[1] `http://manikvarma.org/downloads/XC/XMLRepository.html`

[2] `https://pytorch.org/`

[3] `https://pyg.org/`

[4] `https://www.dgl.ai/`

[5] `https://github.com/lucidrains/vector-quantize-pytorch`

Table 5: Ablation study on effect of proposed components on the Minesweeper dataset.

	Graph Tokenizer			Token Modulation			Augmentation		Model	Performance
	RVQ	GMAE2	DGI	Codebook Embeddings	Positional Encoding	Structural Gating	Semantic Edges	PPR Sequence		ROC-AUC↑
(1)	✓	✓	✓	✓					Linear	90.11
(2)					✓			✓	Transformer	90.65
(3)		✓	✓	✓	✓	✓	✓	✓	Transformer	95.33
(4)	✓		✓	✓	✓	✓	✓	✓	Transformer	92.86
(5)	✓	✓		✓	✓	✓	✓	✓	Transformer	93.85
(6)	✓	✓	✓		✓	✓	✓	✓	Transformer	93.12
(7)	✓	✓	✓	✓		✓	✓	✓	Transformer	94.89
(8)	✓	✓	✓	✓	✓		✓	✓	Transformer	93.97
(9)	✓	✓	✓	✓	✓	✓		✓	Transformer	92.45
(10)	✓	✓	✓	✓	✓	✓	✓	✓	Transformer	95.28