CAN GRAPH QUANTIZATION TOKENZIER CAPTURE TRANSFERABLE PARTTERNS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph tokenization aims to convert graph-structured data into discrete representations that can be used in foundation models. Recent methods propose to use vector quantization to map nodes or subgraphs into discrete token IDs. However, it remains unclear whether these quantized tokenizers truly capture high-level, transferable graph patterns across diverse domains. In this work, we conduct a comprehensive empirical study to analyze the representational consistency of quantized graph tokens across different datasets. We introduce the Token Information Discrepancy Score (TIDS) to quantify the alignment of structural and feature information between source and target graphs for each token. Our results reveal that current graph quantized tokenizers often assign the same token to structurally inconsistent patterns across graphs, resulting in high TIDS and degraded transfer performance. We further demonstrate that TIDS is positively correlated with the generalization gap in downstream tasks. Finally, we propose a simple yet effective structural hard encoding (SHE) strategy to enhance the structural awareness of the tokenizer. SHE leads to lower TIDS and improved transferability, highlighting the importance of explicitly encoding transferable graph structure in token design.

1 INTRODUCTION

In recent years, graph deep learning has emerged as a powerful toolkit for modeling data with inherent relational structures (Ma & Tang, 2021; Xia et al., 2021a; Wu et al., 2020). Unlike traditional data formats such as sequences (e.g., text) or grids (e.g., images), many real-world datasets, ranging from citation networks to molecules (Xia et al., 2023; Jumper et al., 2021) , can be naturally represented as graphs (Xia et al., 2021b). To effectively process graph-structured data, a variety of graph neural networks (GNNs) have been proposed, including Graph Convolutional Networks (Yao et al., 2019), Graph Attention Networks (Veličković et al., 2018), and Graph Transformers (Yun et al., 2019; Rampášek et al., 2022; Chen et al., 2022). These graph learning methods can model non-Euclidean data well and enable learning representations for nodes, edges, and entire graphs. However, despite the impressive success of GNNs in many tasks, they usually can be trained and applied to a single dataset. The efforts of the generalizing deep graph learning models to multiple datasets have only made limited progress due to the diversity and complexity of the graph data (Mao et al., 2024).

On the other hand, the success of foundation models (Brown et al., 2020; Achiam et al., 2023; Team et al., 2023) in natural language processing (NLP) and computer vision (CV) has motivated researchers to explore analogous approaches for graphs (Mao et al., 2024). One of the important attempts is graph tokenization (Yang et al., 2023; Chen et al., 2024a), a method inspired by text and image tokenization, where raw graph inputs are transformed into a sequence or set of ”tokens” that can be processed by powerful sequence models like transformers. Just as words or subwords serve as basic units in language modeling, graph tokens aim to represent meaningful atomic or composite units of graph data.

However, unlike the tokens can be naturally defined in langugae, the are no obvious basic unit in the graph data. Hence, following the successful examples in CV (van den Oord et al., 2017; Lee et al., 2022b; Tian et al., 2024), researchers recently proposed the quantization graph tokenizer (Wang et al., 2024b; Luo et al., 2024) to learn the token representations. Specifically, the graph quantization tokenization will learn to convert a graph or subgraph into a set of vectorized representations (tokens) that encapsulate both the structural and feature information present in the original graph. Once the

tokenizer is trained, they can be applied to more datasets and generate graph tokens. Currently, the quantized graph tokenizer has achieved certain success in both supervised and unsupervised learning senarios and on different downstream tasks such as node classification, link prediction, or graph classification (Luo et al., 2024; Liu et al., 2023b; Wang et al., 2024a;b).

However, a fundamental question arises: *Do current graph tokenization methods actually capture the high-level, transferrable patterns inherent in graph data?* In other words, do the quantized tokens encode the vital graph structural information, instead of assigning the tokens heavily based on the *raw node features*?

This question is related to the fundamental capabilities of graph quantized tokenizers. Many downstream tasks in graph learning rely heavily on recognizing high-level structural patterns, such as degree distribution, homophily, and centrality. For example, in drug discovery, subtle topological variations in molecular graphs—captured by molecular topology and centrality descriptors—can directly influence biological activity and binding affinity (Zhang et al., 2025; Csermely et al., 2012). In social networks, tasks like community detection or influence modeling also depend critically on network connectivity and central nodes (Barabási & Oltvai, 2004; McPherson et al., 2001). When quantization tokenization fails to preserve these essential graph properties, the resulting graph tokens may omit meaningful structural patterns, impairing downstream task performance.

In this study, we present the first comprehensive empirical investigation into the knowledge encoded by graph quantized tokenizers. Specifically, we measure the discrepancy in both structural and feature information of nodes that are mapped to the same token across different datasets. Our analysis reveals that identical tokens often correspond to markedly different structural distributions in different graphs, indicating that *current graph quantized tokenizers fail to capture high-level, transferable patterns*. This deficiency undermines both the stability and cross-domain generalization ability of such tokenizers. The contributions of this work are as follows:

- We have analyzed both the structural and feature information encoded by the graph tokenizer. We find that there are significant information distribution discrepancies for the same token across different graphs.
- We show that the information discrepency of the tokens will hinder the model's transferrability, resulting in sub-optimal performance on the downstream tasks.
- Based on the findings above, we propose a trick to explicitly help the graph quantization tokenizer to encode the structural information. We show that the trick could mitigate the information discrepancy of tokens on different graphs, further affirming the value of our observations.

2 Related Works

Graph tokenization sits at the intersection of representation learning, graph neural networks (GNNs), and transformer-based models, drawing inspiration from tokenization practices in natural language processing and computer vision. Several strands of related research contribute to the development of tokenization methods for graph data.

Quantization and Discrete Representation Learning. Quantized latent representation learning has emerged as a powerful strategy to bridge the gap between continuous data and discrete symbolic reasoning. Among the most influential approaches, Vector Quantized Variational Autoencoder (VQ-VAE) (van den Oord et al., 2017; Esser et al., 2021) introduced a discrete bottleneck into the autoencoding framework, enabling learning of a codebook of latent embeddings that can compactly represent high-dimensional inputs. VQ-VAE has seen broad success in areas such as *image generation*, *speech modeling*, and *language modeling*, where discrete tokens enable autoregressive decoding and large-scale pretraining. Its extension, Residual Quantized VAE (RQ-VAE) (Lee et al., 2022a) addresses the limited capacity of shallow codebooks by employing *multi-level quantization*, decomposing inputs into multiple additive residuals. This yields richer token representations and better compression, making it particularly suitable for complex modalities.

Quantized Representations in Graph Learning. Despite the success of VQ-VAE in vision and language domains, its adaptation to *graph-structured data* remains relatively underexplored. Unlike pixels or words, graphs are *non-Euclidean* and *permutation-invariant*, posing significant challenges

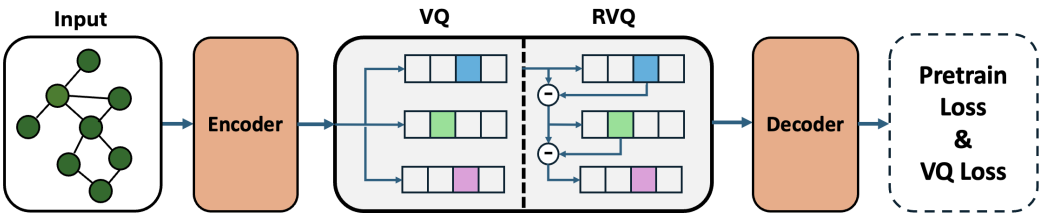

Figure 1: The pipeline of graph quantized tokenizer.

for tokenization. Several recent efforts have sought to bridge this gap. GraphMAE (Hou et al., 2022) and GPT-GNN (Hu et al., 2020) introduced self-supervised frameworks for node- and graph-level representation learning, but they rely on continuous encodings. A more direct attempt at tokenization can be found in GVT (You et al., 2023), which integrates VQ-VAE to learn discrete node prototypes and supports masked autoencoding on graphs. However, such methods typically apply quantization at the node level, ignoring higher-order structures or global subgraph semantics.

Graph Pretraining with Structural Discreteness. Recent works such as OneForAll (Liu et al., 2023a) and GFT (Graph Foundation Model with Transferable Tree Vocabulary) (Wang et al., 2024b) argue for discrete graph vocabulary learning to enable large-scale generalization across domains. OneForAll explores cross-domain pretraining with task-level tokenization, while GFT builds hierarchical tree vocabularies based on rooted subtrees, which are then quantized for structural reuse. Other notable approaches include AnyGraph (Xia & Huang, 2024), which aims to unify different graph modalities with plug-and-play architecture, and GraphPrompt (Jin et al., 2022), which leverages discrete prompts to guide downstream adaptation.

3 Methodology

In this section, we will introduce the graph quantized tokenizer to be investigated. We will first introduce the key components, namely Vector Quantization (VQ) and Residual Vector Quantization (RVQ). Next, we will introduce the whole pipeline as shown in Fig. 1.

3.1 Vector Quantization Methods

Vector Quantization (VQ) (Gray, 1984; Gong et al., 2014; Esser et al., 2021) aims to represent a large set of vectors, $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1}^{N}$, with a small set of prototype (code) vectors of a codebook $\boldsymbol{C} = \{\boldsymbol{e}_k\}_{k=1}^{K}$, where $N \gg K$. The codebook is often created using algorithms such as k-means clustering via optimizing the following objective:

$$\min_{\boldsymbol{C}} \sum_{i=1}^{N} \min_{k=1}^{K} ||\boldsymbol{z}_i - \boldsymbol{e}_k||_2^2 . \tag{1}$$

Once the codebook is learned, each vector $\boldsymbol{z}_i$ can be approximated by its closet prototype vector $\boldsymbol{e}_t$, where $t = \arg\min_k ||\boldsymbol{z}_i - \boldsymbol{e}_k||_2^2$ is the index of the prototype vector.

Residual Vector Quantization (RVQ) (Juang & Gray, 1982; Martinez et al., 2014; Lee et al., 2022a) is an extension of the basic VQ. After performing an initial VQ, the *residual vector* is calculated:

$$\boldsymbol{r}_i = \boldsymbol{z}_i - \boldsymbol{e}_t, \tag{2}$$

which represents the quantization error from the initial quantization. Then, the residual vectors $\boldsymbol{r}_i$ are quantized using a second codebook. This process can be repeated multiple times, with each stage quantizing the residual error from the previous stage.

3.2 GRAPH QUANTIZED TOKENIZER

The graph quantized tokenizer intends to assign a token ID to the given node based on its own feature and neighboring nodes. To generate structure-aware node IDs, we employ an L-layer MPNN to capture multi-order neighborhood structures. At each layer, we use vector quantization to encode the node embeddings produced by the MPNN into M codewords (integer indices). For each node v, we define the node ID of v as a tuple composed of $L \times M$ codewords, structured as follows:

$$\text{Node\_ID}(v) = (c_{11}, \cdots, c_{1M}, c_{21}, \cdots, c_{2M}, \cdots) \tag{3}$$

where c_{lm} represents the m-th codeword at the l-th layer. Both M and L can be very small.

As illustrated in Fig. 1, at each layer l ($1 \leq l \leq L$) of the MPNN, we employ VQ/RVQ to quantize the node embeddings and produce M digits of codewords for each node v. Each codeword c_{lm} ($1 \leq m \leq M$) is generated by a distinct codebook $\boldsymbol{C}_{lm} = \{\boldsymbol{e}_k^{lm}\}_{k=1}^K$, where K is the size of the codebook. Hence, there are a total of $L \times M$ codebooks, indexed by lm. Let $\boldsymbol{r}_{lm}$ denote the vector to be quantized. Note that $\boldsymbol{r}_{l1}$ is the node embedding $\boldsymbol{h}_v^l$ produced by the MPNN. When $m > 1$, $\boldsymbol{r}_{lm}$ represents the residual vector. Then, $\boldsymbol{r}_{lm}$ is approximated by its nearest code vector from the corresponding codebook $\boldsymbol{C}_{lm}$:

$$c_{lm} = \arg\min_k ||\boldsymbol{r}_{lm} - \boldsymbol{e}_k^{lm}||, \tag{4}$$

producing the codeword c_{lm}, which is the index of the nearest code vector.

We follow the existing framework for learning node token IDs (codewords c_{lm}) by jointly training the MPNN and the codebooks with the following loss function:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\mathcal{G}} + \mathcal{L}_{\text{VQ}}, \tag{5}$$

where $\mathcal{L}_{\mathcal{G}}$ is a (self)-supervised graph learning objective, and $\mathcal{L}_{\text{VQ}}$ is a vector quantization loss. $\mathcal{L}_{\mathcal{G}}$ aims to train the MPNN to produce effective node embeddings, while $\mathcal{L}_{\text{VQ}}$ ensures the codebook vectors align well with the node embeddings. For a single node v, $\mathcal{L}_{\text{VQ}}$ is defined as

$$\mathcal{L}_{\text{VQ}} = \sum_{l=1}^{L} \sum_{m=1}^{M} \|\text{sg}(\boldsymbol{r}_{lm}) - \boldsymbol{e}_{c_{lm}}^{lm}\| + \beta \|\boldsymbol{r}_{lm} - \text{sg}(\boldsymbol{e}_{c_{lm}}^{lm})\|, \tag{6}$$

where sg denotes the stop gradient operation, and β is a weight parameter. The first term in Eq. (6) is the *codebook loss*, which only affects the codebook and brings the selected code vector close to the node embedding. The second term is the *commitment loss*, which only influences the node embedding and ensures the proximity of the node embedding to the selected code vector. In practice, we can use exponential moving averages (Razavi et al., 2019) as a substitute for the *codebook loss*.

The graph learning objective $\mathcal{L}_{\mathcal{G}}$ can be a self-supervised learning task, such as graph reconstruction (i.e., reconstructing the node features or graph structures) or contrastive learning (Liu et al., 2021). In this paper, we follow most of the existing works that utilize GraphMAE (Hou et al., 2022). GraphMAE involves sampling a subset of nodes $\tilde{\mathcal{V}} \subset \mathcal{V}$, masking the node features as $\tilde{\boldsymbol{X}}$, encoding the masked node features using an MPNN, and subsequently reconstructing the masked features with a decoder. The reconstruction loss is based on the scaled cosine error, expressed as:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{|\tilde{\mathcal{V}}|} \sum_{v \in \tilde{\mathcal{V}}} \left(1 - \frac{\boldsymbol{x}_v^T \boldsymbol{z}_v}{\|\boldsymbol{x}_v\| \cdot \|\boldsymbol{z}_v\|} \cdot \gamma\right),$$

where $\tilde{\mathcal{V}}$ is the set of masked nodes, $\boldsymbol{z}_v = f_D(\tilde{\boldsymbol{h}}_v^L)$ is the reconstructed node features by a decoder f_D, $\tilde{\boldsymbol{h}}_v^L = \text{MPNN}(v, \boldsymbol{A}, \tilde{\boldsymbol{X}})$, and $\gamma \geq 1$ is a scaling factor. Let $\tilde{\boldsymbol{r}}_{l1} := \tilde{\boldsymbol{h}}_v^l$ denote the node embedding generated by the l-th layer of the MPNN with the masked features. The overall training loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MAE}} + \sum_{v \in \tilde{\mathcal{V}}} \sum_{l=1}^{L} \sum_{m=1}^{M} \|\text{sg}(\tilde{\boldsymbol{r}}_{lm}) - \boldsymbol{e}_{c_{lm}}\| + \beta \|\tilde{\boldsymbol{r}}_{lm} - \text{sg}(\boldsymbol{e}_{c_{lm}})\|. \tag{7}$$

4 PRELIMILARY

4.1 EXPERIMENT SETUPS

Here we first introduce our experiment setups, i.e., how we train and evaluate the graph quantized tokenizer. In order to obtain the comprehensive results, we train and evaluate both VQ and RVQ methods. For all the models, we set the number of MPNN layers to be 2, and the number of codewords to be 3. We train the tokenizer on the datasets from two domains: citation graphs and e-commerce networks. The citation graphs include: cora, citesser, dblp, arxiv and pubMed. The e-commerce graphs include: bookhis, bookchild, elecomp, elephoto and sportsfit. The detailed information of the datasets can be found in Appendix A. For each domain, we pretrain the tokenzier on 1 to 4 datasets and then use infer on the remaining datasets in the domain. On both training and test datasets, we will record the subgraphs that assigned to each token ID. For instance, for a node token ID c_{mn}, we will record the subgraphs in training set assigned to it as a $\mathcal{S}_{mn,train}$, and we will record the the subgraphs in test set assigned to it as a $\mathcal{S}_{mn,test}$ Then we would calculate the information discrepancy between $\mathcal{S}_{mn,train}$ and $\mathcal{S}_{mn,test}$ for each token.

4.2 EVALUATION METRIC

Here we will introduce the metric we design to measure the information discrepancy of tokens. Specifically, we design a metric named Graph Token Information Discrepency Score (GTID) to calculate the discrepancy between $S_{mn,train}$ and $S_{mn,test}$. Suppose the representations of $S_{mn,train}$ and $S_{mn,tes}$ are $f_{mn,train}$ and $f_{mn,test}$. Following the previous works (Yan et al., 2017; Wang et al., 2021), we use Maximum Mean Discrepancy (MMD) to calculate the discrepancy between $f_{mn,train}$ and $f_{mn,test}$. Specifically, we tend to compare the MMD computed on node features and structures. Therefore, we adapt Normalized Maximum Mean Discrepancy (NMMD) in this work. First we normalize the vectors in $f_{mn,train}$ and $f_{mn,test}$ and denote $\hat{f}_{mn,train} = \{\mathbf{p}_i\}_{i=1}^{v}$ and $\hat{f}_{mn,test} = \{\mathbf{q}_i\}_{i=1}^{w}$: Then we first calculate the MMD of the two vector sets:

$$\widehat{\mathrm{MMD}}^2 = \frac{1}{v^2}\sum_{i=1}^{v}\sum_{i'=1}^{v} k(\mathbf{p}_i, \mathbf{p}_{i'}) + \frac{1}{w^2}\sum_{j=1}^{w}\sum_{j'=1}^{w} k(\mathbf{q}_j, \mathbf{q}_{j'}) - \frac{2}{vw}\sum_{i=1}^{v}\sum_{j=1}^{w} k(\mathbf{p}_i, \mathbf{q}_j). \tag{8}$$

where $k(\cdot, \cdot)$ is an RKHS kernel. And next we calculate the variance-normalized MMD:

$$\widehat{\mathrm{NMMD}}^2 = \frac{\widehat{\mathrm{MMD}}^2}{\widehat{V}}, \qquad \widehat{V} = \frac{1}{v}\sum_{i=1}^{v} k(\mathbf{p}_i, \mathbf{q}_i) + \frac{1}{w}\sum_{j=1}^{w} k(\mathbf{q}_j, \mathbf{q}_j). \tag{9}$$

Finally, the GTID between the train and test domains is calculated with average of the normalized maximum mean discrepancy on all the codewords:

$$\mathrm{GTID} = \frac{\sum_m \sum_n \mathrm{NMMD}(f_{mn,train}, f_{mn,test})}{mn} \tag{10}$$

The more information of calculation of Maximum Mean Discrepancy and Normalized Maximum Mean Discrepancy can be found in the Appendix. In general, the larger GTID is, the larger information discrepancy is.

Since the subgraphs contain both structural and feature information, we will calculate the GTID for node features and structures respectively. For node features, we adapt the set of center node features as $f_{mn,train}$ and $f_{mn,test}$. For the structures, we calculate the structure property vectors [degree, clustering coefficient, closeness centrality, density, assortativity, transitivity, homophily] as the representations. We give the details of calculating the structural properties in Appendix C. Next, we will analyze the GTID and their relations to the model generalization. We observed similar phenomena for both RVQ and VQ tokenizers. Hence, we mainly discuss the results based on RVQ tokenizer and leave the results of VQ tokenizer to Appendix E.

4.3 THEORETICAL ANALYSIS

Before diving into the empirical observations, we first derive theorectical analysis to prove the relationship between the token information discrepancy and the model transferability. We tend to prove that low information discrepancy in tokens can lead to higher transferability and do this for both node features and ego-graph structures. We will first give the theorems and provide the full proof in Appendix D.

Theorem 1 (Code-Conditional Transfer Bound: Feature View)**.** *Let $\mathcal{D}_s, \mathcal{D}_t$ be source/target node datasets (from graphs G_s, G_t). Each node v has an L-hop ego-subgraph $g(v)$ with feature tensor. A fixed encoder $\phi : \mathcal{G} \to \mathbb{R}^m$ maps g to $z = \phi(g)$. A codebook Q with codes $\{c_1, \ldots, c_K\}$ assigns $K(g) = Q(z) \in [K]$ by nearest center. A predictor h consumes a feature summary $u(g) \in \mathbb{R}^{d'}$ (e.g., pooled/root features), and the loss $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is bounded.*

Notation. *Let $\pi_\alpha(k) = \Pr_{(g,y)\sim\mathcal{D}_\alpha}[K(g) = k]$ for $\alpha \in \{s, t\}$, and let $\mathbb{P}^k_\alpha$ be the conditional law of (g, x, y) given $K(g) = k$. Define risks $\varepsilon_\alpha(h \circ Q \circ \phi) = \mathbb{E}_{\mathcal{D}_\alpha}[\ell(h(Q(\phi(g))), y)]$. Let the* code-marginal drift *be $\Delta_{\text{code}} := \frac{1}{2}\sum_{k=1}^{K} |\pi_t(k) - \pi_s(k)|$. Define the* quantization distortion *$\Delta_{\text{q}} := \sup_{(g,y)} |\ell(h(Q(\phi(g))), y) - \ell(h(\phi(g)), y)|$, and let $p_{\text{mis}} := \Pr[K(g)$ is a misassignment$]$ (e.g., stale codebook/ANN search).*

Assumptions. *(i) There exists a (possibly identity) preprocessing $S : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$ such that $u \mapsto \ell(h(u), y)$ is L_u-Lipschitz uniformly in y. (ii) For each code k, define the within-code feature discrepancy*

$$\Delta_k^{\text{feat}} := W_1\big(\mathcal{L}_t(S(u) \mid K = k), \mathcal{L}_s(S(u) \mid K = k)\big),$$

the 1-Wasserstein distance between the conditional feature summaries.

Claim. *For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the draws of the (finite) datasets and the code-conditional estimates,*

$$\varepsilon_t - \varepsilon_s \le \sum_{k=1}^{K} \pi_t(k)\, L_u\, \Delta_k^{\text{feat}} + \Delta_{\text{code}} + \Delta_{\text{q}} + p_{\text{mis}} + c_1\sqrt{\frac{\log(2K/\delta)}{\min_k n_t(k)}} + c_2(\mathfrak{R}_{n_s} + \mathfrak{R}_{n_t})$$

Here $n_t(k)$ is the number of target samples with $K = k$, $\mathfrak{R}_{n_\alpha}$ denotes the Rademacher complexity of the induced loss class on domain α, and constants c_1, c_2 depend only on sub-Gaussian/boundedness parameters of $S(u)$ and on standard symmetrization constants.

Theorem 2 (Code-Conditional Transfer Bound: Structure View)**.** *Same as Theorem 1, except the predictor h depends on a structural representation $\psi(g)$ that lies in an RKHS $(\mathcal{H}, \langle\cdot,\cdot\rangle_{\mathcal{H}})$ with kernel $k(\cdot,\cdot)$ and $\|\psi(g)\|_{\mathcal{H}} \le B$. Risks, $\pi_\alpha(k)$, $\mathbb{P}^k_\alpha$, Δ_{code}, Δ_{q}, and p_{mis} are as defined there.*

Assumptions. *(i) For each code k, the conditional loss as a function of $\psi(g)$ belongs to a bounded RKHS ball: there exists $f_k \in \mathcal{H}$ with $\|f_k\|_{\mathcal{H}} \le C$ such that $\mathbb{E}[\ell(h(\psi(g)), y) \mid g, K = k] = \langle f_k, \psi(g)\rangle_{\mathcal{H}}$. (ii) For each code k, define the within-code structural discrepancy*

$$\Delta_k^{\text{struct}} := \text{MMD}_{\mathcal{H}}\big(\mathcal{L}_t(\psi \mid K = k), \mathcal{L}_s(\psi \mid K = k)\big).$$

Claim. *For any $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\varepsilon_t - \varepsilon_s \le \sum_{k=1}^{K} \pi_t(k)\, C\, \Delta_k^{\text{struct}} + \Delta_{\text{code}} + \Delta_{\text{q}} + p_{\text{mis}} + \tilde{c}_1\sqrt{\frac{\log(2K/\delta)}{\min_k n_t(k)}} + \tilde{c}_2(\mathfrak{R}_{n_s} + \mathfrak{R}_{n_t}),$$

where $\tilde{c}_1, \tilde{c}_2$ depend only on the kernel bound $k(x, x) \le B^2$ and standard generalization constants.

5 RESULTS AND OBSERVATIONS

5.1 THE GTID DISTRIBUTIONS

Following the evaluation process above, we can pretrain the tokenizers and evaluate them. Specifically, we would pretrain the tokenizer with different combinations of datasets: from single dataset to four

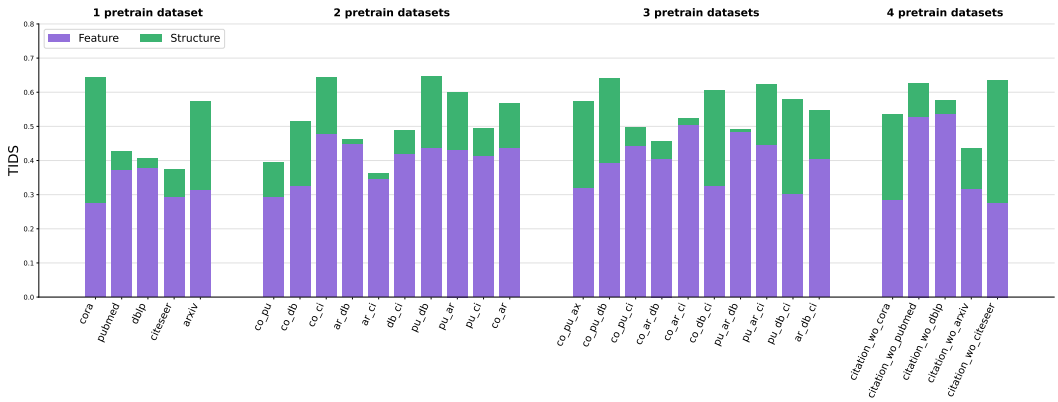

Figure 2: The distributions of GTID of RVQ models on citation datasets.

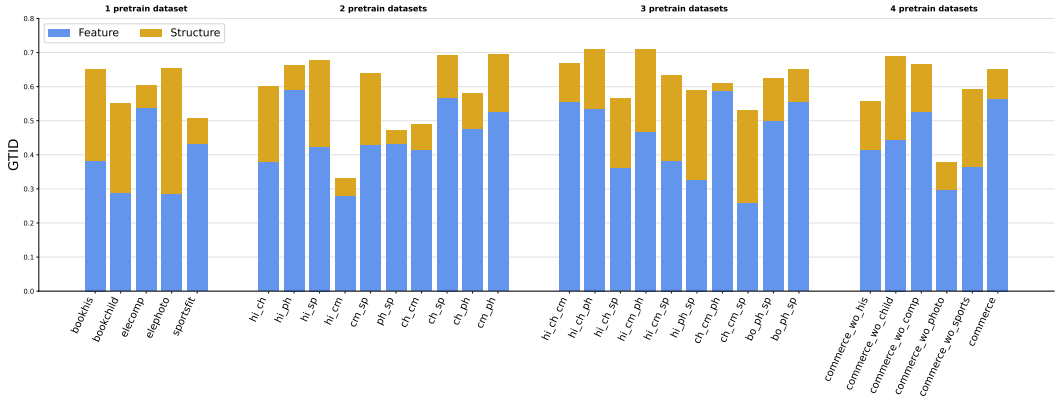

Figure 3: The distributions of GTID of RVQ models on e-commerce datasets.

datasets together. Then we will evaluate them on the remaining datasets in the same domain and calculate the corresponding GTID. The results are shown in Figure 2 and Figure 3 for the Citation domain and E-commerce domain, respectively.

Across both domains, we observe a consistent and obvious gap between structure-based and feature-based GTID. While feature discrepancy tends to decrease gradually as the number of pretraining datasets increases, the structural GTID remains relatively high and fluctuates across settings. This suggests that even with multi-dataset pretraining, the tokenizer struggles to align structural information consistently. For instance, in the Citation domain (Figure 2), structural GTID plateaus after the second pretraining dataset, indicating limited marginal gains in structural transferability. A similar trend is seen in the E-Commerce domain where feature-based discrepancy steadily decreases but structural discrepancy remains elevated, particularly in dataset groups that are structurally diverse.

Furthermore, while tokenizers benefit from more diverse feature distributions during pretraining, their ability to generalize structural semantics is far more constrained. This asymmetry highlights a key limitation of current quantization-based tokenizers: their reliance on local node features or first-order neighborhoods makes it difficult to internalize structural motifs that generalize across domains with heterogeneous graph topology. Hence, we would have the following observation:

Observation 1: The graph quantized tokenizes have difficulty capturing the transferrable patterns across graphs, especially the structural patterns.

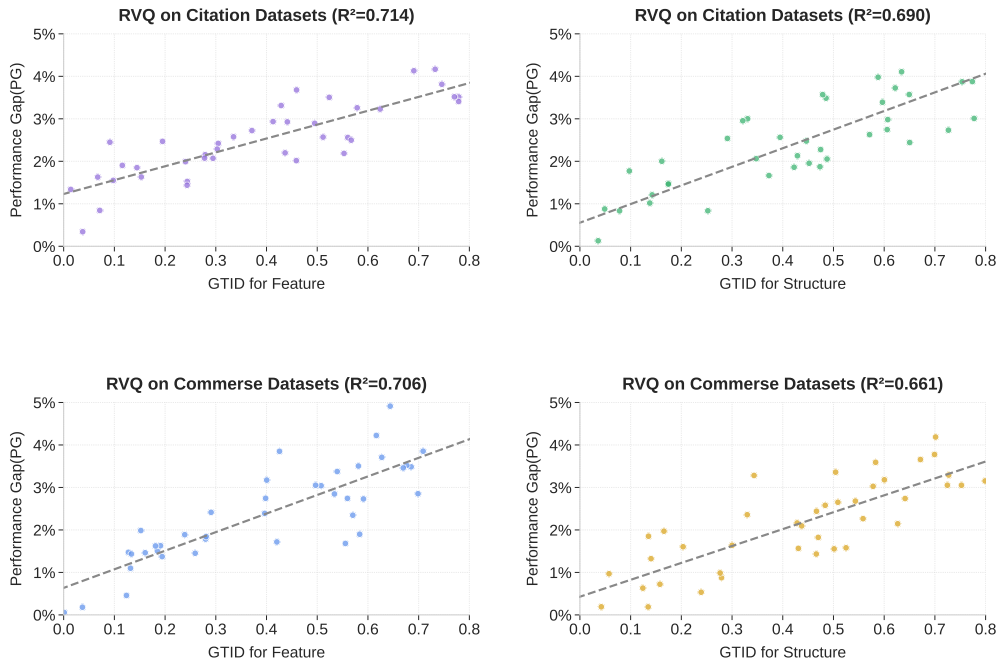

Figure 4: The correlation between GTID and the model performance gap.

5.2 The Correlations between GTID and Model's Transferability

Furthermore, we leverage the results above to analyze the relationship between model generalization and performance gaps. Specifically, we define the *performance gap* (PG) as a metric to quantify generalization ability, measured by the accuracy difference between inter-dataset and intra-dataset pretraining.

For example, consider two datasets, A and B. Let P_1 denote the node classification accuracy of a model pretrained on A and fine-tuned on B_{train}, and P_2 denote the accuracy of a model both pretrained and fine-tuned on B_{train} (B_{train} and B_{train} are the training part and test part of dataset B, respectively). The performance gap is then computed as:

$$\text{PG} = \frac{P_2 - P_1}{P_2}.$$

This normalized gap reflects how well the pretrained knowledge transfers across datasets. The results are shown in Figure 4. The reported *coefficient of determination* (R^2) quantifies the extent to which GTID explains the transfer performance degradation.

The results are shown in Figure 4. Across all settings, we observe a strong positive correlation between GTID and performance gap. In the Citation domain, feature-based GTID achieves an R^2 of 0.714, while structure-based GTID yields 0.707. A similar trend is observed in the E-Commerce domain, where the feature and structure correlations yield R^2 values of 0.709 and 0.692, respectively. These results suggest that both forms of token discrepancy significantly affect downstream transferability, with feature discrepancy often exhibiting slightly higher explanatory power, potentially due to its stronger alignment with task-relevant attributes.

These findings indicate that token consistency across domains is critical for effective transfer learning. When the same token index encodes semantically or structurally divergent patterns across graphs, the transfer model struggles to leverage pre-learned representations. This mismatch leads to notable performance degradation during cross-domain adaptation.

Observation 2: The GTID is positively correlated the performance gap, indicating that the information discrepancy of the tokens will hinder model's transferability.

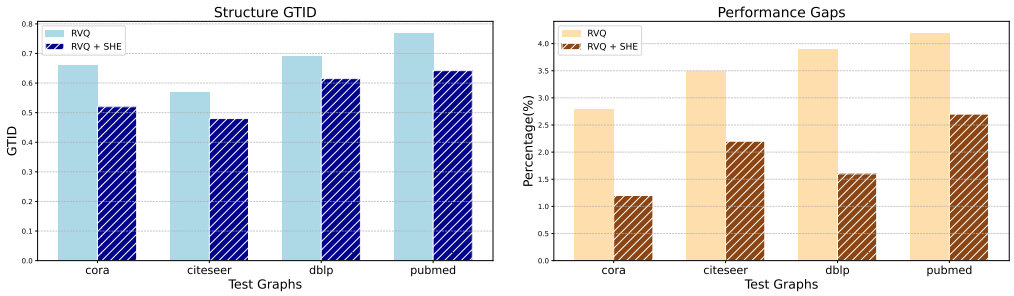

Figure 5: Comparison with the original RVQ tokenizer after utilizing the structural hard encoding.

5.3 Structural Hard Encoding

To evaluate whether adding structural information to the tokenizer can improve transferability, we incorporate a simple yet effective inductive bias: **Structural Hard Encoding (SHE)**. SHE explicitly encodes high-level structural cues (e.g., node degree bins, positional encodings) into the input of the quantized tokenizer, aiming to reduce the mismatch in structural semantics across graphs. For instance, nodes with degree 1 or 2 can only be assigned to ID 0 to 31, nodes with degree 3 can only assigned to ID 32 to 63, etc. In this way, we force the token ID distinguish with each other as their corresponding subgraphs will have structural properties' differece.

As shown in Figures 5, SHE leads to a notable improvement in both structural alignment and downstream task performance. In Figure 5 Left, we observe that for all test graphs (Cora, Citeseer, DBLP, Pubmed), the *structure-based GTID* is consistently lower when using RVQ with SHE compared to vanilla RVQ. This reduction is especially pronounced on datasets with higher structural variability (e.g., DBLP and Pubmed), indicating that SHE effectively mitigates token inconsistency arising from structural heterogeneity.

The benefits of this structural regularization also translate into improved model generalization. Figure 5 Right shows that the *performance gap* between source-pretrained and target-finetuned models is also reduced across the same set of graphs when SHE is applied. This reinforces the claim that lower GTID correlates with improved transferability, and affirms that enhancing structural awareness during tokenization is a viable pathway to better cross-graph generalization. Hence, we would have the following observation:

Observation 3: With structural hard encoding (SHE), the RVQ tokenizer can reduce the structural GTID and performance gaps, which further affirm our previous observations and the importance of capturing transferrable for tokens.

6 Conclusion

In this paper, we investigate whether graph quantized tokenizers can capture transferable patterns across graph datasets. Through a detailed empirical analysis, we show that tokenized representations suffer from significant information discrepancies, particularly in structural properties, across different domains. We introduce the Token Information Discrepancy Score (TIDS) to quantify this phenomenon and demonstrate its strong correlation with performance degradation in transfer learning settings. These findings indicate that current quantized tokenization schemes are limited in their ability to produce consistent, reusable representations for graph data. To address this, we propose Structural Hard Encoding (SHE), a simple inductive bias that explicitly incorporates structural signals into the token assignment process. Our experiments show that SHE significantly reduces structural TIDS and improves cross-domain performance, validating our core hypothesis. This work provides actionable insights into the limitations of current graph tokenizers and opens up future research directions on structure-aware, transferable graph token learning.

Ethics Statement

We acknowledge that we have read and commit to adhering to the ICLR Code of Ethics. Our study relies solely on publicly available benchmark datasets Chen et al. (2024b). While our proposed method presents no direct ethical concerns, the improved performance could be leveraged for both ethical and unethical applications involving generative recommendation systems. We emphasize the importance of applying machine learning algorithms responsibly to achieve socially beneficial results.

Reproducibility Statement

Our experiments are based on the public datasets and code (Wang et al., 2024b; Chen et al., 2024b). To help reproducibility of the results, we provide experiment settings in the main text.

Usage of Large language models

In this manuscript, we solely utilize LLMs to polish the writing and check grammatical errors. We have reviewed the generated contents provided by large language models and will be responsible for the correctness of the polished content.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Albert-László Barabási and Zoltan N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 2004.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International conference on machine learning*, pp. 3469–3489. PMLR, 2022.

Yongqiang Chen, Quanming Yao, Juzheng Zhang, James Cheng, and Yatao Bian. Improving graph-language alignment with hierarchical graph tokenization. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024a.

Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, et al. Text-space graph foundation models: Comprehensive benchmarks and new insights. *Advances in Neural Information Processing Systems*, 37:7464–7492, 2024b.

Peter Csermely, Tamas Korcsmaros, Huba J. M. Kiss, Gabor London, and Ruth Nussinov. Structure and dynamics of molecular networks: A novel paradigm of drug discovery. a comprehensive review. *arXiv preprint arXiv:1210.0330*, 2012.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

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.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 1857–1867, 2020.

Wei Jin, Ke Yang, Xianfeng Tang, Yujia Liu, and Jiliang Tang. Graphprompt: Towards universal graph pre-training via prompt-based learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Biing-Hwang Juang and A Gray. Multiple stage vector quantization for speech coding. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pp. 597–600. IEEE, 1982.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 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, 2022a.

Youngjung Lee, Taesung Kim, Seunghoon Kim, and Heechul Song. Residual vector quantization: A robust discrete representation learning framework. In *European Conference on Computer Vision (ECCV)*, 2022b.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023a.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876, 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. *Advances in Neural Information Processing Systems*, 36:25854–25875, 2023b.

Yuankai Luo, Hongkang Li, Qijiong Liu, Lei Shi, and Xiao-Ming Wu. Node identifiers: Compact, discrete representations for efficient graph learning. *arXiv preprint arXiv:2405.16435*, 2024.

Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *Forty-first International Conference on Machine Learning*, 2024.

Julieta Martinez, Holger H Hoos, and James J Little. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*, 2014.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 2001.

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.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

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

Limei Wang, Kaveh Hassani, Si Zhang, Dongqi Fu, Baichuan Yuan, Weilin Cong, Zhigang Hua, Hao Wu, Ning Yao, and Bo Long. Learning graph quantized tokenizers. *arXiv preprint arXiv:2410.13798*, 2024a.

Wei Wang, Haojie Li, Zhengming Ding, Feiping Nie, Junyang Chen, Xiao Dong, and Zhihui Wang. Rethinking maximum mean discrepancy for visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):264–277, 2021.

Zehong Wang, Zheyuan Zhang, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. Gft: Graph foundation model with transferable tree vocabulary. *Advances in Neural Information Processing Systems*, 37: 107403–107443, 2024b.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021a.

Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021b.

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. 2023.

Lianghao Xia and Chao Huang. Anygraph: Graph foundation model in the wild. *arXiv preprint arXiv:2408.10700*, 2024.

Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2272–2281, 2017.

Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, Bin Cui, Muhan Zhang, and Jure Leskovec. Vqgraph: Graph vector-quantization for bridging gnns and mlps. *arXiv preprint arXiv:2308.02117*, 2023.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7370–7377, 2019.

Jiaxuan You, Rex Ying, and Jure Leskovec. Gvt: Graph discrete vae tokenizer for transferable graph pretraining. In *International Conference on Learning Representations (ICLR)*, 2023.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

Odin Zhang, Haitao Lin, Xujun Zhang, and et al. Graph neural networks in modern ai-aided drug discovery. *arXiv preprint arXiv:2506.06915*, 2025.

A Dataset Details

Table 1 presents the detailed statistics of datasets we used in our experiments, including the dataset's domain and sizes.

Table 1: Dataset statistics.

Dataset	Domain	# Nodes	# Edges
Cora	Citation	2708	10556
Citeseer	Citation	3186	8450
Pubmed	Citation	19717	88648
DBLP	Citation	14376	431326
Arxiv	Citation	169343	2315598
Bookhis	E-commerce	41551	503180
Bookchild	E-commerce	76875	2325044
Elecomp	E-commerce	87229	1256548
Elephoto	E-commerce	48362	873782
Sportsfit	E-commerce	173055	3020134

B Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) is a statistical distance metric used to measure the discrepancy between two probability distributions P and Q over a domain $\mathcal{X}$. Formally, given a function class $\mathcal{F}$, the MMD is defined as

$$\mathrm{MMD}[\mathcal{F}, P, Q] = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim Q}[f(y)] \right).$$

When $\mathcal{F}$ is chosen to be the unit ball in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with kernel function k, the squared MMD can be computed in closed form as

$$\mathrm{MMD}^2(P, Q) = \mathbb{E}_{x, x' \sim P}[k(x, x')] + \mathbb{E}_{y, y' \sim Q}[k(y, y')]$$

For empirical distributions based on samples $\{x_i\}_{i=1}^m$ from P and $\{y_j\}_{j=1}^n$ from Q, an unbiased estimator of the squared MMD is given by

$$\mathrm{MMD}^2(P, Q) = \frac{1}{m(m-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(y_i, y_j)$$

This formulation makes MMD particularly useful for two-sample tests and as a loss function in machine learning tasks such as domain adaptation and generative modeling. The Normalized Maximum Mean Discrepancy is calculated as

$$\mathrm{Normalized\_MMD}^2(P, Q) = \frac{\mathrm{MMD}^2(P, Q)}{\mathrm{MMD}^2(P, P) + \mathrm{MMD}^2(Q, Q)}$$

C The Definations of the Structural Properties

Degree (node & average)

$$k_i \;=\; \sum_{j=1}^{n} A_{ij}, \qquad \bar{k} \;=\; \frac{1}{n} \sum_{i=1}^{n} k_i \;=\; \frac{2m}{n}.$$

Local clustering coefficient & global averages:

$$C_i \;=\; \begin{cases} \dfrac{2\,t_i}{k_i(k_i-1)}, & k_i \geq 2, \\ 0, & k_i < 2, \end{cases} \quad \text{where} \quad t_i \;=\; \sum_{1 \leq p < q \leq n} A_{ip} A_{iq} A_{pq}.$$

$$C_{\text{avg}} \;=\; \frac{1}{n}\sum_{i=1}^{n} C_i.$$

Closeness centrality:

$$\text{clo}(i) \;=\; \sum_{\substack{j=1\\ j\neq i}}^{n} d(i,j), \qquad \text{CC}(i) \;=\; \frac{n-1}{\text{clo}(i)}.$$

Density:

$$\delta(G) \;=\; \frac{2m}{n(n-1)}.$$

Degree assortativity: Let $\mu \;=\; \frac{1}{2m}\sum_{(u,v)\in E}\big(k_u + k_v\big)$.

$$r_{\text{deg}} \;=\; \frac{\dfrac{1}{m}\displaystyle\sum_{(u,v)\in E} k_u k_v \;-\; \mu^2}{\dfrac{1}{m}\displaystyle\sum_{(u,v)\in E} \frac{k_u^2 + k_v^2}{2} \;-\; \mu^2}.$$

Transitivity:

$$T \;=\; \frac{3\,\triangle}{\wedge} \;=\; \frac{\displaystyle\sum_{i=1}^{n} 2\,t_i}{\displaystyle\sum_{i=1}^{n} k_i(k_i-1)},$$

where $\triangle$ is the number of triangles and $\wedge = \sum_i \binom{k_i}{2}$ is the number of connected triples.

Homophily: Given a discrete node attribute $x : V \to \{1,\dots,C\},$ define

$$H_{\text{edge}} \;=\; \frac{1}{m}\sum_{(u,v)\in E} \mathbf{1}[x(u) = x(v)] \quad \text{(edge homophily rate).}$$

Let $p_c = \frac{|\{i\in V : x(i)=c\}|}{n}$ and $H_0 = \sum_{c=1}^{C} p_c^2$. A normalized (chance-corrected) homophily index is

$$H_{\text{norm}} \;=\; \frac{H_{\text{edge}} - H_0}{1 - H_0}.$$

D Proof for the Theorems

Since the two theorems have similar structures, we will prove them parallely in this section. We will first introduce some definations and notations and will then move to the proof.

Setting. Let $\mathcal{D}_s, \mathcal{D}_t$ be source/target node datasets drawn from graphs G_s, G_t, respectively. Each node v has an L-hop ego-subgraph $g(v)$ with feature tensor; let $\phi : \mathcal{G} \to \mathbb{R}^m$ be a (fixed) encoder, and Q a codebook with codes $\{c_1,\dots,c_K\}$. Write $Z = \phi(g)$ and $K(g) = Q(Z) \in [K]$. A predictor h maps either (i) a *feature summary* $u(g) \in \mathbb{R}^{d'}$ or (ii) a *structural embedding* $\psi(g) \in \mathcal{H}$ to a prediction; the loss ℓ is bounded in $[0,1]$.

Let $\pi_\alpha(k) = \mathbb{P}_{(g,y)\sim\mathcal{D}_\alpha}[K(g) = k]$ for $\alpha \in \{s,t\}$, and $\mathbb{P}_\alpha^k$ be the law of (g,x,y) conditional on $K(g) = k$. Define risks $\varepsilon_\alpha(h \circ Q \circ \phi) = \mathbb{E}_{\mathcal{D}_\alpha}\ell(h(Q(\phi(g))), y)$.

We also consider the *pre-quantization predictor* $\tilde{f} \;=\; h \circ \phi$ and the *post-quantization predictor* $f = h \circ Q \circ \phi$. Define the *quantization distortion*

$$\Delta_{\text{q}} := \sup_{(g,y)} \left|\ell(h(Q(\phi(g))), y) - \ell(h(\phi(g)), y)\right|.$$

Let $M(g)$ be the event that g is assigned to a code whose center lies outside a radius-τ cell around $\phi(g)$ (misassignment due to finite codebook update); set $p_{\text{mis}} = \mathbb{P}[M(g)]$.

Code-wise discrepancy metrics. For each code k:

(Feature view) Fix a 1-Lipschitz map $S : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$ (possibly identity) and suppose the composed map $u \mapsto \ell(h(u), y)$ is L_u-Lipschitz uniformly in y. Let

$$\Delta_k^{\text{feat}} \;:=\; W_1\big(\,\mathcal{L}_t(S(u) \mid k)\,,\,\mathcal{L}_s(S(u) \mid k)\,\big).$$

(Structure view) Let $\psi(g) \in \mathcal{H}$ be a bounded kernel embedding with $\|\psi(g)\|_{\mathcal{H}} \le B$; assume the function $f_\psi : \mathcal{H} \to [0,1]$ defined by $f_\psi(\psi(g)) = \mathbb{E}[\ell(h(\psi(g)), y) \mid g]$ lies in the RKHS ball $C\mathbb{B}_{\mathcal{H}}$. Define

$$\Delta_k^{\text{struct}} \;:=\; \text{MMD}_{\mathcal{H}}\big(\,\mathcal{L}_t(\psi \mid k)\,,\,\mathcal{L}_s(\psi \mid k)\,\big).$$

Additionally define the *code-marginal drift*

$$\Delta_{\text{code}} := \text{TV}(\pi_t, \pi_s) = \frac{1}{2}\sum_{k=1}^{K} |\pi_t(k) - \pi_s(k)|\,.$$

Loss class and calibration. Let $\mathcal{F} = \{\; g \mapsto \ell(h(\cdot), y) \;\}$ be the induced loss class after u or ψ. Assume a *margin-calibrated* property: there exists a non-decreasing $\Gamma : [0,1] \to [0,1]$ s.t. $|\mathbb{E}_{\mathbb{P}} f - \mathbb{E}_{\mathbb{Q}} f| \le \Gamma\big(\text{IPM}(\mathbb{P}, \mathbb{Q})\big)$ for $f \in \mathcal{F}$, where IPM $= W_1$ in the feature case, and IPM $=$ $\text{MMD}_{\mathcal{H}}$ in the structure case. For Lipschitz/$\mathcal{H}$-bounded classes we can take $\Gamma(r) = L_u r$ and $\Gamma(r) = Cr$, respectively.

Finite-sample estimation. Suppose we observe n_α i.i.d. nodes from $\mathcal{D}_\alpha$, with $n_\alpha(k)$ landing in code k. Let $\widehat{\Delta}_k^{\text{feat}}$ (resp. $\widehat{\Delta}_k^{\text{struct}}$) be empirical estimators. Assume $S(u)$ is sub-Gaussian with proxy σ^2 (per coordinate), and the kernel for ψ is bounded by B. Let $\delta \in (0,1)$.

Theorem. With probability at least $1 - \delta$, simultaneously for the feature and structure views,

$$\varepsilon_t(f) \;-\; \varepsilon_s(f) \le \underbrace{\sum_{k=1}^{K} \pi_t(k)\,\Gamma\big(\Delta_k\big)}_{\text{code-conditional shift}} \;+\; \Delta_{\text{code}} \;+\; \Delta_{\text{q}} \;+\; p_{\text{mis}}$$
$$+\; \underbrace{c_1\sqrt{\frac{\log(2K/\delta)}{\min_k n_t(k)}}}_{\text{conditional estimation}} \;+\; \underbrace{c_2\Big(\mathfrak{R}_{n_s}(\mathcal{F}) + \mathfrak{R}_{n_t}(\mathcal{F})\Big)}_{\text{function class complexity}},$$

where Δ_k equals Δ_k^{feat} in the feature view (with $\Gamma(r) = L_u r$) and equals Δ_k^{struct} in the structure view (with $\Gamma(r) = Cr$). Constants c_1, c_2 depend only on universal sub-Gaussian/kernel bounds.

Remarks. (i) The first three additive terms quantify, respectively, *within-code* conditional mismatch, *code-marginal* mismatch, and *quantization* distortion; p_{mis} captures assignment noise (e.g., stale codebook). (ii) The last two terms are finite-sample effects: conditional-IPM estimation error and richness of the induced loss class. (iii) If ϕ is L_ϕ-Lipschitz on $(\mathcal{G}, d)$ and Q has cells of diameter τ, then $\Delta_{\text{q}} \le L_\ell L_h L_\phi\, \tau$.

Proof. We start from the risk decomposition by code:

$$\varepsilon_t(f) - \varepsilon_s(f) = \sum_{k=1}^{K} \pi_t(k)\big(\mathbb{E}_{\mathbb{P}_t^k}\ell(h(c_k), y) - \mathbb{E}_{\mathbb{P}_s^k}\ell(h(c_k), y)\big) + \sum_{k=1}^{K}(\pi_t(k) - \pi_s(k))\,\mathbb{E}_{\mathbb{P}_s^k}\ell(h(c_k), y). \tag{11}$$

The second sum is bounded by $\text{TV}(\pi_t, \pi_s)$ since $\ell \in [0,1]$.

Step 1 (replace $Q \circ \phi$ by ϕ with distortion). Insert and subtract $\ell(h(\phi(g)), y)$ inside each conditional expectation. By the definition of Δ_{q} and the misassignment indicator $M(g)$,

$$\big|\mathbb{E}_{\mathbb{P}_\alpha^k}\ell(h(c_k), y) - \mathbb{E}_{\mathbb{P}_\alpha^k}\ell(h(\phi(g)), y)\big| \le \Delta_{\text{q}} + \mathbb{P}_{\mathbb{P}_\alpha^k}[M(g)] \le \Delta_{\text{q}} + p_{\text{mis}}.$$

Applying to $\alpha \in \{s, t\}$ and summing, we accrue an additive $2(\Delta_{\mathrm{q}} + p_{\mathrm{mis}})$; absorb constants to keep a single $(\Delta_{\mathrm{q}} + p_{\mathrm{mis}})$ term.

Step 2 (conditional IPM bound). Define F_k as the function class $\{(g, y) \mapsto \ell(h(\cdot), y)$ restricted to code $k\}$.

Feature view. Assume $u \mapsto \ell(h(u), y)$ is L_u-Lipschitz, uniformly in y. By Kantorovich–Rubinstein duality,

$$\left|\mathbb{E}_{\mathbb{P}_t^k} \ell(h(\phi(g)), y) - \mathbb{E}_{\mathbb{P}_s^k} \ell(h(\phi(g)), y)\right| \le L_u \, W_1\big(\mathcal{L}_t(S(u) \mid k),\, \mathcal{L}_s(S(u) \mid k)\big) = L_u \, \Delta_k^{\mathrm{feat}}.$$

Structure view. Let $\mathcal{H}$ be the RKHS with kernel $k(\cdot, \cdot)$ and unit ball $\mathbb{B}_{\mathcal{H}}$. Assume the conditional expectation functional over $\psi(g)$ lies in $C\mathbb{B}_{\mathcal{H}}$: $\ell(h(\psi(g)), y) = \langle f_k, \psi(g) \rangle_{\mathcal{H}}$ with $\|f_k\|_{\mathcal{H}} \le C$. Then by the MMD IPM property,

$$\left|\mathbb{E}_{\mathbb{P}_t^k} \ell(h(\phi(g)), y) - \mathbb{E}_{\mathbb{P}_s^k} \ell(h(\phi(g)), y)\right| \le C \, \mathrm{MMD}_{\mathcal{H}}\big(\mathcal{L}_t(\psi \mid k),\, \mathcal{L}_s(\psi \mid k)\big) = C \, \Delta_k^{\mathrm{struct}}.$$

Thus, in either view,

$$\left|\mathbb{E}_{\mathbb{P}_t^k} \ell(h(\phi(g)), y) - \mathbb{E}_{\mathbb{P}_s^k} \ell(h(\phi(g)), y)\right| \le \Gamma(\Delta_k).$$

Multiply by $\pi_t(k)$ and sum over k to control the first sum in (11).

Step 3 (finite-sample estimation of conditional IPMs). Let $\widehat{\Delta}_k$ be an empirical estimator based on $n_t(k)$ and $n_s(k)$ samples in code k.

Feature view. Assume $S(u)$ is sub-Gaussian with parameter σ^2 and bounded support radius R (w.l.o.g. by truncation). Then standard Wasserstein concentration (e.g., Bobkov–Götze type or transportation inequalities) yields, for each k and any $\eta > 0$, with probability $\ge 1 - \eta$,

$$\left|\widehat{W}_1(\widehat{\mathbb{P}}_t^k, \widehat{\mathbb{P}}_s^k) - W_1(\mathbb{P}_t^k, \mathbb{P}_s^k)\right| \le C_1 \sigma \Big(\sqrt{\tfrac{1}{n_t(k)}} + \sqrt{\tfrac{1}{n_s(k)}}\Big) \; + \; C_1' \sqrt{\tfrac{\log(1/\eta)}{\min\{n_t(k), n_s(k)\}}}.$$

A union bound over k with $\eta = \delta/(2K)$ gives the displayed c_1 term.

Structure view. For bounded kernels, MMD admits sub-Gaussian concentration; with $k(x, x) \le B^2$,

$$\left|\widehat{\mathrm{MMD}}_{\mathcal{H}} - \mathrm{MMD}_{\mathcal{H}}\right| \;\le\; C_2 B \Big(\sqrt{\tfrac{1}{n_t(k)}} + \sqrt{\tfrac{1}{n_s(k)}}\Big) + C_2' \sqrt{\tfrac{\log(1/\eta)}{\min\{n_t(k), n_s(k)\}}}.$$

Apply a union bound across k.

Step 4 (function class complexity for empirical risk plug-in). If $\varepsilon_\alpha(f)$ is replaced by empirical risks $\hat{\varepsilon}_\alpha(f)$ in (11) to obtain data-driven guarantees, standard symmetrization yields

$$\mathbb{E}\Big[\sup_{f \in \mathcal{F}} |\varepsilon_\alpha(f) - \hat{\varepsilon}_\alpha(f)|\Big] \;\le\; c \, \mathfrak{R}_{n_\alpha}(\mathcal{F}),$$

and concentration around the mean (e.g., Bousquet inequality) adds a term $O(\sqrt{\log(1/\delta)/n_\alpha})$. Since $\mathcal{F}$ is the composition of Lipschitz h, ℓ with ϕ and either u or ψ, $\mathfrak{R}_n(\mathcal{F})$ inherits Lipschitz contractions.

Collecting all pieces completes the proof. □

E RESULTS ON VQ TOKENIZER

In this section, we repeat the experiments in the main text and report the results in Figure 6, 7 and 8. Overall, we get similar observations as on RVQ, further supporting our conclusions.

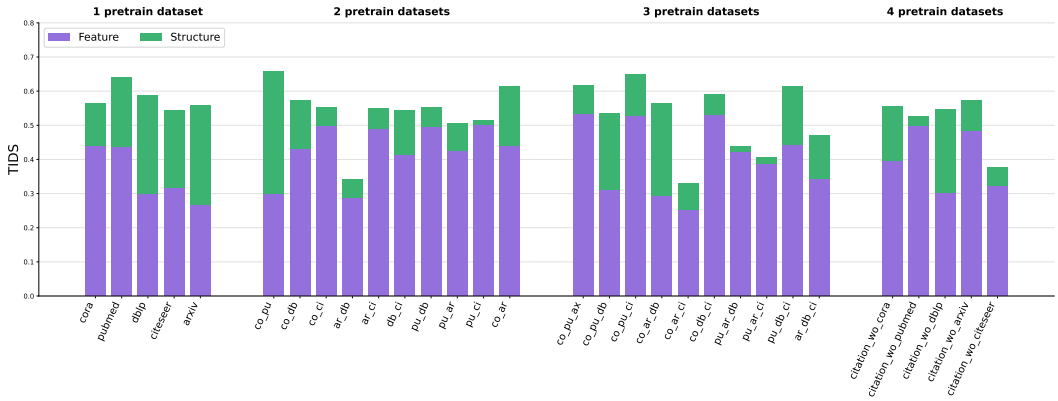

Figure 6: The distributions of GTID of VQ models on citation datasets.

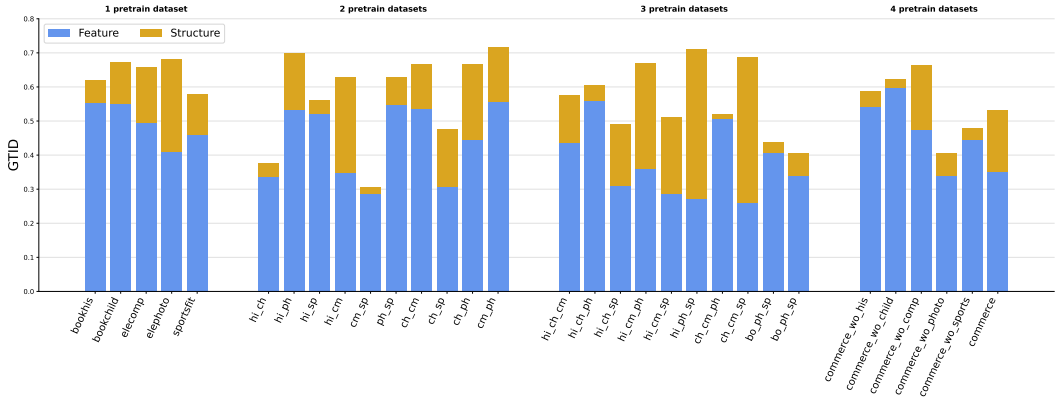

Figure 7: The distributions of GTID of VQ models on e-commerce datasets.

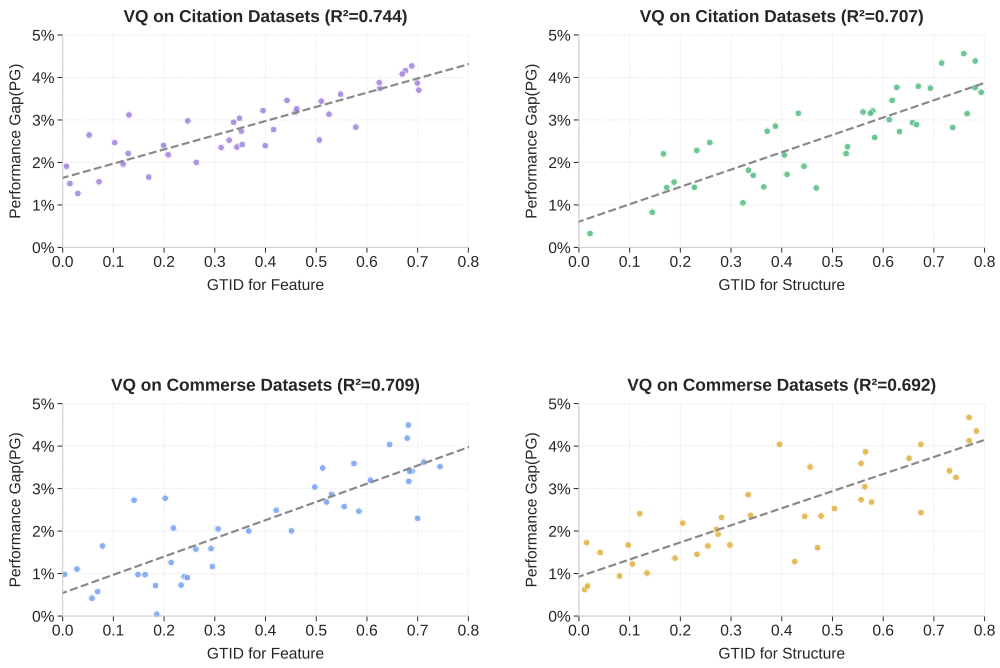

Figure 8: The correlation between GTID and the VQ model performance gap. We observe similar phenomena as RVQ tokenizers.