IVQ-GNN: Mitigating Performance Gap from Graph Connection Pattern Inconsistency via Vector Quantization

Di Jin
School of Computer Science and Technology
Tianjin University
Tianjin, China
jindi@tju.edu.cn

Yixuan Du
School of Computer Science and Technology
Tianjin University
Tianjin, China
yixuantu42@gmail.com

Cuiying Huo*
School of Computer Software
Tianjin University
Tianjin, China
huocuiying@tju.edu.cn

Xiaotong Huang
School of Mathematics
Tianjin University
Tianjin, China
huangxt_3604@tju.edu.cn

Ruqiong Zhang
School of Computer Science and Technology
Tianjin University
Tianjin, China
j1301771092@gmail.com

Xiaobao Wang
School of Artificial Intelligence
Tianjin University
Tianjin, China
wangxiaobao@tju.edu.cn

Yawen Li
School of Economics and Management
Beijing University of Posts and Telecommunications
Beijing, China
warmly0716@126.com

Abstract

Heterophily in graphs is a key challenge for Graph Neural Networks (GNNs). By proposing various homophily measures, recent work has provided insights into how heterophily affects node classification. However, while both graph homophily and heterophily can be further refined into diverse connection patterns, previous work has largely overlooked the role of connection pattern inconsistency. In this paper, we delve deeper into heterophily and homophily by shifting from coarse-grained heterophily ratios to a unified, fine-grained formulation based on connection patterns, and we further reveal an uneven distribution and a train–test gap of these patterns. Empirical studies indicate that this inconsistency leads to severe performance disparity. To address this issue, we propose a novel two-stage method named IVQ-GNN. In the pre-training phase, IVQ-GNN encodes diverse connection patterns into a codebook that serves as an orthogonal basis for the representation space. In the fine-tuning phase, a self-attention module linearly combines these orthogonal bases to expand the learned token space of connection patterns, thereby improving adaptation to rare and out-of-distribution (OOD) patterns. Experimental results on multiple datasets demonstrate that IVQ-GNN significantly improves model performance and validate that the proposed method effectively addresses the connection pattern inconsistency. Our code is available at https://github.com/Duyx5149/IVQ-GNN.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Mathematics of computing** → *Graph algorithms.*

Keywords

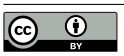
Graph Neural Network, Heterophily, Vector Quantization, Connection Pattern Inconsistency

ACM Reference Format:
Di Jin, Yixuan Du, Cuiying Huo, Xiaotong Huang, Ruqiong Zhang, Xiaobao Wang, and Yawen Li. 2026. IVQ-GNN: Mitigating Performance Gap from Graph Connection Pattern Inconsistency via Vector Quantization. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3774904.3792455

1 Introduction

Graph Neural Networks (GNNs) have emerged as powerful models for processing graph-structured data. By leveraging message-passing mechanisms, GNNs can effectively capture the topological structure of graphs and have achieved remarkable success in a wide range of domains, including social network analysis[11], recommendation systems[14, 28], molecular property prediction[8, 18], and knowledge graphs[25]. However, from the perspective of graph signal processing, classical GNNs can be theoretically interpreted as low-pass filters, which smooth node representations by aggregating

*Cuiying Huo is the corresponding author.



WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792455

features from neighboring nodes. This mechanism implicitly relies on the *homophily assumption* in the graph—that is, connected nodes are likely to belong to the same class. While this assumption holds for many real-world graphs, it fails in heterophilic graphs, where nodes are often connected to others of different classes. In such cases, the performance of traditional GNNs degrades significantly.

To better understand how heterophily affects the performance of GNNs, an increasing body of research has focused on characterizing and quantifying heterophilic structures in graphs[30, 32]. A commonly used metric is the homophily ratio, which measures the proportion of edges connecting nodes of different classes. Based on this metric, many insightful findings have been uncovered regarding how varying levels of homophily influence GNN performance. For example, Luan et al. [16] identified the mid-homophily pitfall—a phenomenon in which graphs with a moderate level of homophily can lead to even worse performance than those with extremely low or high heterophily. In parallel, a number of methods have been proposed to address the challenges posed by heterophily. These existing approaches can be broadly categorized into two types. The first adopts a spectral perspective, designing more flexible filters that go beyond simple low-pass filtering to capture graph signals across a wider range of frequencies (e.g., BernNet[10], ACM-GNN[15], GPR-GNN[6]). The second takes a spatial perspective, focusing on modifying the message-passing mechanism to better accommodate heterophilic structures (e.g., H2GCN[33], Geom-GCN[21]).

However, since links between a node and neighbors from different classes are all categorized as heterophilic edges, it is challenging to explore the impact of these specific connection patterns on node classification tasks solely relying on the homophily ratio. Recently, several works have studied heterophily from a finer-grained perspective, focusing on the various connection patterns. For instance, Luan et al. [16] observed that nodes of the same class may have different neighborhood distributions. Wang et al. [24] pointed out that perturbations in intra-class connection patterns lead to a decrease in node separability.

In this paper, we extend the understanding of heterophily from the perspective of connection patterns. Building upon prior work [19] that suggests an inconsistency in the distribution of homophilic and heterophilic patterns contributes to performance discrepancies, we empirically observe that real-world graphs exhibit substantial *distributional imbalance* and a clear *train–test gap* across different connection patterns. We further verify that these inconsistencies degrade the predictive performance of nodes associated with the affected patterns. To address this issue, we formulate **CSBM-CP**, which theoretically demonstrates that the representation space of a GNN is essentially a low-dimensional image space of the underlying connection-pattern space. Building upon this insight, we propose a two-stage method called **IVQ-GNN** (resolve graph connection pattern **I**nconsistency via **V**ector **Q**uantization). Specifically, leveraging the VQ-VAE [22] framework, IVQ-GNN first explicitly captures the embeddings of the graph's dominating connection patterns in the codebook via Feature and Topology Reconstruction in the pre-training stage. Sequentially, in the fine-tuning stage, IVQ-GNN utilizes and fine-tunes a new cross-attention module that linearly combines primary connection patterns encoded in the codebook, thereby achieving better adaptation of inconsistent patterns. Experimental results across different datasets demonstrate the superiority of IVQ-GNN and validate that it assigns more effective embeddings to minority patterns, thereby improving GNN performance.

We highlight our contribution as follows:

- **Insight**: We reveal that connection patterns in graphs exhibit significant inconsistencies, characterized by an uneven distribution across connection patterns and a noticeable train–test gap. These inconsistencies cause nodes associated with the affected connection patterns to struggle in learning effective representations, ultimately leading to degraded GNN performance.
- **Method**: Building upon this observation, we leverage **CSBM-CP** to theoretically demonstrate that the representation space of a GNN is essentially a low-dimensional image of the underlying connection-pattern space. Guided by this insight, we further propose a two-stage method named **IVQ-GNN**. Through Connection Pattern Codebook Pre-training and Downstream Code-Attention Fine-tuning, IVQ-GNN assigns more effective embeddings for inconsistent patterns with well-learned ones, thereby improving GNN performance.
- **Empirical Validation**: Experimental results demonstrate the superior performance of IVQ-GNN and show that it effectively mitigates the challenges posed by Graph Connection Pattern Inconsistency.

2 Preliminaries

We denote an undirected, connected graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of $N = |\mathcal{V}|$ nodes and $\mathcal{E}$ is the set of edges. The graph is represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, where $A_{u,v} = 1$ indicates an edge between nodes u and v, and $A_{u,v} = 0$ otherwise. The degree matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ is diagonal with entries $D_{u,u} = \sum_v A_{u,v}$. The normalized adjacency matrix is defined as $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Each node $u \in \mathcal{V}$ has a feature vector $\mathbf{x}_u \in \mathbb{R}^d$, and the node feature matrix is $\mathbf{X} \in \mathbb{R}^{N\times d}$. The set of node u's neighbors is denoted by $\mathcal{N}(u) = \{v \in \mathcal{V} \mid A_{u,v} = 1\}$. We use C to denote the number of node classes. The node classification task aims to predict the label vector $\mathbf{Y} \in \mathbb{R}^N$ using both $\mathbf{X}$ and $\tilde{\mathbf{A}}$.

Graph Neural Networks Graph Neural Networks (GNNs), such as GCN[12], GAT[23], and GraphSAGE[9], are powerful frameworks for learning over graph-structured data by aggregating information from local neighborhoods. At each layer, node embeddings are updated based on their own features and the features of neighboring nodes. Formally, the representation $\mathbf{h}'_i$ of node v_i in the next layer is computed as:

$$\mathbf{h}'_i = f_\theta\left(\mathbf{h}_i, \text{AGGREGATE}\left(\{\mathbf{h}_j \mid (v_i, v_j) \in \mathcal{E}\}\right)\right), \tag{1}$$

where $f_\theta(\cdot)$ is a trainable function, and AGGREGATE$(\cdot)$ is a permutation invariant function (e.g., mean, sum, or attention-based) that combines messages from neighboring nodes.

Heterophily and Connection Patterns Previous studies have typically quantified the level of heterophily with homophily ratio. The most representative one is: $\mathcal{H}(\mathcal{G}) = \frac{1}{n}\sum_{i\in[n]} \frac{\sum_{j\in\Gamma_i}(y_i=y_j)}{D_{ii}}$ [21]. However, recent studies have suggested that homophily ratio may fail to accurately reflect the impact of heterophily on node classification performance. For instance, Mao et al. [19] observed that real-world graphs often contain an imbalanced mixture of homophilic and heterophilic nodes, resulting in structural disparities

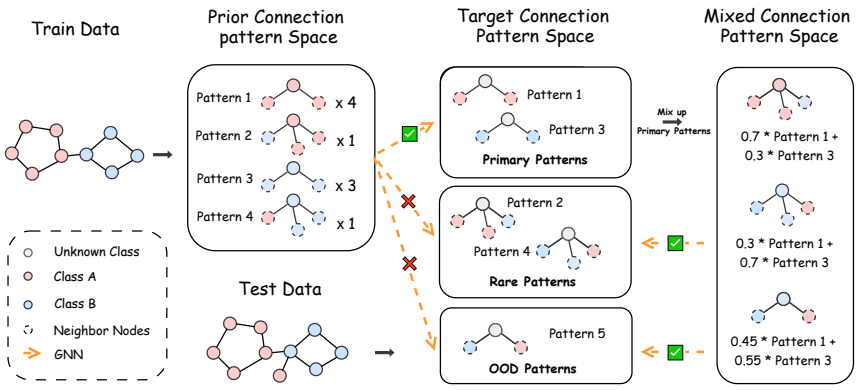


Figure 1: Demonstration for Graph Connection Pattern Inconsistency. Training graphs induce a prior connection-pattern space with imbalanced frequencies. The target space for evaluation comprises *primary* (frequent), *rare* (infrequent), and *OOD* (unseen) patterns, revealing a train–test gap. Many rare/OOD instances can be represented as linear mixtures of primary patterns, yet models fitted to the prior typically succeed on primary patterns and fail on rare/OOD ones.

that lead to performance gaps between majority and minority patterns. Furthermore, Wang et al. [24] pointed out that inconsistency in intra-class connection patterns has a detrimental effect on the separability of node representations.

Inspired by these works, we seek to explore whether connection patterns exhibit distributional imbalance, and how this imbalance impacts GNN performance. To facilitate this study, we begin by formalizing the concept of a node's connection pattern.

Definition 2.1 (Normalized Node Connection Pattern)**.** Given a node labeling function $y : \mathcal{V} \to \{1, 2, \ldots, C\}$ that assigns each node to one of C classes, we define the *normalized connection pattern* of a node $v \in \mathcal{V}$ as a vector $\mathbf{p}_v \in [0, 1]^C$, where the i-th entry is computed as:

$$[\mathbf{p}_v]_i = \frac{|\{u \in \mathcal{N}(v) \mid y(u) = i\}|}{|\mathcal{N}(v)|}. \tag{2}$$

Based on the definition, $\mathbf{p}_v$ represents the class-wise distribution of v's neighbors, normalized to form a probability vector. It is evident that $\mathbf{p}_v$ can be seen as a more general concept compared to homophily ratios. In this work, we focus on exploring the role of this concept in the context of homophilic or heterophilic graph.

Vector Quantization The core idea of Vector Quantization (VQ) is to compress high-dimensional continuous data by training a discrete codebook. VQ-VAE [22] represents a pioneering application of VQ techniques in deep learning, combining a discrete latent space with a continuous encoder. This design significantly improves detail fidelity in image generation tasks. Due to its strong capacity for data compression and discrete representation learning, VQ has been widely adopted in various domains [26, 27]. In the field of graph machine learning, VQGraph [29] adapts the VQ-VAE framework to capture local topological patterns. In the context of this paper, it effectively encodes different connection patterns, thereby enabling efficient knowledge distillation from GNNs to MLPs. Following VQGraph, we propose a variant of VQ-VAE that explicitly models and encodes diverse connection pattern representations, aiming to further address the issue of connection pattern imbalance in graph data.

Specifically, VQ-VAE adopts an auto-encoder architecture. First, the input is encoded into a continuous latent representation via an encoder: $\mathbf{h}_d = E(\mathbf{x})$. This latent vector is then quantized by replacing it with the nearest codebook entry: $\text{quantize}(E(\mathbf{x})) = \mathbf{e}_k$, where $k = \arg\min_j \|E(\mathbf{x}) - \mathbf{e}_j\|$. The quantized vector is fed into the decoder for reconstruction. To train the model, gradients from the reconstruction loss are propagated through the decoder, and the encoder is updated via a straight-through estimator[3]. In addition to reconstruction loss, VQ-VAE introduces a codebook loss to pull code vectors toward encoder outputs, and a commitment loss to encourage encoder outputs to stay close to the selected codes. The overall objective combines these terms to jointly update the encoder, decoder, and codebook:

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = ||\mathbf{x} - D(\mathbf{e})||_2^2 + ||\text{sg}[E(\mathbf{x})] - \mathbf{e}||_2^2 + \eta||\text{sg}[\mathbf{e}] - E(\mathbf{x})||_2^2, \tag{3}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator, and η is a hyperparameter balancing the commitment cost.

3 Connection Pattern Inconsistencies lead to Performance Disparity

As highlighted in [19], the distribution of homophily/heterophily patterns in graphs is often inherently inconsistent. This naturally raises an important question when we further refine homophily/heterophily into diverse connection patterns: *What specific inconsistencies arise among different connection patterns, and how these inconsistencies affect node classification performance.*

To intuitively answer the question, we present a toy example in the schematic diagram shown in Figure 1. As shown in the figure, when we enumerate every distinct connection pattern around a node (center and its neighbors, indicated here by solid and dashed lines), we observe a highly uneven frequency distribution: Pattern 1 occurs 4 times, Pattern 3 occurs 3 times, but Patterns 2 and 4 appear only once each. If we train a standard GNN on this graph, the vast

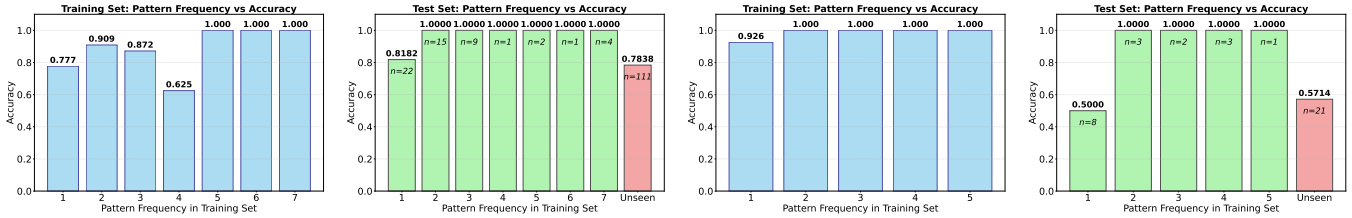


Figure 2: Performance Disparity of Inconsistent Connection Patterns. a–b: Squirrel (one-hot), c–d: Chameleon. Panels (a, c) show the relationship between pattern frequency and accuracy on the training set, while panels (b, d) report the corresponding results on the test set, grouped by their training frequencies and including unseen patterns (denoted as Unseen, with the number of instances n annotated within each bar). The training patterns exhibit a clear long-tailed distribution, where high-frequency patterns achieve higher accuracy. In contrast, test accuracy drops markedly for low-frequency and unseen patterns.

majority of its representation capacity will be devoted to fitting the four or three examples of Pattern 1 and Pattern 3, while the solitary instances of Patterns 2 and 4 are effectively ignored. Moreover, in the test data we may encounter connection patterns that never appear in the training set, such as pattern 5. This observation suggests not only inconsistencies in the distribution of connection patterns within the training set, but also discrepancies between training and testing distributions. Consequently, nodes whose local neighborhoods correspond to such rare or out-of-distribution (ood) patterns are more likely to be misclassified during node classification.

To verify whether the inconsistency of connection patterns leads to performance degradation on certain patterns, we conduct a validation experiment on the **Squirrel** and **Chameleon** datasets using GCN as the base model. Fixing a target class c, we consider only nodes v with $y(v) = c$. For each node, we define its connection pattern as the normalized neighbor-label count vector over all classes following the definition 2.1. We first compute the frequency distribution of these patterns in the training set, then evaluate test accuracy by grouping nodes according to the frequency of their corresponding patterns observed during training, with an additional group for unseen patterns that do not appear in the training data. To eliminate the potential influence of the original feature distribution, we replace node features in **Squirrel** with synthetic one-hot vectors derived directly from ground-truth labels: for each node, we assign a C-dimensional one-hot vector corresponding to its label, thereby isolating the analysis from feature-side variations and focusing purely on structural pattern distribution and its train–test discrepancy. We then repeat the same procedure on **Chameleon** using real features to examine whether the phenomenon persists under realistic conditions.

As shown in Figure 2, both datasets exhibit a pronounced long-tailed distribution of connection patterns, where high-frequency patterns achieve substantially better classification accuracy than low-frequency ones. Moreover, test accuracy is positively correlated with the training frequency, while unseen patterns suffer from a further drop in performance. These observations indicate that the core challenge of homophily and heterophily lies in the imbalanced distribution of connection patterns and the gap between training and testing distributions, which jointly cause generalization degradation on inconsistent connection patterns.

While the above analysis reveals how connection-pattern inconsistency affects performance, what makes this inconsistency particularly noteworthy is the observation that the under-trained Patterns (2, 4, and 5) in Figure 1 are not entirely novel or structurally independent from the primary Patterns 1 and 3. This can be illustrated by Pattern 2, which shares its central node and two neighbor types with Pattern 1, while its remaining neighbor resembles that in Pattern 3. Consequently, inconsistent patterns in the figure can be seen as intermediate forms—essentially linear interpolations between these dominant patterns. This observation naturally motivates the idea of leveraging linear combinations of a few dominant patterns to generalize across diverse inconsistent ones.

Therefore, rather than treating connection patterns as isolated and discrete entities, we propose to view them as points embedded in a latent representation space, where smooth interpolation between patterns becomes possible. If we can learn meaningful embeddings of several base connection patterns that capture their structural and semantic characteristics, it becomes feasible to synthesize plausible representations for under-represented patterns by linearly combining the embeddings of high-frequency patterns. In this way, the performance gap induced by connection-pattern inconsistency can be alleviated, allowing GNNs to better generalize to rare or out-of-distribution yet structurally related patterns. In the next subsection, we further validate the feasibility of this intuition through a mathematical analysis.

4 Mathematical Analysis of Embedding Space of Connection Patterns (CSBM-CP)

To theoretically justify this intuition of learning and linearly combining a set of primary connectivity pattern encodings to generalize to rare or inconsistent patterns, we introduce a variant of the **C**ontextual **S**tochastic **B**lock **M**odel with diverse node-level **C**onnection **P**atterns(**CSBM-CP**).

Definition 4.1 (CSBM-CP $(\mu_1, \mu_2, \ldots, \mu_C, \sigma_1^2 I, \ldots, \sigma_C^2 I, \mathbb{P})$)**.** The generated graph consists of C classes. For a node $v \in C_i$ in class i, its feature vector $\mathbf{x}_v$ is drawn from a multivariate normal distribution $\mathcal{N}(\mu_i, \sigma_i^2 I)$, where $\mu_i \in \mathbb{R}^d$ is the class center and σ_i^2 controls the

intra-class variance. Additionally, each node v samples a normalized connection pattern $\mathbf{p}_v \sim \mathbb{P}$, where $\mathbf{p}_v \in \mathbb{R}^C$ represents the probabilities of node v connecting to nodes in each of the C classes. Specifically, the i-th entry of $\mathbf{p}_v$ represents the probability of node v being connected to nodes in class i. It follows that: $\sum_{i=1}^{C} [\mathbf{p}_v]_i = 1$.

This formulation allows us to formalize how connection patterns determine the aggregated node representations, and to verify that the representation space derived from message passing can indeed be expressed as a linear combination of a few basis patterns. According to the definition of the **CSBM-CP**, the neighborhood structure of the nodes is no longer solely determined by fixed heterophilic/homophilic connection probabilities, but is instead modeled through a connection pattern vector associated with each node. Subsequently, we perform first-order message passing to obtain the aggregated representation of each node: $\mathbf{h}_v = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{x}_u$. It is straightforward to observe that the aggregated node representation still follows a Gaussian distribution, with its expected value given by: $\mathbb{E}[\mathbf{h}_v] = \sum_{i=1}^{C} [\mathbf{p}_v]_i \, \mu_i$. Let the feature matrix before aggregation be defined as: $M = [\mu_1, \mu_2, \ldots, \mu_C] \in \mathbb{R}^{C \times d}$, then the aggregated representation can be further written as:

$$\mathbb{E}[\mathbf{h}_v] = \mathbf{p}_v M. \tag{4}$$

Thus, the space of the aggregated representations is essentially the image space of the connection pattern vector space, and the rank of this representation space satisfies:

$$r = \operatorname{rank}(M) \leq \min(d, C). \tag{5}$$

Assuming the feature dimension $d \gg C$ and that the class centers $\{\mu_i\}$ are linearly independent, we have $\operatorname{rank}(M) \leq C$, indicating that the aggregated representation space forms a low-dimensional image space of at most C dimensions. Therefore, at most C basis vectors corresponding to distinct connection patterns are sufficient to linearly represent all other vectors in the aggregated representation space.

The analysis further provides the theoretical grounding for our proposed model design: if connection patterns span a low-dimensional subspace, it is sufficient to learn a compact set of prototype embeddings (via vector quantization) and generalize to unseen or inconsistent ones through interpolation mechanisms such as code-level attention.

5 IVQ-GNN

The primary motivation behind IVQ-GNN for mitigating Connection Pattern Inconsistency is to assign more reliable representations to inconsistent patterns by leveraging well-represented ones. To achieve this, IVQ-GNN first explicitly encodes the primary connection patterns into a discrete codebook via Vector Quantization (VQ), where the codes are learned through a self-supervised pretraining objective. These quantized codes serve as prototypical embeddings that capture the most representative connection patterns learned from data. Building upon this codebook, IVQ-GNN further introduces a code-level cross-attention module during downstream node classification. This module adaptively generates representations for inconsistent or under-represented patterns by linearly combining the embeddings of well-learned codes according to the learned attention weights. In this way, IVQ-GNN effectively expands the representation space and enhances the model's generalization ability to rare or out-of-distribution connection patterns.

The overall architecture of IVQ-GNN is illustrated in Figure 3. In the following, we elaborate on its two key stages in detail: **Connection Pattern-Aware Codebook Pretraining** and **Downstream Code-Attention Fine-tuning**.

5.1 Connection Pattern-Aware Codebook Pretraining

Inspired by VQGraph, we adopt a variant of VQ-VAE as a tokenizer to encode node connection patterns. This tokenizer consists of a **GNN-based encoder**, a **learnable codebook** of size M: $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_M] \in \mathbb{R}^{M \times d}$ and a **learnable projection layer** $\mathbf{W}$. The encoder, codebook and projection layer are randomly initialized and optimized during pre-training. Concretely, given an input graph with adjacency matrix $\mathbf{A}$ and feature matrix X, the tokenizer first employs a GNN encoder to generate initial node embeddings: $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}$. Each embedding $\mathbf{h}_i$ is then quantized by finding the index of the closest code in the codebook.

It is worth noting that this quantization strategy follows the classical hard indexing paradigm widely used in vector quantization (VQ) methods, which enables explicit retrieval of the dominant connection pattern codes from the codebook. However, such VQ-based paradigms are prone to the well-known codebook collapse problem, where only a small subset of codes are effectively utilized while others remain inactive, leading to low codebook utilization. This phenomenon potentially undermines our goal of using the codebook to explicitly encode the primary connection patterns as orthogonal bases in the representation space. To mitigate this issue, we incorporate the codebook regularization strategy proposed in SimVQ [34], which applies a learnable linear transformation before quantization. This transformation improves code usage during training and prevents the collapse of the learned dictionary. The quantization process can thus be formulated as: $z_i = \arg\min_j \|\mathbf{h}_i - \mathbf{e}'_j\|_2$, where $\mathbf{e}'_j = \mathbf{e}_j \mathbf{W}$. In this way, we obtain the discrete encoding of the input graph structure: $\{\mathbf{e}'_{z_1}, \mathbf{e}'_{z_2}, \cdots, \mathbf{e}'_{z_N}\}$.

To ensure that the codes in the codebook represent the desired connection patterns, we use two linear decoders, along with two associated self-supervised tasks: **node feature reconstruction** and **edge reconstruction**. The losses for these tasks are defined as follows:

$$\mathcal{L}_{Rec} = \underbrace{\frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{\boldsymbol{x}_i{}^T \hat{\boldsymbol{x}}_i}{\|\boldsymbol{x}_i\| \cdot \|\hat{\boldsymbol{x}}_i\|}\right)^{\gamma}}_{\text{node reconstruction}} + \underbrace{\left\| \boldsymbol{A} - \sigma(\hat{\boldsymbol{S}} \cdot \hat{\boldsymbol{S}}^T) \right\|_2^2}_{\text{edge reconstruction}}, \tag{6}$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ denotes node features reconstructed by the Feature Decoder and $\hat{\boldsymbol{S}} \in \mathbb{R}^{N \times d}$ denotes the output of the Structure Decoder. Here, σ represents the sigmoid function. And γ is the scaling factor.

Furthermore, combined with the optimization objective 3 of VQ-VAE, we derive the overall training objective during pre-training:

$$\mathcal{L}_{Pre-train} = \mathcal{L}_{Rec} + \frac{1}{N} \sum_{i=1}^{N} \|\text{sg}[h_i] - \boldsymbol{e}'_{z_i}\|_2^2 + \frac{\eta}{N} \sum_{i=1}^{N} \|\text{sg}[\boldsymbol{e}'_{z_i}] - h_i\|_2^2, \tag{7}$$

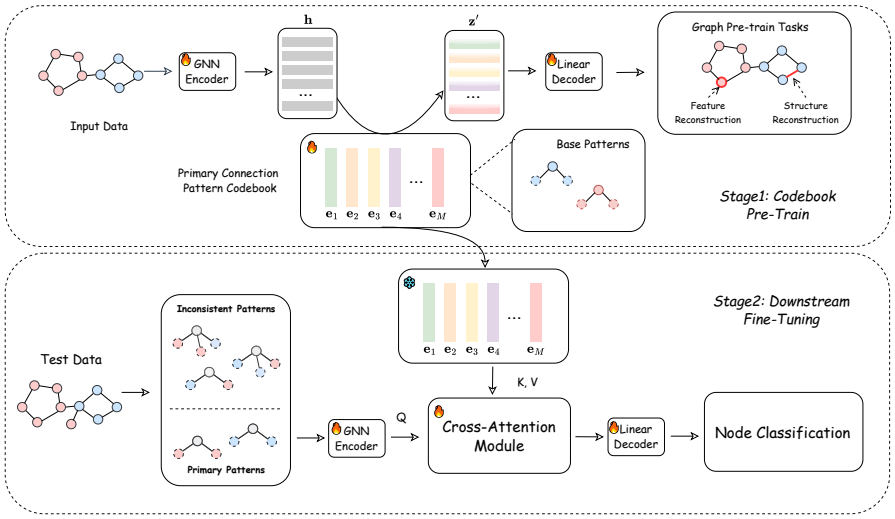


Figure 3: Overview of IVQ-GNN. *Stage 1 (Codebook Pre-Train):* A GNN encoder maps a graph to node embeddings h, which are vector-quantized to tokens z'. A codebook is learned to represent primary connection patterns via reconstruction objectives (feature and structure). *Stage 2 (Downstream Fine-Tuning):* The pretrained codebook is reused; a cross-attention module composes patterns by taking the encoder output as queries (Q) and codebook entries as keys/values (K, V), and a linear decoder performs node classification. Composing codebook entries expands the token space, improving adaptation to rare and OOD connection patterns.

where sg[·] stands for the stop-gradient operator and η is a hyper-parameter set to 0.25 in our experiments.

The proposed pretraining objective naturally drives the model to capture diverse connection patterns as defined in Definition 2.1. Specifically, the node reconstruction term ensures that each quantized code retains sufficient semantic information about node attributes, while the edge reconstruction term enforces that nodes quantized to similar codes can accurately reconstruct their observed connectivity. Together with the VQ-VAE commitment and embedding losses in Eq. 3, the optimization process aligns the encoder and codebook such that each code specializes in representing a distinct structural pattern frequently observed in the graph. Consequently, nodes with similar neighborhood compositions are quantized to similar codes, allowing the learned codebook to discretely partition the connection pattern space and effectively encode various structural motifs present in the data.

6 Downstream Code-Attention Fine-tuning

Motivated by the observations illustrated in Figure 1, we utilize a **codebook attention mechanism** to address the generalization challenge arising from the inconsistent distribution of connection patterns in heterophilic graphs. This mechanism generates diverse and adaptive node representations, thereby enhancing the model's ability to adapt to under-represented and inconsistent patterns.

Building on the pretrained connection pattern codebook, IVQ-GNN performs downstream fine-tuning to further generalize to rare or out-of-distribution connection patterns. In this stage, we reuse the **connection pattern codebook** and the **graph encoder** obtained from the pretraining phase. On top of them, we use a **code-level cross-attention module** and a **classification head** to enable end-to-end learning for node classification. During fine-tuning, the codebook is frozen to preserve the learned connection-pattern prototypes, while the graph encoder, attention module, and classifier are trainable. The parameters of the attention module and classifier are newly initialized before fine-tuning.

Given an input graph, the encoder first produces node representations $\mathbf{h}_i$. Each node embedding is projected into a query vector with residual enhancement:

$$\mathbf{q}_i = \mathbf{h}_i \mathbf{W}_q + \mathbf{h}_i. \tag{8}$$

The frozen codebook entries $\{\mathbf{e}_k\}_{k=1}^{M}$ are mapped into key and value vectors in a similar way:

$$\mathbf{k}_k = \mathbf{e}_k \mathbf{W}_k + \mathbf{e}_k, \quad \mathbf{v}_k = \mathbf{e}_k \mathbf{W}_v + \mathbf{e}_k. \tag{9}$$

Further, SoftVQ [7] is adapted, and all codebook entries serve as global key–value tokens, allowing each node to perform cross-attention over the learned connection-pattern prototypes:

$$\boldsymbol{\alpha}_i = \mathrm{Softmax}\left(\frac{\mathbf{q}_i \mathbf{K}^\top}{\sqrt{d}}\right), \quad \mathbf{o}_i = \sum_{k=1}^{M} \alpha_{ik} \mathbf{v}_k. \tag{10}$$

The outputs are refined through a feed-forward layer to obtain enhanced representations:

$$\tilde{\mathbf{h}}_i = \mathrm{FFN}(\mathbf{o}_i). \tag{11}$$

The enhanced representations $\tilde{\mathbf{h}}_i$ are then passed to a newly initialized decoder $\mathbf{W}_{\mathrm{clf}}$, which is trained from scratch for the downstream classification task:

$$\hat{\mathbf{y}}_i = \mathrm{Softmax}(\tilde{\mathbf{h}}_i \mathbf{W}_{\mathrm{clf}}), \tag{12}$$

Table 1: Performance comparison of different models across heterophily/homophily datasets. The best results are highlighted in bold, and the second-best results are underlined.

Datasets	Cora	Citeseer	Pubmed	Chameleon	Squirrel	Roman-Empire	Minesweeper
GCN	88.23±1.75	75.01±0.91	87.30±0.45	43.38±4.71	38.74±1.29	46.23±1.23	79.80±0.25
SAGE	87.97±1.20	75.40±0.92	88.51±0.21	43.60±3.79	37.48±1.11	72.41±0.64	85.11±0.69
GAT	88.93±1.23	75.88±0.94	87.22±0.27	43.26±1.95	37.26±2.78	64.78±0.92	79.66±0.28
GCNII	87.90±1.35	75.50±1.28	86.32±0.23	38.88±3.30	35.60±2.25	57.52±0.42	80.78±0.59
APPNP	86.90±1.27	76.46±1.56	84.81±0.59	38.31±3.60	35.01±2.07	49.97±1.08	79.81±0.28
Half-Hop	87.08±0.94	75.86±1.36	88.62±0.26	38.88±1.68	33.18±0.65	66.76±0.63	79.72±0.30
ACM-GNN	88.16±1.10	76.32±1.17	**89.74±0.34**	38.35±1.66	35.38±1.71	67.88±0.30	84.43±0.75
GPR-GNN	88.93±1.28	76.23±0.62	87.58±0.27	39.66±3.00	36.26±2.01	63.62±0.74	80.47±0.55
FAGCN	89.00±1.14	76.24±1.07	89.25±0.33	40.00±2.08	37.71±1.59	65.76±0.55	79.80±0.64
FSGNN	87.20±0.76	75.29±1.17	89.41±0.37	43.09±3.06	39.01±1.45	71.08±2.39	82.94±0.67
BernNet	88.30±1.55	76.33±0.68	87.13±0.70	39.46±0.24	38.16±2.18	64.43±0.76	79.93±0.23
IVQ-GNN(Ours)	**89.23±1.18**	**76.49±1.16**	87.22±0.34	**43.71±2.47**	**39.19±2.75**	**72.99±0.79**	**85.74±0.54**

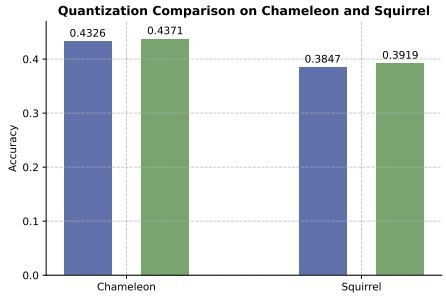


Figure 4: Quantization comparison on Chameleon and Squirrel datasets. Green bars represent our proposed IVQ-GNN, while blue bars denote its hard-indexing variant.

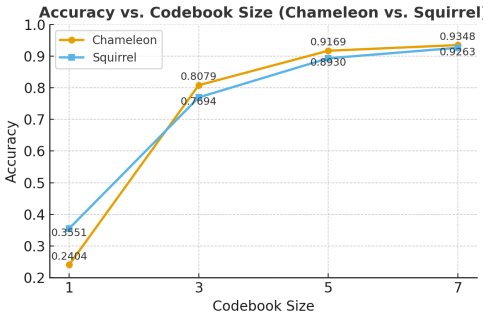


Figure 5: Accuracy vs. codebook size on Chameleon and Squirrel datasets.

and the model is optimized with a cross-entropy loss over the labeled nodes.

Through this design, each node dynamically aggregates structural knowledge from the pretrained codebook. Rather than treating connection patterns as static or isolated, the attention mechanism linearly combines multiple well-learned prototype embeddings to synthesize new representations for inconsistent or under-represented patterns. Consequently, IVQ-GNN effectively expands the representational capacity of GNNs and enhances generalization to long-tail and out-of-distribution connection patterns.

7 Experiments

In this section, we present comprehensive experiments to evaluate the effectiveness of IVQ-GNN. Specifically, we aim to answer the following research questions: **RQ1:** Can IVQ-GNN consistently outperform existing baselines on both heterophilic and homophilic graph benchmarks? **RQ2:** Does the Code-Attention mechanism provide more effective representations for under-represented or inconsistent connection patterns? **RQ3:** How does the number of codes in the codebook affect the performance of IVQ-GNN?

7.1 Evaluation on Heterophilic/Homophilic Graphs

In this experiment, we compare IVQ-GNN with various baseline methods on both heterophilic and homophilic graph datasets and answer **RQ1**.

Dataset and Evaluation We evaluate our proposed model on a diverse collection of benchmark datasets, including both homophilic and heterophilic graphs. For the heterophilic setting, we adopt four recently proposed datasets [17]: Squirrel, Chameleon, Roman-Empire, Minesweeper. For the homophilic setting, we use standard citation graphs [17] including Cora, CiteSeer, PubMed. We follow standard node classification settings and report node-level classification accuracy. For each dataset, we report the mean and standard deviation of accuracy over 5 random splits. Detailed dataset statistics and split strategies are provided in Appendix A.1.

Setup For a fair comparison, we benchmark IVQ-GNN against classical GNNs (GCN [12], GraphSAGE [9], GAT [23]) and strong

graph filters / heterophily-oriented baselines: Half-Hop [2], ACM-GNN [15], GPR-GNN [6], FAGCN [4], FSGNN [20], BernNet [10], APPNP [13], and GCNII [5]. All methods follow the same train/validation/test splits per dataset; results are averaged over 5 random splits and reported as mean ± standard deviation. For our method, we adopt a 2-layer GAT as the GNN encoder, which is shared across all datasets. All models are trained until convergence using early stopping on the validation set. The results on homophilic and heterophilic datasets are reported in Table 1. Detailed hyperparameter settings for both our method and the baselines can be found in Appendix A.2.

Results From the results from Table 1, we observe that the proposed IVQ-GNN consistently outperforms other baseline methods across most datasets. Notably, it achieves significant accuracy improvements of 0.58% on **Roman-Empire** and 0.63% on **Minesweeper**, which highlights its effectiveness. In the following experiment, we further investigate whether this framework can provide better representations for infrequent connection patterns.

7.2 Ablation Study

To answer **RQ2**—whether the proposed codebook cross-attention improves adaptation to under-represented connection patterns—we compare IVQ-GNN with an ablated variant that replaces cross-attention in fine-tuning with hard indexing, i.e., the same nearest-code assignment used during pretraining.

Variant (w/o Code-Attention) After the shared pretraining stage, we *freeze* the learned codebook and encoder. During fine-tuning, each node embedding selects its *single nearest code* via argmin Euclidean distance (hard index); the selected code embedding is fed directly to the classifier (no attention mixing across codes). All other components, including the classifier head and optimization protocol, match IVQ-GNN.

Datasets and Metric We use two representative benchmarks in Experiment 7.1: Chameleon, Squirrel. We report node-level accuracy (%). Both models (IVQ-GNN and the hard-index variant) use the same backbone, identical pretraining, identical codebook size, and identical train/validation/test splits. We average results over 5 random splits and report mean ± standard deviation. Training uses early stopping on the validation set. Hyperparameters are selected on the validation split within the same ranges as Experiment 7.1. The result is reported in Figure 4

Result Based on the result in Figure 4, on *Chameleon*, IVQ-GNN attains **43.71%** accuracy vs. **43.26%** for hard indexing (**+0.45** points). While on *Squirrel*, IVQ-GNN reaches **39.19%** vs. **38.47%** (**+0.72** points). Under identical pretraining, backbones, and codebook size, these consistent gains indicate that the cross-attention module over codebook entries yields more expressive node representations at fine-tuning time. This supports our hypothesis that linearly mixing learned connection pattern prototypes helps the model adapt to *under-represented* connection patterns, aligning with our analysis.

7.3 Analysis of Codebook Size

To investigate **RQ3** and validate the theoretical analysis on the intrinsic dimensionality of the GNN embedding space (Section 4), we study how the *codebook size* affects classification performance of IVQ-GNN.

Datasets and Feature Treatment We use Chameleon and Squirrel. To avoid confounding effects from the original feature distributions and isolate the role of *connection-pattern*, we replace all node features on both datasets with *one-hot vectors derived from ground-truth labels*: for each node, a C-dimensional one-hot vector is assigned with the y-th entry set to 1.

Setup We keep the architecture, optimizer, and training protocol fixed, and vary the *codebook size* $M \in \{1, 3, 5, 7\}$. The full IVQ-GNN (with cross-attention in fine-tuning) is used in all cases; only the number of codes differs. Pretraining and fine-tuning follow the identical objectives as in Experiment 7.1. We use the same train/validation/test splits as Experiment 7.1, and report node-level accuracy (%). Results are averaged over 5 random splits and reported. Early stopping is applied on the validation set. Hyperparameters (excluding M) are selected on the validation split using the same search ranges as Experiment 7.1. The result is plotted in Figure 5.

Result From the result in Figure 5, we observe a clear *increasing-then-saturating* trend on both datasets: from M=1 to M=3, accuracy jumps sharply (Chameleon: **24.04%** → **80.79%**; Squirrel: **35.51%** → **76.94%**). Further increasing to M=5 yields substantial but smaller gains (Chameleon: **91.69%**; Squirrel: **89.30%**), while moving from M=5 to M=7 provides only marginal improvements (Chameleon: **93.48%**; Squirrel: **92.63%**).

Notably, both datasets have C=5 **classes**. Our analysis (Section 4) reveals that the aggregated GNN embedding space is an image of the connection-pattern space with rank at most C, i.e., requiring at most C **orthogonal bases**. Empirically, M=5 is exactly the *knee point*: accuracy growth slows markedly beyond this size. This alignment between the observed elbow at M=5 and the theoretical upper bound on the basis size provides direct evidence that enlarging the codebook improves performance up to the intrinsic dimensionality of the embedding space, after which returns diminish.

8 Conclusion

In this work, we investigated the inconsistencies of connection patterns in both homophilic and heterophilic graphs, characterized by their uneven distribution and train–test gap, and revealed their negative impact on node classification. To address this issue, we theoretically justified that the message-passing embedding space is a low-dimensional image of the underlying connection-pattern space with rank at most the number of classes, implying that a compact set of basis patterns is sufficient to represent and generalize to inconsistent ones. Guided by this insight, we proposed IVQ-GNN, a novel framework that enhances the representations of inconsistent connection patterns by connection pattern codebook pre-training and code-level attention fine-tuning. Through various experiments, we demonstrate the effectiveness of IVQ-GNN in improving node classification performance, particularly for inconsistent connection patterns. These findings offer insightful implications for future research in graph quantization and graph heterophily.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 92370111, No. 62422210, No. 62272340 and No. 62276187), the Postdoctoral Fellowship Program of CPSF under Grant No. GZC20251059, and Hebei Natural Science Foundation (No. F2024202047).

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International conference on machine learning (ICML)*. 21–29.

[2] Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Veličković, and Eva L. Dyer. 2023. Half-Hop: A graph upsampling approach for slowing down message passing. arXiv:2308.09198 [cs.LG] https://arxiv.org/abs/2308.09198

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432 [cs.LG] https://arxiv.org/abs/1308.3432

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. arXiv:2101.00797 [cs.LG] https://arxiv.org/abs/2101.00797

[5] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. arXiv:2007.02133 [cs.LG] https://arxiv.org/abs/2007.02133

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[7] Zhangyang Gao, Cheng Tan, Jue Wang, Yufei Huang, Lirong Wu, and Stan Z. Li. 2024. FoldToken: Learning Protein Language via Vector Quantization and Beyond. arXiv:2403.09673 [q-bio.BM] https://arxiv.org/abs/2403.09673

[8] Jonathan Godwin, Michael Schaarschmidt, Alexander Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. 2021. Simple gnn regularisation for 3d molecular property prediction & beyond. *arXiv preprint arXiv:2106.07971* (2021).

[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of NeurIPS*. 1024–1034.

[10] Mingguo He, Zhewei Wei, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems* 34 (2021), 14239–14251.

[11] Lokesh Jain, Rahul Katarya, and Shelly Sachdeva. 2023. Opinion leaders for information diffusion using graph neural network in online social networks. *ACM Transactions on the Web* 17, 2 (2023), 1–37.

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*.

[13] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proceedings of ICLR*.

[14] Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S. Yu. 2020. BasConv: Aggregating Heterogeneous Interactions for Basket Recommendation with Graph Convolutional Neural Network. In *Proceedings of SDM*. 64–72.

[15] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification? arXiv:2109.05641 [cs.LG] https://arxiv.org/abs/2109.05641

[16] Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. 2024. When Do Graph Neural Networks Help with Node Classification? Investigating the Impact of Homophily Principle on Node Distinguishability. arXiv:2304.14274 [cs.SI] https://arxiv.org/abs/2304.14274

[17] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. arXiv:2406.08993 [cs.LG] https://arxiv.org/abs/2406.08993

[18] Hehuan Ma, Yatao Bian, Yu Rong, Wenbing Huang, Tingyang Xu, Weiyang Xie, Geyan Ye, and Junzhou Huang. 2020. Multi-view graph neural networks for molecular property prediction. *arXiv preprint arXiv:2005.13607* (2020).

[19] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. 2023. Demystifying Structural Disparity in Graph Neural Networks: Can One Size Fit All?. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=oef30oScVB

[20] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2021. Improving Graph Neural Networks with Simple Architecture Design. *CoRR* abs/2105.07634 (2021).

[21] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. arXiv:2002.05287 [cs.LG] https://arxiv.org/abs/2002.05287

[22] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2018. Neural Discrete Representation Learning. arXiv:1711.00937 [cs.LG] https://arxiv.org/abs/1711.00937

[23] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of ICLR*.

[24] Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. 2024. Understanding Heterophily for Graph Neural Networks. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 50489–50529.

[25] Zhe Wang, Suxue Ma, Kewen Wang, and Zhiqiang Zhuang. 2025. Rule-Guided Graph Neural Networks for Explainable Knowledge Graph Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 12784–12791.

[26] Lirong Wu, Yijun Tian, Haitao Lin, Yufei Huang, Siyuan Li, Nitesh V Chawla, and Stan Z Li. 2024. Learning to Predict Mutation Effects of Protein-Protein Interactions by Microenvironment-aware Hierarchical Prompt Learning. *arXiv preprint arXiv:2405.10348* (2024).

[27] 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 *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=jevY-DtiZTR

[28] Fengli Xu, Jianxun Lian, Zhenyu Han, Yong Li, Yujian Xu, and Xing Xie. 2019. Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation. In *Proceedings of CIKM*. 529–538.

[29] Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, Bin Cui, Muhan Zhang, and Jure Leskovec. 2024. VQGraph: Rethinking Graph Representation Space for Bridging GNNs and MLPs. arXiv:2308.02117 [cs.LG] https://arxiv.org/abs/2308.02117

[30] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S. Yu. 2022. Graph Neural Networks for Graphs with Heterophily: A Survey. *CoRR* abs/2202.07082 (2022).

[31] Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed, and Danai Koutra. 2021. Graph Neural Networks with Heterophily. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. 11168–11176.

[32] Jiong Zhu, Yujun Yan, Mark Heimann, Lingxiao Zhao, Leman Akoglu, and Danai Koutra. 2023. Heterophily and graph neural networks: Past, present and future. *IEEE Data Engineering Bulletin* (2023).

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

[34] Yongxin Zhu, Bocheng Li, Yifei Xin, and Linli Xu. 2024. Addressing Representation Collapse in Vector Quantized Models with One Linear Layer. arXiv:2411.02038 [cs.LG] https://arxiv.org/abs/2411.02038

A Experimental Setup

A.1 Dataset Statistics and Splitting

We evaluate our model on a variety of widely-used benchmark datasets, including both homophilic and heterophilic graphs. Table 2 summarizes the key statistics. For each dataset, we use 5 random splits with fixed train/validation/test ratios (60%/20%/20%) following standard setting used in [17]. Apart from the dataset splitting, all other dataset-related operations are based on the official DGL implementation.

Table 2: Dataset statistics.

Dataset	#Nodes	#Edges	#Classes	#Features
Cora	2,708	10,556	7	1,433
Citeseer	3,327	9,228	6	3,703
Pubmed	19,717	88,651	3	500
Chameleon	2,277	36,101	5	2,325
Squirrel	5,201	217,073	5	2,089
Roman-Empire	22,662	65,854	18	300
Minesweeper	10,000	78,804	2	7

A.2 Model Architecture and Hyperparameters

The encoder of our method is a 2-layer GAT with 8 attention heads per layer and 64 hidden dimensions per head. The model integrates a vector quantization module with one codebook of embedding dimension 64.

Other key configurations:

- Codebook size: [4, 8, 16]
- Commitment cost: 0.25
- Dropout: [0.1, 0.3, 0.5, 0.7]

For all baseline methods, we ran experiments using the default parameter settings provided in the official code release.

A.3 Training Procedure

The training consists of two sequential phases:

Pre-training Phase. In the pretraining stage, we optimize the quantized representation by minimizing a joint reconstruction loss that includes:

- Vector quantization loss
- Feature reconstruction loss
- Edge reconstruction loss

The optimizer is Adam with a learning rate of 0.001. Models are trained for 1000 epochs with early stopping disabled in pretraining.

Classification Phase. After pretraining, the model switches to node classification mode. Only classification-related parameters are updated using cross-entropy loss on labeled nodes. The optimizer is Adam with:

- Learning rate: [1e-3, 5e-4, 1e-4]
- Weight decay: [0, 5e-4]

Validation accuracy is monitored, and the best model is selected according to the highest validation performance.

Evaluation. Final performance is measured on the test set using the best validation checkpoint. The test accuracy is averaged over 5 random splits, and we report mean and standard deviation.

A.4 Implementation and Hardware Details

All experiments are implemented with Deep Graph Library (DGL) and Pytorch Geometric (PyG), and run on a server with the following configuration:

- **CPU**: Intel Xeon Platinum 8358 @ 2.60GHz
- **GPU**: Tesla V100 SXM2 32GB
- **Environment**: Ubuntu 20.04, Python 3.9

We will release our source code and data splits upon publication to ensure full reproducibility.

B Related Work

To address the problem of heterophily, numerous works have emerged recently. Mixhop [1] proposes to mix representations from multi-hop neighborhoods to get more information. Geom-GCN [21] identifies and aggregates nearby and distant nodes with similarity to the target node in a constructed hidden space to address heterophily; H2GCN [33] proposes several designs including ego and neighbor embedding separation, higher-order neighborhood aggregation, and combination of intermediate representations to enhance the performance of GNNs on heterophilic graphs; GPR-GNN [6] introduces a novel Generalized PageRank (GPR) GNN architecture, assigning each feature propagation step a learnable weight to address heterophily and oversmoothing issues; CPGNN [31] introduces an interpretable compatibility matrix to model heterophily or homophily levels in the graph, enabling the generalization of GNNs for graphs with either homophily or heterophily; FAGCN [4] proposes a self-gating mechanism to adaptively integrate low-frequency signals, high-frequency signals, and raw features, enhancing adaptability without requiring knowledge of network types; ACM-GNN [15] improves GNN performance by allowing adaptive exploitation of aggregation, diversification, and identity channels based on a post-aggregation node similarity matrix.