A Survey of Quantized Graph Representation Learning: Connecting Graph Structures with Large Language Models

Qika Lin[1], **Zhen Peng**[2], **Kaize Shi**[3], **Kai He**[1], **Yiming Xu**[2], **Jian Zhang**[2], **Erik Cambria**[4], **Mengling Feng**[1]
[1]National University of Singapore [2]Xi'an Jiaotong University
[3]University of Southern Queensland [4]Nanyang Technological University
qikalin@foxmail.com ephfm@nus.edu.sg

Abstract

Recent years have witnessed rapid advances in graph representation learning, with the continuous embedding approach emerging as the dominant paradigm. However, such methods encounter issues regarding parameter efficiency, interpretability, and robustness. As a promising solution, Quantized Graph Representation (QGR) learning has recently gained increasing interest, which encodes graph structures with discrete codes instead of conventional continuous embeddings. Given its analogous representation form to natural language, QGR also possesses the capability to seamlessly integrate graph structures with large language models (LLMs). As this emerging paradigm is still in its infancy yet holds significant promise, we undertake this thorough survey to promote its rapid future prosperity. We first present the background of the general quantization methods and their merits. An in-depth demonstration of existing QGR studies is then carried out from the perspectives of quantized strategies, training objectives, distinctive designs, knowledge graph quantization, and applications. Furthermore, we explore the strategies for code dependence learning and integration with LLMs. The final discussion on strengths and limitations aims to provide a comprehensive picture of QGR and inspire future research directions.

1 Introduction

Graph representation usually involves converting nodes, edges, or structures within various graph data into low-dimensional dense embeddings, which is called continuous graph representation (Xia et al., 2021; Lin et al., 2023) as shown in Figure 1 (a). The learned representation is obligated to preserve both node attributes and topological structures (Zhu et al., 2020). This field has made great progress in recent years, and the emergence of numerous advanced technologies has invariably drawn significant attention within the graph community, becoming focal points of extensive studies, *e.g.*, the well-known node2vec (Grover and Leskovec, 2016) model, graph neural networks (GNNs) (Kipf and Welling, 2017), and self-supervised graph learning (Liu et al., 2023c). They have achieved significant empirical success in graph-related domains, including social networks, recommendation systems, and anomaly detection (Xia et al., 2021; Liu et al., 2023c).

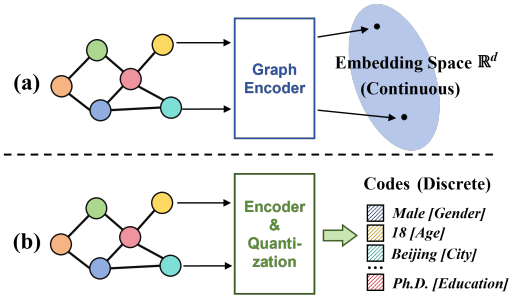

Figure 1: Illustration of different strategies for graph representations. (a) is the continuous graph representation. (b) is the quantized graph representation which represents the graph structure with discrete codes instead of conventional continuous embeddings.

Although continuous graph representation successfully extracts rich semantic information from the input graph to adapt to a wide range of machine learning and deep learning tasks, concerns may arise regarding the embedding efficiency, interpretability, and robustness (Wang et al., 2024). Specifically, with the great growth of graph scales, there will be a linear increase in embedding parameters, becoming a considerable challenge in graph learning contexts involving millions or even more nodes. Besides, continuous embeddings are generated in a black-box fashion and the meaning of the overall representation or any one dimension is unknowable, leading to a lack of interpretability. Moreover, these continuous representations are typically task-specific and show no significant abstraction at the representation level, making them

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 2765–2778
October 24-29, 2026 ©2026 Association for Computational Linguistics

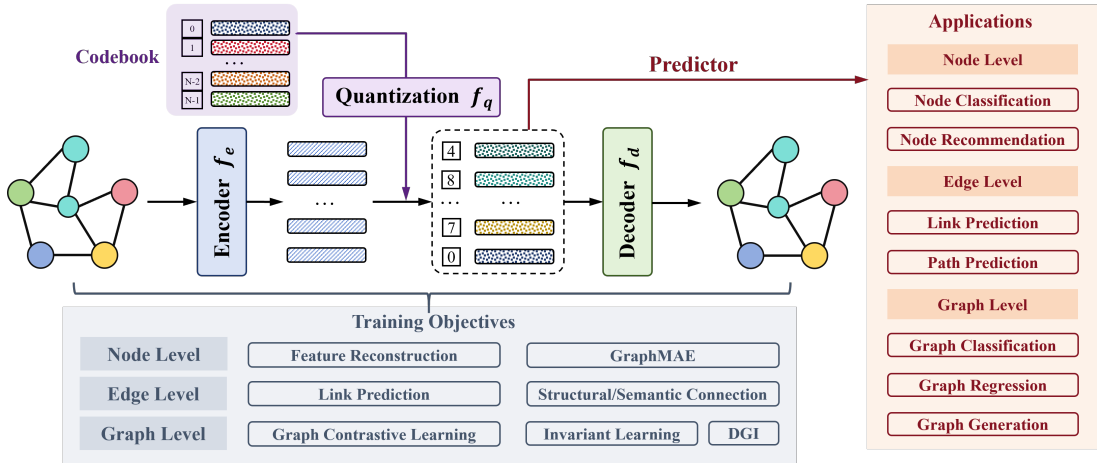

Figure 2: The general framework of the QGR studies, which mainly comprises an encoder, decoder, and quantization process. Training objectives of different levels can be utilized. By combining a predictor, multiple applications can be realized.

not robust enough to generalize to a variety of practical tasks as effectively as large language models (LLMs) (Naveed et al., 2025).

Interestingly, in real life, discrete features are often preferred for representation. For example, as shown in Figure 1 (b), descriptions affiliated to an individual are often a combination of discrete and meaningful features, such as age (*e.g.*, *18*), gender (*male* or *female*), and place of residence (*e.g.*, *Beijing*). This scheme is simple and easy to understand. Based on this intuition, quantized representation came into being, which is now a hot topic in the computer vision domain that can facilitate image generation by combining with Transformer or LLMs (Esser et al., 2021; Lee et al., 2024).

Inspired by general quantized representation techniques, quantized graph representation (QGR) learning has recently been proposed to address the aforementioned shortcomings of continuous graph learning. The core idea is to learn discrete codes for nodes, subgraphs, or the overall graph structures, instead of employing continuous embeddings. As shown in Figure 1 (b), a node can be represented as discrete codes of (*male*, *18*, *Beijing*, *Ph.D.*). In QGR methods, codes can be learned by differentiable optimization with specific graph targets or self-supervised strategies, illustrated in Figure 2. It has the capacity to learn more high-level representations due to a compact latent space, which would enhance the representation efficiency, interpretability, and robustness. Recall that when faced with graph-text scenarios, incorporating continuous embeddings into advanced LLMs is inherently challenging and prone to introducing the loss of semantic information. In contrast, due to the consistency with the discrete nature of natural language, the output of QGR would be seamlessly integrated into LLMs, like early-fusion strategy (Team, 2024). Considering that this emerging paradigm is still in its initial stage but carries substantial potential in the era of LLMs, we undertake this thorough survey. It encompasses various viewpoints, including different graph types, quantized strategies, training objectives, distinctive designs, applications and so forth. We also conclude future directions, intending to motivate future investigation and contribute to the advancement of the graph and natural language community.

To start with, we first describe the general quantization method in §2. Then, we provide a comprehensive introduction to QGR frameworks from multiple perspectives in §3. Next, we present the code dependence learning as well as the integration with LLMs in §4 and §5. Finally, we give future directions in §6.

2 General Quantization Methods

We provide a detailed summary of the merits of QGR in Appendix §A. The mainstream quantization methods can be roughly categorized into product quantization, vector quantization, finite scalar quantization, and anchor selection and assignment.

Product quantization (PQ). The core idea of PQ (Jegou et al., 2011) is to divide a high-dimensional space into multiple low-dimensional subspaces and then quantize them independently within each subspace. Specifically, it splits a high-dimensional vector into multiple smaller subvec-

tors and quantizes each subvector separately. In the quantization process, each subvector is mapped to a set of a finite number of center points by minimizing the quantization error.

Vector Quantization (VQ). To effectively learn quantized representations in a differentiable manner, the VQ strategy (van den Oord et al., 2017; Esser et al., 2021) is proposed and has emerged as the predominant method in the field. The core idea involves assigning the learned continuous representations to the codebook's index and employing the Straight-Through Estimator (Bengio et al., 2013) for effective optimization. The formulation is as follows. f_e, f_d, f_q denote the encoder, decoder, and quantization process, respectively. $\mathbf{C} \in \mathbb{R}^{M \times d}$ is the codebook representation with dimension d, which corresponds to M discrete codewords $\mathcal{C} = \{1, 2, \cdots, M\}$. $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y}\}$ is the graph, where $\mathcal{V}$ is the node set with size n and $\mathcal{E}$ is the edge set. The presence of $\mathbf{X}$ or $\mathbf{Y}$ is not guaranteed, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the input feature of a graph and $\mathbf{Y} = \{y_1, y_2, \cdots\}$ is the label set of each node. $\widetilde{\mathbf{X}} \in \mathbb{R}^{n \times d}$ is the latent representation after encoder and $\widehat{\mathbf{X}} \in \mathbb{R}^{n \times d}$ is the reconstruction feature after quantization and decoder, *i.e.*, $\widetilde{\mathbf{X}} = f_e(\mathbf{X})$, $\widehat{\mathbf{X}} = f_d(f_q(\widetilde{\mathbf{X}}))$.

Specifically, the VQ strategy yields a nearest-neighbor lookup between latent representation and prototype vectors in the codebook for quantized representations:

$$f_q(\widetilde{\mathbf{X}}_i) = \mathbf{C}_m, \; m = \arg\min_{j \in \mathcal{C}} \|\widetilde{\mathbf{X}}_i - \mathbf{C}_j\|_2^2, \tag{1}$$

indicating that node i is assigned codeword c_m. Further, the model can be optimized by the Straight-Through Estimator:

$$\mathcal{L}_{vq} = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\|\text{sg}[\widetilde{\mathbf{X}}_i] - \mathbf{C}_m\|}_{codebook} + \underbrace{\beta \|\text{sg}[\mathbf{C}_m] - \widetilde{\mathbf{X}}_i\|}_{commitment}, \tag{2}$$

where sg represents the stop-gradient operator. β is a hyperparameter to balance and is usually set to 0.25. In this loss, the first codebook loss encourages the codeword vectors to align closely with the encoder's output. The second term of the commitment loss aids in stabilizing the training process by encouraging the encoder's output to adhere closely to the codebook vectors, thereby preventing excessive deviation.

Using VQ techniques, a continuous vector can be quantized to the counterpart of a discrete code. However, there would be distinctions and differences in the vector pre and post-quantization. To address it, Residual Vector Quantization (RVQ) (Lee et al., 2022) is proposed to constantly fit residuals caused by the quantization process. RVQ is actually a multi-stage quantization method and usually has multiple codebooks. The codebook stores the residual values of the quantized vector and the original vector at each step, and the original vector can be approximated infinitely by quantizing the residual values at multiple steps.

Finite Scalar Quantization (FSQ). Because of the introduction of the codebook, there would be complex technique designs and codebook collapse in VQ techniques. Thus, FSQ (Mentzer et al., 2024) introduces a very simple quantized strategy, *i.e.*, directly *rounding* to integers rather than explicitly introducing parameters of the codebook. FSQ projects the hidden representations into a few dimensions (usually fewer than 10) and each dimension is quantized into a limited selection of predetermined values, which inherently formulates a codebook as a result of these set combinations. It is formulated as:

$$\text{FSQ}(\widetilde{\mathbf{X}}_i) = \mathcal{R}[(L-1)\sigma(\widetilde{\mathbf{X}}_i)] \in \{0, 1, \cdots, L-1\}^d, \tag{3}$$

where $\mathcal{R}$ is the rounding operation and $L \in \mathbb{N}$ is the number of unique values. The process of rounding simply adjusts a scalar to its nearest integer, thereby executing quantization in every dimension. σ is a sigmoid or tanh function. The operation is implemented with a Straight-Through Estimator:

$$\begin{aligned} \mathcal{L}_{fsq} = & \frac{1}{n} \sum_{i=1}^{n} (L-1)\sigma(\widetilde{\mathbf{X}}_i) \\ & + \text{sg}[\mathcal{R}[(L-1)\sigma(\widetilde{\mathbf{X}}_i)] - (L-1)\sigma(\widetilde{\mathbf{X}}_i)]. \end{aligned} \tag{4}$$

FSQ has the advantages of better performance, faster convergence and more stable training, particularly when dealing with large codebook sizes.

Anchor Selection and Assignment (ASA). It is typically unsupervised and often employs prior strategies to identify informative nodes within the graph, *e.g.*, the Personalized PageRank (Page, 1999) and node degree. Consequently, each node can be attributed to a combination of anchors that exhibit structural or semantic relationships, like using the shortest path. This form of quantization designates the code as a real node, unlike the aforementioned three methods.

3 Quantized Graph Learning Framework

We summarize the main studies of QGR in Table 1 from the primary perspectives of graph types, quantization methods, training strategies, and applications. Additionally, we also highlight if there are

Table 1: The summary of QGR learning studies. Abbreviations are as follows: HOG: homogeneous graph that includes attributed graph (AG), text-attributed graph (TAG), and molecular graph (MG), HEG: heterogeneous graph. For training and applications, there are STC: structure connection, SEC: semantic connection, NC: node classification, NR: node recommendation, FR: feature reconstruction, LP: Link Prediction, PP: path prediction, GC: graph classification, GR: graph regression, GG: graph generation, MCM: masked code modeling, GCL: graph contrastive learning, NTP: next-token prediction, IC: invariant learning, QA: question answering. ⊂⊃ and ◐ indicate for one-stage or multi-stage learning, respectively. In the training column, "/"split the training stages.

Model	one/p	Self-Sup	Graph Type	Method	Training	Application	Dependence	w/ LM
SNEQ (2020)	⊂⊃	✓	AG (HOG)	PQ	STC, SEC	NC, LP, NR	✗	✗
d-SNEQ (2023)	⊂⊃	✓	AG (HOG)	PQ	STC, SEC	NC, LP, NR, PP	✗	✗
Mole-BERT (2023)	◐	✓	MG (HOG)	VQ	FR/MCM, GCL	GC, GR	✓	✗
iMoLD (2023)	⊂⊃	✓	MG (HOG)	VQ	IL, GC (GR)	GC, GR	✗	✗
VQGraph (2024)	◐	✓	HOG	VQ	FR, LP/NC, Distill	NC	✗	✗
Dr.E (2025)	◐	✓	TAG (HOG)	RVQ	FR, LP, NC/FR, NTP	NC	✓	✓
STAG (2025)	◐	✓	TAG (HOG)	VQ	GraphMAE, Contrast/	NC	✗	✓
DGAE (2024)	◐	✓	HEG	VQ	LP/NTP	GG	✓	✗
LLPS (2025)	⊂⊃	✓	MG (HOG)	FSQ	Reconstruction	GG	✗	✗
GLAD (2025)	◐	✓	MG (HOG)	FSQ	FR+LP/Diffusion Bridge	GG	✓	✗
Bio2Token (2024)	⊂⊃	✓	MG (HOG)	FSQ	FR	GG	✗	✗
GQT (2024)	◐	✓	HOG	RVQ	DGI, GraphMAE2/NC	NC	✗	✗
NID (2024)	◐	✓	HOG	RVQ	GraphMAE (GraphCL)/	NC, LP, GC, GR, etc	✗	✗
UniMoT (2024)	◐	✓	MG (HOG)	VQ	FR/NTP	GC, GR, Captioning, etc.	✗	✓
NodePiece (2022)	◐	✗	KG	ASA	–/LP	LP	✗	✗
EARL (2023)	◐	✗	KG	ASA	–/LP	LP	✗	✗
RandomEQ (2023a)	◐	✗	KG	ASA	–/LP	LP	✗	✗
SSQR (2025a)	◐	✓	KG	VQ	LP, Distilling/NTP	LP, Triple Classification	✗	✓
ReaLM (2025)	◐	✓	KG	RVQ	FR/NTP	LP, Triple Classification	✗	✓
MedTok (2025)	◐	✓	KG	VQ	Contrast/	Classification, QA	✗	✓

in pipeline and self-supervised manners, as well as whether they learn code dependence and integrate with language models.

The general process of QGR is illustrated in Figure 2, which comprises the encoder f_e, decoder f_d, and quantization process f_q. The encoder f_e is to model the original graph structure into latent space, where MLPs (He et al., 2020) and GNNs (Xia et al., 2023) are usually utilized. The decoder f_d also commonly utilizes GNNs for structure embedding and the quantization process f_q implements the strategies outlined in Section 2. In particular, Bio2Token (Liu et al., 2024) utilizes Mamba (Gu and Dao, 2023) as both encoder and decoder. UniMoT (Zhang et al., 2024) leverages the pre-trained MoleculeSTM molecule encoder (Liu et al., 2023a) to connect the molecule graph encoder and causal Q-Former (Li et al., 2023b). In the following sections, we will delve into the details of the model design.

3.1 Quantized Strategies

Except for SNEQ (He et al., 2020) and d-SNEQ (He et al., 2023), the majority of research tends to employ VQ-related strategies. Beyond the general techniques in VQ, Dr.E (Liu et al., 2025) utilizes RVQ in one specific layer, which is called intra-layer residue. For more effective in encoding the structural information of the central node, it involves preserving the multi-view of a graph. Specifically, between GCN layers, the inter-layer residue is carried out to enhance the representations:

$$\mathbf{h}_v^{l+1} = \sigma\big(\mathbf{W} \cdot \mathrm{Con}(\mathbf{h}_v^l + \mathrm{Pool}(\{\mathbf{c}_{v,k}^l\}_{k=1}^K), \mathbf{h}_{\mathcal{N}(v)}^l)\big), \quad (5)$$

where $\mathbf{h}$ and $\mathbf{c}$ denote the latent representations and quantized ones. K is the number of learned codes for each layer. Using this process, the discrete codes of each node are generated in a gradual (auto-regressive) manner, following two specific orders: from previous to subsequent, and from the bottom layer to the top. This approach guarantees comprehensive multi-scale graph structure modeling and efficient code learning. iMoLD (Zhuang et al., 2023) proposes another type of *Residual* VQ, incorporating both the continuous and discrete representations, *i.e.*, $\widetilde{\mathbf{X}}_i + f_q(\widetilde{\mathbf{X}}_i)$, as the final node representation.

GLAD (Nguyen et al., 2025) and Bio2Token (Liu et al., 2024) utilize the straight-

forward approach, FSQ, to facilitate the learning of discrete codes. For KG quantization, early methods such as NodePiece (Galkin et al., 2022), EARL (Chen et al., 2023), and RandomEQ (Li et al., 2023a) have adopted the unsupervised learning approach ASA to achieve effective learning outcomes. For more condensed code representations, SSQR (Lin et al., 2025a) employs the VQ strategy to capture both the structures and semantics of KGs.

3.2 Training Objectives

In this section, we will delve deeper into training details from the perspectives of node, edge, and graph levels.

Node Level. Feature reconstruction usually serves as a simple self-supervised method for QGR learning, which is to compare the original node features with reconstructed features based on the learned codes. It mainly has the following two implementation types:

$$\mathcal{L}_{fr} = \frac{1}{n}\sum_{i=1}^{n}\left(1-\frac{\mathbf{X}_i^\top \widehat{\mathbf{X}}_i}{\|\mathbf{X}_i\|\|\widehat{\mathbf{X}}_i\|}\right)^\gamma, \mathcal{L}_{fr} = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{X}_i - \widehat{\mathbf{X}}_i\|^2, \tag{6}$$

The first involves employing cosine similarity with scaled parameter $\gamma \geq 1$, as implemented in Mole-BERT and VQGraph. The second is the mean square error as in Dr.E. NID (Luo et al., 2024) utilizes GraphMAE (Hou et al., 2022) for self-supervised learning, which can be viewed as a masked feature reconstruction. It involves selecting a subset of nodes, masking the node features, encoding by a message-passing network, and subsequently reconstructing the masked features with a decoder.

Edge Level. It is a primary strategy to acquire the graph structures in QGR codes using a self-supervised paradigm. The direct link prediction, *i.e.*, edge reconstruction, is a widely adopted technique to assess the presence or absence of each edge before and after the reconstruction process as:

$$\begin{gathered}\mathcal{L}_{lp} = \|\mathbf{A} - \sigma(\widehat{\mathbf{X}} \cdot \widehat{\mathbf{X}}^\top)\|^2, \\ \mathcal{L}_{lp} = -\frac{1}{|\mathcal{E}|}\sum_{j=1}^{|\mathcal{E}|}\left[y_j \log(\hat{y}_j) + (1-y_j)\log(1-\hat{y}_j)\right],\end{gathered} \tag{7}$$

where the mean square error (*e.g.*, VQGraph) and the binary cross-entropy are utilized to optimize (*e.g.*, Dr.E), respectively. $\hat{y}_j$ is the prediction of probability for the existence of the edge based on the learned representations. $y_j \in \{0, 1\}$ is the ground label.

Beyond direct edge prediction, He et al. (2020) proposes high-level connection prediction between two nodes, *i.e.*, structural connection, and semantic connection. The former aims to accurately maintain the shortest distances δ within the representations:

$$\mathcal{L}_{stc} = \frac{1}{n}\sum_{(i,j,k)}^{n}\max\big(D_{i,j} - D_{i,k} + \delta_{i,j} - \delta_{i,k}, 0\big), \tag{8}$$

where D is to calculate the distance of two nodes based on the learned codes. Considering that nodes sharing identical labels ought to be situated closer together within the embedding space, semantic connection is proposed:

$$\mathcal{L}_{sec} = \frac{1}{n \cdot T}\sum_{i}^{n}\sum_{i,j=1}^{T}(D_{i,j} - S_{i,j})^2, \tag{9}$$

where T is the sample size. S presents the constant semantic margin, which is set to 0 if the labels of two nodes are totally distinct, otherwise, it is set to a constant.

Graph Level. For high-level representations, the graph modeling targets are proposed. For instance, graph contrastive learning (van den Oord et al., 2018) is a widely utilized method for modeling relationships between graph pairs:

$$\mathcal{L}_{gcl} = -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{G}\in\mathcal{D}}\log\frac{e^{sim(\mathbf{h}_{g1},\mathbf{h}_{g2})/\tau}}{\sum_{\mathcal{G}'\in\mathcal{B}} e^{sim(\mathbf{h}_{g1},\mathbf{h}_{\mathcal{G}'})/\tau}}, \tag{10}$$

where $\mathcal{D}$ is the dataset that contains all graphs and $\mathbf{h}_g$ is the graph representation based on $f_q(\widetilde{\mathbf{X}})$. $g1$ and $g2$ can be seen as positive samples for the target graph, which are usually transformed by the target graph $\mathcal{G}$. For example, Mole-BERT (Xia et al., 2023) utilizes different masking ratios (*e.g.*, 15% and 30%) for molecular graphs to obtain $g1$ and $g2$. $\mathcal{B}$ is contrastive samples, typically derived from the sampled batch. GraphCL (You et al., 2020) is also a well-known framework of this kind based on graph augmentations, including node dropping, edge perturbation, attribute masking, and subgraph, which is used by NID (Luo et al., 2024).

Furthermore, motivated by a straightforward self-supervised learning framework that identifies varied augmentation views as similar positive pairs, iMoLD (Zhuang et al., 2023) considers the learned invariant and global aspects (invariant plus spurious) as positive pairs, aiming to enhance their similarity. An MLP-based predictor is introduced to transform the output of one view and align it

with the other. The whole process can be interpreted as *invariant learning*. Deep Graph Infomax (DGI) (Velickovic et al., 2019) can also be used as a high-level QGR learning method, which is a contrastive approach that compares local (node) representations with global (graph or sub-graph) ones. GQT (Wang et al., 2024) utilize both high-level DGI (Velickovic et al., 2019) and node-level GraphMAE2 (Hou et al., 2023) as training targets.

3.3 Distinctive Designs

Codebook Design. Beyond randomly setting the codebook for updating, there are several specifically designed strategies for domain knowledge adaption and generalization. For example, Dr.E (Liu et al., 2025) sets the vocabulary of LLaMA as the codebook, realizing token-level alignment between GNNs and LLM. In this way, each node of graphs can be quantized to a permutation of language tokens and then can be directly input to LLMs to make predictions. When modeling molecular graphs, Mole-BERT (Xia et al., 2023) categorizes the codebook embeddings into various groups, each representing a distinct type of atom. For instance, the quantized codes of carbon, nitrogen and oxygen are confined to different groups.

Training Pipelines. Aside from the single-stage learning process applied to QGR and particular tasks, some methods incorporate multiple stages into the training framework. GQT (Wang et al., 2024) modulate the learned codes through hierarchical encoding and structural gating, which are subsequently fed into the Transformer network and aggregate the learned representations through an attention module. Based on the learned quantized representation, NID (Luo et al., 2024) trains an MLP network for downstream tasks, such as node classification and link prediction.

3.4 KG Quantization

Recently, several KG quantization methods have been proposed for effective embedding, responding to the increasing demand for larger KGs. Unlike the techniques employed in the HOG or HEG setting, these methods tend to leverage an unsupervised paradigm for quantization. NodePiece (Galkin et al., 2022), EARL (Chen et al., 2023), and random entity quantization (RandomEQ for short) (Li et al., 2023a) are all in such a framework. They first select a number of entities as anchors and then utilize the structural statistical strategy to match them for each entity. Specifically, NodePiece employs metrics such as Personalized PageRank (Page, 1999) and node degree to select certain anchor entities and subsequently view top-k closest anchors as entity codes. EARL and RandomEQ sample 10% entities as anchors and then assign soft weights to each anchor according to the similarity between connected relation sets. Given the limitations of these methods in thoroughly capturing the structure and semantics of KGs, SSQR (Lin et al., 2025a) presents a self-supervised approach that employs learned discrete codes to reconstruct structures and imply semantic text, offering more holistic modeling with only 16 quantized codes. Furthermore, ReaLM (Guo et al., 2025) introduces an RVQ framework that compresses high-dimensional and continuous KG embeddings (pretrained RotatE (Sun et al., 2019) embeddings) into compact and discrete code sequences while effectively preserving rich relational and semantic information. More detailed comparison between SSQR and ReaLM are shown in Appendix §C.

3.5 Application Scenarios

General Graph Tasks. Based on the learned quantized codes, many general graph tasks can be addressed, for example, node classification, link prediction, graph classification, graph regression, and graph generation. In addition, SNEQ (He et al., 2020) can be employed for node recommendation that ranks all nodes based on a specific distance metric and suggests the nearest node. d-SNEQ (He et al., 2023) is further utilized for path prediction that predicts the path (*e.g.*, the shortest path) between two nodes.

Based on the learned structure-aware codes for nodes, VQGraph (Yang et al., 2024) enhances the GNN-to-MLP distillation by proposing a new distillation target, namely *soft code assignments*, which utilizes the Kullback–Leibler divergence to make two distributions (codes distributions by GNN and MLP) be close together. This can directly transfer the structural knowledge of each node from GNN to MLP. The results show it can improve the expressiveness of existing graph representation space and facilitate structure-aware GNN-to-MLP distillation. In the KG scenarios, NodePiece (Galkin et al., 2022), EARL (Chen et al., 2023), RandomEQ (Li et al., 2023a), SSQR (Lin et al., 2025a) all can be used for KG link prediction, which is a ranking task

that predicts the object entity based on the given subject entity and the relation, *i.e.*, $(s, r, ?)$. Using the learned codes as features, SSQR can also address the triple classification problem, predicting the validity of the given triple (s, r, t).

Molecular Tasks & AI for Science. Recently, QGR methods have been widely used for molecular tasks and AI for science scenarios (Chen et al., 2024), including molecular property prediction (classification and regression), molecule-text prediction, and protein & RNA reconstruction. Specifically, DGAE (Boget et al., 2024) conduct the graph generation. It first iteratively samples discrete codes and then generates the graph structures by the pre-trained decoder, which can be used in the generation for molecular graphs and has the potential for drug design, material design and protein design. LLPS (Donà et al., 2025) shows the potential for the reconstruction of protein sequences by training a de novo generative model for protein structures using a vanilla decoder-only Transformer model. Bio2Token (Liu et al., 2024) conducts efficient representation of large 3D molecular structures with high fidelity for molecules, proteins, and RNA, holding the potential for the design of biomolecules and biomolecular complexes. UniMoT (Zhang et al., 2024) unifies the molecule-text applications, including molecular classification & regression, molecule captioning, molecule-text retrieval, and caption-guided molecule generation.

4 Code Dependence Learning

Inspired by the distribution learning of the discrete codes in computer vision field (Esser et al., 2021) that predicts the next code by the auto-regressive Transformer, QGR methods also implement this technique, facilitating comprehensive semantic modeling and supporting the generation tasks. Similar to BERT (Devlin et al., 2019) pre-training style, MOLE-BERT adopts a strategy similar to Masked Language Modeling (MLM). It employs this method to pre-train the GNN encoder by randomly masking certain discrete codes. Subsequently, it pre-trains GNNs to predict these masked codes, a process known as Masked Code Modeling (MCM):

$$\mathcal{L}_{mcm} = -\sum_{\mathcal{G}\in\mathcal{D}} \sum_{j\in\mathcal{M}} \log p(c_j|\mathcal{G}_{\mathcal{M}}), \tag{11}$$

where $\mathcal{M}$ is the set of masked nodes in the graph and c_j is the quantized codes. Auto-regressive code dependence learning is another mainstream strategy

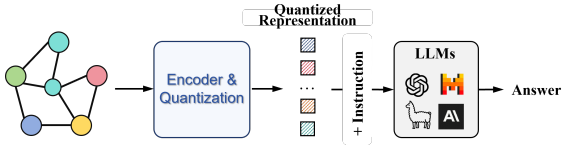

Figure 3: Illustration of integrating QGR with LLMs.

(usually using Transformer) based on next token prediction:

$$\mathcal{L}_{ntp} = -\frac{1}{N}\prod_{j=1}^{N} \log P_\theta(\boldsymbol{c}_j|\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_{j-1}), \tag{12}$$

where network P_θ is with parameter θ. N is the length of the code sequence $\boldsymbol{c}$. DGAE (Boget et al., 2024) splits the latent representation of each node into C parts. So after the quantization, each graph can be represented as $n \times C$ code permutation. To learn the dependence of it, the 2D Transformer (Vaswani et al., 2017) is introduced to auto-regressively generate the codes, *i.e.*, the joint probability can be $\prod_{i=1}^{n}\prod_{j=1}^{C} P_\theta(\boldsymbol{c}_{i,j}|\boldsymbol{c}_{<i,1}, \cdots, \boldsymbol{c}_{<i,C}, \boldsymbol{c}_{i,<j})$.

Moreover, GLAD (Nguyen et al., 2025) implements diffusion bridges (Liu et al., 2023b) for codes' dependence, where Brownian motion is utilized as a non-conditional diffusion process defined by a stochastic differential equation (SDE). Based on the learned model bridge, discrete practical codes can be acquired by the iterative denoising process from the random Gaussian noise. Followed by the pre-trained decoder, these codes can be used for the graph generation. Although Dr.E does not directly or explicitly learn the dependence of the codes, it introduces intra-layer and inter-layer residuals to gradually generate the codes, which enhances the representation of sequential information as the newly generated code depends on the previously obtained codes. It can be viewed as an implicit auto-regressive manner.

5 Integration with LLMs

The quantized codes, being discrete, share a similar structure with natural language. As such, QGR methods can be seamlessly incorporated with LLMs to facilitate robust modeling and generalization as shown in Figure 3. Table 2 provides examples of tuning instructions for both Dr.E and SSQR, arranged for better understanding and intuitive interpretation.

Dr.E implements token-level alignment between GNNs and LLMs. Based on the quantized codes

Input: Given a node, you need to classify it among 'Case Based', 'Genetic Algorithms'.... With the node's 1-hop information being 'amass', 'traverse', 'handle'..., 2-hop information being 'provable', 'revolution', 'creative'..., 3-hop information being 'nous', 'hypothesis', 'minus', the node should be classified as:
Output: Rule Learning

(a) Node classification, taken from Dr.E (Liu et al., 2025).

Input: Given a triple in the knowledge graph, you need to predict its validity based on the triple itself and entities' quantized representations.
The triple is: (h, r, t)
The quantized representation of entity h is: [Code(h)]
The quantized representation of entity t is: [Code(t)]
Please determine the validity of the triple and respond True or False.
Output: True/False

(b) KG triple classification, from SSQR (Lin et al., 2025a).

Table 2: Instruction examples with learned the discrete codes. In (a), the codeword corresponds to the real word in the LLM's vocabulary. In (b), the learned codewords are virtual tokens, which need to expand the LLM's vocabulary for fine-tuning.

through intra-layer and inter-layer residuals, the preservation of multiple perspectives in each convolution step is guaranteed, fostering reliable information transfer from the surrounding nodes to the central point. Moreover, by utilizing the LLMs' vocabulary as the codebook, Dr.E can represent each node with real words from different graph hops, as shown in Table 2 (a). By fine-tuning LLaMA-2-7B (Touvron et al., 2023) with the designed instruction data, the node label can be easily predicted. Different from the fixed vocabulary in Dr.E, UniMoT (Zhang et al., 2024) expands the original LLMs' vocabulary through the incorporation of learned molecular tokens from the codebook, forming a unified molecule-text vocabulary. Based on the specific instructions for molecular property prediction, molecule captioning, molecule-text retrieval, caption-guided molecule generation, LLaMA2 is fine-tuned with LoRA (Hu et al., 2022), thereby bestowing upon it an impressive and comprehensive proficiency in molecule-text applications. Similarly, in KG-related scenarios, SSQR leverages the learned discrete entity codes as additional features to directly input LLMs, helping to make accurate predictions. Both LLaMA2-7B and LLaMA3.1-8B (Dubey et al., 2024) are utilized for the KG link prediction task and triple classification task. During the fine-tuning phase, the introduction of new tokens necessitates the expansion of the LLM's tokenizer vocabulary, which is in line with the size of the codebook, *e.g.*, 1024 or 2048. All other elements of the network remain unaltered. Similarly, ReaLM employs LoRA fine-tuning on Llama3.2 to adapt the model for KG tasks.

6 Future Directions

Based on this systematic review, we identify five key directions for future research.

Choice of Codeword. For QGR learning, one can incorporate certain domain knowledge into the construction of the codebook. This knowledge might encompass semantic concepts, structural formations, and textual language. Such an approach can augment the interpretability of the whole framework, facilitating a better human understanding.

Advanced Quantized Methods. QGR can leverage cutting-edge VQ methods to enhance the effectiveness and efficiency of graph codes, drawing on techniques that have already proven successful in the computer vision domain. For instance, the rotation trick (Fifty et al., 2024) is a proven method to mitigate the issues of codebook collapse and underutilization in VQ. SimVQ (Zhu et al., 2024) simply incorporates a linear transformation into the codebook, yielding substantial improvement.

Unified Graph Foundation Model. While the QGR methods have indeed made some notable strides, the current research primarily follows a schema that tackles various tasks using distinct training targets, resulting in multiple models. This diversity may restrict its applicability, particularly in the age of LLMs. Drawing inspiration from the remarkable success of unified LLMs and multimodal LLMs (Zhao et al., 2023; Lin et al., 2025b), the unified graph foundation model could be realized to execute various tasks across diverse graphs through QGR, by learning unified codes for different graphs and then tuning with LLM techniques (Ji et al., 2025).

Graph RAG with QGR. Retrieval-Augmented Generation (RAG) has emerged as a significant focal point for improving the capabilities of LLMs within specific fields. Graph RAG (Edge et al., 2024; Luo et al., 2025) would be beneficial for downstream tasks in the context of LLMs, where the QGR learning would provide an effective manner to retrieve relevant nodes or subgraphs by calculating the similarity of the codes. It can be easily realized by the statistical metrics of the codes or the numerical calculations among embeddings.

Long-Term Memory with QGR. Memory mechanisms have become increasingly important for LLM agents to support long-horizon interaction and continual adaptation (Zhang et al., 2026a; Yan et al., 2026). Recent studies have explored structured and graph-based memory to organize historical experiences and capture their complex relationships. QGR provides a promising approach to compress such graph-structured memories into compact discrete codes, enabling efficient storage, code-level retrieval, and seamless integration with LLMs. Future research could explore dynamically evolving quantized memory graphs that preserve temporal, relational, and provenance information, while jointly optimizing memory updating, forgetting, and retrieval to support accurate and efficient long-term reasoning.

Limitations

Beyond the above methods, there are also some methods that do not use quantized code for the final representation, such as VQ-GNN (Ding et al., 2021), VQSynery (Wu et al., 2024), and MSPmol (Lu et al., 2024). For example, instead of employing discrete motif features as the ultimate representation, MSPmol (Lu et al., 2024) utilizes the Vector Quantization (VQ) method on motif nodes within specific layers of the molecular graph. There are also benefits for the representation, which can be viewed as a paradigm of *continuous+quantized* with the network, rather than the outputs. They differ from the studies in Table 1, so we do not include them in this survey.

Despite some achievements of the QGR methods, there are still some limitations from both methodology and application perspectives (Li et al., 2025). First, the quantization process may result in the loss of some details of the original data, which could pose challenges in areas that require accurate modeling. Second, the quantization process requires complex techniques to ensure effective and efficient optimization, while pipeline's schema and integration with LLMs further increase the complexity of the overall framework. Third, compared with the wide application and success of general LLMs, efforts to integrate QGR and LLMs still need to be carried out. Leveraging multi-agent frameworks is a compelling direction (Zhang et al., 2026b,c), as it could learn semantic representations to substantially refine the planning and collaborative reasoning capabilities of these systems.

Acknowledgments

This project was funded by the National Medical Research Council (NMRC) Healthy and Meaningful Longevity - Cognition Grant (NICCOG2024-0028), the AI for Public Health (AI4PH) Program, Saw Swee Hock School of Public Health, National University of Singapore (NUS) and the NUS Innovation Venture Creation (IVC) - Provost's Innovation Chair Professor Awards.

References

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Jianyuan Bo, Hao Wu, and Yuan Fang. 2025. Quantizing text-attributed graphs for semantic-structural integration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 107–118. ACM.

Yoann Boget, Magda Gregorova, and Alexandros Kalousis. 2024. Discrete graph auto-encoder. *Transactions on Machine Learning Research*.

Mingyang Chen, Wen Zhang, Zhen Yao, Yushan Zhu, Yang Gao, Jeff Z. Pan, and Huajun Chen. 2023. Entity-agnostic representation learning for parameter-efficient knowledge graph embedding. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 4182–4190.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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 (NAACL-HLT)*, pages 4171–4186.

Mucong Ding, Kezhi Kong, Jingling Li, Chen Zhu, John Dickerson, Furong Huang, and Tom Goldstein. 2021. Vq-gnn: A universal framework to scale up graph neural networks using vector quantization. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:6733–6746.

Jérémie Donà, Benoit Gaujac, Timothy Atkinson, Liviu Copoiu, Thomas Pierrot, and Thomas D. Barrett. 2025. Learning the language of protein structure. *Transactions on Machine Learning Research*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman,

Akhil Mathur, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130.

Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883.

Christopher Fifty, Ronald G. Junkins, Dennis Duan, Aniketh Iger, Jerry W. Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. 2024. Restructuring vector quantization with the rotation trick. *CoRR*, abs/2410.06424.

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

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752.

Wenbin Guo, Xin Wang, Jiaoyan Chen, Lingbing Guo, Zhao Li, and Zirui Chen. 2025. Realm: Residual quantization bridging knowledge graph embeddings and large language models. *CoRR*, abs/2510.09711.

Tao He, Lianli Gao, Jingkuan Song, and Yuan-Fang Li. 2023. Semisupervised network embedding with differentiable deep quantization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):4791–4802.

Tao He, Lianli Gao, Jingkuan Song, Xin Wang, Kejie Huang, and Yuanfang Li. 2020. Sneq: Semi-supervised attributed network embedding with attention-based quantisation. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 4091–4098.

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

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Yanbiao Ji, Chang Liu, Xin Chen, Dan Luo, Mei Li, Yue Ding, Wenqing Lin, and Hongtao Lu. 2025. From anchors to answers: A novel node tokenizer for integrating graph structure into large language models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1124–1134. ACM.

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

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11513–11522. IEEE.

Suhyeon Lee, Won Jun Kim, Jinho Chang, and Jong Chul Ye. 2024. LLM-CXR: instruction-finetuned LLM for CXR image understanding and generation. In *International Conference on Learning Representations (ICLR)*.

Jiaang Li, Quan Wang, Yi Liu, Licheng Zhang, and Zhendong Mao. 2023a. Random entity quantization for parameter-efficient compositional knowledge graph representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2917–2928.

Jindong Li, Yali Fu, Jiahong Liu, Linxiao Cao, Wei Ji, Menglin Yang, Irwin King, and Ming-Hsuan Yang. 2025. Discrete tokenization for multimodal llms: A comprehensive survey. *CoRR*, abs/2507.22920.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023b. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, pages 19730–19742.

Qika Lin, Jun Liu, Lingling Zhang, Yudai Pan, Xin Hu, Fangzhi Xu, and Hongwei Zeng. 2023. Contrastive graph representations for logical formulas embedding. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3563–3574.

Qika Lin, Tianzhe Zhao, Kai He, Zhen Peng, Fangzhi Xu, Ling Huang, Jingying Ma, and Mengling Feng. 2025a. Self-supervised quantized representation for seamlessly integrating knowledge graphs with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13587–13602.

Qika Lin, Yifan Zhu, Xin Mei, Ling Huang, Jingying Ma, Kai He, Zhen Peng, Erik Cambria, and Mengling Feng. 2025b. Has multimodal learning delivered universal intelligence in healthcare? a comprehensive survey. *Information Fusion*, page 102795.

Andrew Liu, Axel Elaldi, Nathan Russell, and Olivia Viessmann. 2024. Bio2token: All-atom tokenization of any biomolecular structure with mamba. *CoRR*, abs/2410.19110.

Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Animashree Anandkumar. 2023a. Multi-modal molecule structure-text model for text-based retrieval and editing. *Nature Machine Intelligence*, 5(12):1447–1457.

Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. 2023b. Learning diffusion bridges on constrained domains. In *International Conference on Learning Representations (ICLR)*.

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2023c. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5879–5900.

Zipeng Liu, Likang Wu, Ming He, Zhong Guan, Hongke Zhao, and Nan Feng. 2025. Multi-view empowered structural graph wordification for language models. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 24714–24722.

Yangyi Lu, Jing Peng, Yifeng Zhu, and Zhuang Chen. 2024. Pre-training molecular graph representations with motif-enhanced message passing. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Ze-min Kuang, Meina Song, Yifan Zhu, and Luu Anh Tuan. 2025. Hypergraphrag: Retrieval-augmented generation with hypergraph-structured knowledge representation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuankai Luo, Qijiong Liu, Lei Shi, and Xiao-Ming Wu. 2024. Structure-aware semantic node identifiers for learning on graphs. *CoRR*, abs/2405.16435.

Jingying Ma, Feng Wu, Qika Lin, Yucheng Xing, Chenyu Liu, Ziyu Jia, and Mengling Feng. 2025. Codebrain: Bridging decoupled tokenizer and multi-scale architecture for EEG foundation model. *CoRR*, abs/2506.09110.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations (ICLR)*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):106:1–106:72.

Van Khoa Nguyen, Yoann Boget, Frantzeska Lavda, and Alexandros Kalousis. 2025. GLAD: improving latent graph generative modeling with simple quantization. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 19695–19702.

Lawrence Page. 1999. The pagerank citation ranking: Bringing order to the web. Technical report.

Xiaorui Su, Shvat Messica, Yepeng Huang, Ruth Johnson, Lukas Fesser, Shanghua Gao, Faryad Sahneh, and Marinka Zitnik. 2025. Multimodal medical code tokenizer. In *Forty-second International Conference on Machine Learning (ICML)*. OpenReview.net.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations (ICLR)*.

Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *CoRR*, abs/2405.09818.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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

Limei Wang, Kaveh Hassani, Si Zhang, Dongqi Fu, Baichuan Yuan, Weilin Cong, Zhigang Hua, Hao Wu, Ning Yao, and Bo Long. 2024. Learning graph quantized tokenizers for transformers. *CoRR*, abs/2410.13798.

Jiawei Wu, Mingyuan Yan, and Dianbo Liu. 2024. Vqsynery: Robust drug synergy prediction with vector quantization mechanism. *CoRR*, abs/2403.03089.

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

Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. 2023. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *International Conference on Learning Representations (ICLR)*.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2024. Symbol-llm: Towards foundational symbol-centric interface for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13091–13116.

Hang Yan, Fangzhi Xu, Qiushi Sun, Jinyang Wu, Zixian Huang, Muye Huang, Jingyang Gong, Zichen Ding, Kanzhi Cheng, Yian Wang, et al. 2026. Odysseyarena: Benchmarking large language models for long-horizon, active and inductive interactions. *arXiv preprint arXiv:2602.05843*.

Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, CUI Bin, Muhan Zhang, and Jure Leskovec. 2024. Vqgraph: Rethinking graph representation space for bridging gnns and mlps. In *International Conference on Learning Representations (ICLR)*.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2024. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403.

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

Kaiwen Zha, Lijun Yu, Alireza Fathi, David A. Ross, Cordelia Schmid, Dina Katabi, and Xiuye Gu. 2025. Language-guided image tokenization for generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15713–15722. Computer Vision Foundation / IEEE.

Jian Zhang, Yu He, Zhiyuan Wang, Zhangqi Wang, Kai He, Fangzhi Xu, Qika Lin, and Jun Liu. 2026a. A3-bench: Benchmarking memory-driven scientific reasoning via anchor and attractor activation. *arXiv preprint arXiv:2601.09274*.

Jian Zhang, Zhangqi Wang, Haiping Zhu, Kangda Cheng, Kai He, Bo Li, Qika Lin, Jun Liu, and Erik Cambria. 2026b. Mars: Multi-agent adaptive reasoning with socratic guidance for automated prompt optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jian Zhang, Zhiyuan Wang, Zhangqi Wang, Fangzhi Xu, Qika Lin, Lingling Zhang, Rui Mao, Erik Cambria, and Jun Liu. 2026c. Maps: Multi-agent personality shaping for collaborative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Juzheng Zhang, Yatao Bian, Yongqiang Chen, and Quanming Yao. 2024. Unimot: Unified molecule-text language model with discrete token representation. *CoRR*, abs/2408.00863.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, and et al. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *CoRR*, abs/2006.04131.

Yongxin Zhu, Bocheng Li, Yifei Xin, and Linli Xu. 2024. Addressing representation collapse in vector quantized models with one linear layer. *CoRR*, abs/2411.02038.

Xiang Zhuang, Qiang Zhang, Keyan Ding, Yatao Bian, Xiao Wang, Jingsong Lv, Hongyang Chen, and Huajun Chen. 2023. Learning invariant molecular representation in latent discrete space. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Merits of Quantized Graph Representation

Compared to continuous representation methods, QGR offers a series of distinct advantages that fertilize its applications.

Embedding Parameter Efficiency. As large-scale graphs become ubiquitous in reality, there is a corresponding substantial increase in the representation parameters, which necessitates abundant memory usage for storage and substantial computational expenses (Galkin et al., 2022). Supposing n as the number of nodes and d as the dimensions, the required embedding parameters would be $n \times d$. In the quantized settings, the total count of parameters would reduce to $n \times m + M \times d$, where M and m denote the length of the codeword set and the number of codes assigned to each node. For example, if there are 10^6 nodes with 1024 feature dimensions, 1024 codewords, and 8 assigned codes, it would be a 113-fold reduction in required representation parameters.

Explainability and Interpretability. The discrete nature of QGR brings the explainability and interpretability for embeddings and reasoning, as shown in Figure1 (b). Each code would be assigned to the practical item by specific designs for direct interpretability, *e.g.*, Dr. E (Liu et al., 2025) introduces a language vocabulary as the code set and each node in graphs can be represented as the permutation of explainable language tokens.

Table 3: Case study of SSQR. Taken from (Lin et al., 2025a)

Input:
This is a knowledge graph completion task, which needs to predict the tail entity for an incomplete query triplet.
The query triplet is (Valparaiso University, inverse relation of /location/location/contains, ?).
The quantized representation of entity Valparaiso University is [527] [1345] [1849] [1227] [1751] [2038] [818] [515] [1417] [333] [29] [721] [1691] [798] [1033] [153]
The answer candidates and corresponding quantized representations are as follows:
Minnesota, [1532] [258] [1837] [357] [923] [1994] [638] [555] [771] [1003] [1736] [1473] [1495] [1436] [1313] [20]
New York, [661] [1243] [542] [1741] [1907] [1799] [858] [1794] [1916] [458] [1844] [909] [438] [1737] [686] [963]
California, [1059] [1286] [1604] [846] [1086] [451] [1087] [1794] [994] [297] [1463] [159] [556] [1836] [407] [963]
Massachusetts, [202] [1243] [977] [757] [304] [389] [1172] [1308] [1916] [1858] [1323] [11] [841] [1680] [1798] [1885]
Illinois, [961] [1025] [1267] [174] [643] [1951] [1742] [1794] [1720] [1481] [543] [1883] [695] [1921] [182] [963]
New York City, [1458] [326] [1707] [239] [151] [640] [1366] [1794] [610] [458] [1844] [932] [122] [311] [121] [868]
United Kingdom, [51] [193] [1354] [669] [1867] [881] [480] [1271] [392] [1858] [650] [909] [1503] [1126] [1550] [153]
Pennsylvania, [361] [825] [1052] [1655] [1670] [732] [951] [1569] [275] [1995] [543] [4] [753] [351] [331] [637]
Los Angeles, [1584] [1231] [1707] [1461] [1867] [1466] [265] [1933] [850] [1533] [805] [1128] [1824] [1823] [307] [963]
Florida, [2016] [326] [542] [1614] [462] [1433] [1388] [819] [926] [1289] [1321] [563] [1977] [1144] [1268] [662]
Ohio, [1643] [1889] [1604] [88] [1364] [485] [1819] [1569] [54] [1582] [1500] [411] [438] [125] [1636] [20]
Texas, [2012] [1845] [1207] [412] [531] [1394] [1004] [688] [653] [1671] [1790] [1690] [1732] [1686] [1721] [1205]
Virginia, [99] [825] [738] [1859] [1287] [1540] [708] [780] [653] [662] [756] [1873] [1514] [1686] [59] [409]
England, [848] [1220] [1052] [590] [175] [451] [529] [1933] [808] [1598] [1790] [1988] [670] [524] [121] [136]
Connecticut, [633] [1025] [672] [338] [1694] [1799] [1528] [1177] [1949] [458] [1703] [411] [395] [33] [233] [1013]
Indiana, [497] [1875] [1849] [377] [1694] [61] [1471] [1445] [392] [1672] [1500] [300] [711] [1839] [331] [136]
Los Angeles County, [662] [1772] [558] [1623] [304] [1755] [1388] [1794] [1554] [333] [1662] [1258] [1474] [1764] [1021] [409]
Louisiana, [1584] [326] [1309] [50] [1035] [0] [546] [728] [1130] [297] [1284] [11] [1514] [1929] [1248] [136]
Michigan, [172] [326] [382] [174] [1364] [1774] [1165] [664] [994] [1480] [1923] [431] [816] [1587] [1443] [860]
Hennepin County, [584] [1890] [1263] [1168] [1990] [656] [784] [1735] [102] [1533] [1979] [1005] [1312] [786] [873] [1294]
Please generate quantized representations of the **top-3** potential answer entities, ranked from highest to lowest:

LLM Output: 1, [497] [1875] [1849] [377] [1694] [61] [1471] [1445] [392] [1672] [1500] [300] [711] [1839] [331] [136]
2, [1532] [258] [1837] [357] [923] [1994] [638] [555] [771] [1003] [1736] [1473] [1495] [1436] [1313] [20]
3, [661] [1243] [542] [1741] [1907] [1799] [858] [1794] [1916] [458] [1844] [909] [438] [1737] [686] [963]

Robustness and Generalization. The discrete tokens of the QGR are more robust and generalizable, which usually utilizes the self-supervised strategies to learn both local-level and high-level graph structures (*e.g.*, edge reconstruction and DGI (Velickovic et al., 2019)) as well as semantics (*e.g.*, feature reconstruction). Moreover, the acquired tokens can readily adapt to downstream applications directly or indirectly (Yang et al., 2024).

Seamless Integration with NLP Models. The swift advancement in natural language processing (NLP) techniques, particularly LLMs (Xu et al., 2024), has sparked an increased interest in employing NLP models to resolve graph tasks. However, the inherent representation gap between the typical graph structure and natural language poses a significant challenge to their seamless and effective integration. Graph quantization, by learning discrete codes that bear resemblance to natural language forms, can facilitate this integration directly and seamlessly (Lin et al., 2025a; Liu et al., 2025).

B General Framework of QGR Studies

The general framework of the QGR studies is illustrated in Figure 2, which mainly comprises an encoder f_e, decoder f_d, and quantization process f_q. The encoder f_e is to model the original graph structure into latent space, where MLPs (He et al., 2020) and GNNs (Xia et al., 2023) are usually utilized. Training objectives of different levels can be utilized. By combining a predictor, multiple applications (including node level, edge level and graph level) can be realized.

C Comparison between SSQR and ReaLM

From a methodological perspective, SSQR is optimized through structure reconstruction and semantic distillation leveraging VQ techniques. Furthermore, ReaLM performs feature reconstruction

Table 4: Case study of ReaLM. Taken from (Guo et al., 2025)

Input:
Given the query triplet (Iron Man, has Wife, ?).
We first quantize the head entity Iron Man into a 32-entry integer code:
[7] [48] [109] [586] ... [400] [55].
The answer candidates and corresponding quantized representations are as follows:
JARVIS: [7] [51] [61] [589] ... [200] [205];
Mark VI: [7] [54] [504] [454] ... [447] [187];
Pepper Potts: [7] [67] [213] [587] ... [82] [506].
Please generate quantized representations of the **top-1** potential answer entities, ranked from highest to lowest:

LLM Output:
[6] [3] [61] [589] ... [200] [1]

using RVQ to learn discrete codes. Specifically, SSQR calculates the quantization loss $\mathcal{L}_q$, the structure loss $\mathcal{L}_{st}$, and the $\mathcal{L}_{se}$. Finally, these components are combined into a joint objective for optimization:

$$\mathcal{L}_q = \|\text{sg}[\mathbf{e}^L] - \mathbf{q}_e\|_2^2 + \beta\|\mathbf{e}^L - \text{sg}[\mathbf{q}_e]\|_2^2, \quad (13)$$

$$\mathcal{L}_{st} = -\frac{1}{|\mathcal{F}|}\sum_i [y_i \log \tilde{y}_i + (1-y_i)\log(1-\tilde{y}_i)], \quad (14)$$

$$\mathcal{L}_{se} = \frac{1}{|\mathcal{E}|}\sum_i \|\mathbf{W}_s\mathbf{q}_{e_i} - \mathbf{t}_{e_i}\|_2^2, \quad (15)$$

$$\mathcal{L} = \mathcal{L}_q + \mathcal{L}_{st} + \mathcal{L}_{se}. \quad (16)$$

The structural loss $\mathcal{L}_{st}$ is designed to reconstruct the KG topology (*i.e.*, links) using the learned code representations. Meanwhile, the semantic loss $\mathcal{L}_{sm}$ distills textual information, after embedding via LLMs, into the discrete codes. In contrast, ReaLM first learns semantic KG embeddings using RotatE. The model then requires the quantized embedding, defined as $\mathbf{e}_q = \sum_{s=1}^{S} \mathbf{c}_{s,i_s}$, to effectively reconstruct these semantic embeddings, where $\mathbf{c}_{s,i_s}$ represents the candidate codeword representation. Incorporating the residual quantization loss, the joint optimization objective is formulated as:

$$\mathcal{L}_{\text{rc}} = \underbrace{\|\mathbf{e} - \mathbf{e}_q\|^2}_{\text{reconstruction loss}} + \underbrace{\sum_{s=1}^{S} \|\text{sg}[\mathbf{res}_{s-1}] - \mathbf{c}_{s,i_s}\|^2}_{\text{codebook loss}}, \quad (17)$$

$$\mathcal{L}_{\text{RVQ}} = \mathcal{L}_{\text{rc}} + \beta \underbrace{\sum_{s=1}^{S} \|\mathbf{res}_{s-1} - \text{sg}[\mathbf{c}_{s,i_s}]\|^2}_{\text{commitment loss}}, \quad (18)$$

where $\mathbf{res}_{s-1}$ is the residual vector.

To illustrate the integration process with LLMs, we present case studies for SSQR and ReaLM in Tables 3 and 4, respectively. These examples provide a comprehensive comparison of their distinct instruction formats and the resulting LLM outputs. As highlighted in red within the tables, there are two primary differences: SSQR prompts the model to generate quantized representations for the top-3 potential answer entities, whereas ReaLM only requires the top-1. Consequently, these differing instructions lead to distinct model outputs.

D Quantizing Graphs and Other Modalities

Current studies primarily focus on pure graph structures or text-attributed graphs (Liu et al., 2025; Bo et al., 2025), which may limit their real-world applicability given the increasing importance of integrating multi-source and multi-modal information (Yin et al., 2024). Following this trend, MedTok (Su et al., 2025) jointly encodes textual descriptions from Electronic Health Records (EHRs) and KG-based representations of medical codes to generate richer and structured embeddings for medical foundation models. Looking forward, quantizing graphs alongside other modalities (Zha et al., 2025; Ma et al., 2025), such as images or videos in general domains and electroencephalography (EEG) or electrocardiogram (ECG) in medical domains, could further enhance the cognitive capabilities of these models and extend their utility to a broader range of scenarios.