TOKENIZE, DIFFUSE, DECODE: A GENERATIVE APPROACH TO NEIGHBORHOOD DISCOVERY ON GRAPHS

Zhuowen Yuan[1,2*], **Tao Liu**[2], **Kaushik Rangadurai**[2], **Yang Yang**[1], **Minhui Huang**[1], **Yiping Han**[1], **Bo Li**[2], **Shuang Yang**[1]

[1]Meta AI [2]University of Illinois at Urbana-Champaign

ABSTRACT

Node neighbor discovery is a central component of graph representation learning, yet most existing methods rely on heuristic sampling or deterministic neighborhood expansion that limits adaptivity and robustness. SemWalk introduces a generative approach to neighbor discovery that models a conditional distribution over informative neighbor sequences given a source node, using discrete diffusion over semantically tokenized graph representations. The method first learns a discrete semantic space by training a Residual Quantized Variational Autoencoder (RQ-VAE) to tokenize continuous node embeddings, and then trains an order-agnostic autoregressive diffusion model (OA-ARDM) in this space to generate permutation-invariant neighbor sequences. At inference time, discrete neighbors are sampled conditioned on the source node's semantic token and decoded back into continuous embeddings via the RQ-VAE decoder, enabling diverse and high-quality neighborhood generation. Empirical results on large-scale multi-graph benchmarks show that SemWalk consistently matches or outperforms established baselines such as Personalized PageRank (PPR), with particularly strong robustness under test-time noise and graph heterogeneity, while remaining fully inductive and generalizing to unseen graphs.

1 INTRODUCTION

Graphs are ubiquitous in real-world systems, representing relational structures in domains such as social networks, recommendation systems, citation networks, and biological interaction graphs. A central problem in graph learning is how to effectively represent a node by leveraging its structural and semantic context. Traditional methods, including Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2017), rely on message passing mechanisms to aggregate information from neighboring nodes. While powerful, GNNs suffer from key limitations: their expressiveness is constrained by local aggregation; they are sensitive to over-smoothing in deep architectures (Chen et al., 2020); and they struggle to generalize inductively to unseen graphs or dynamic node distributions.

An alternative paradigm gaining traction is to model graphs through discrete token sequences and apply Transformer-based architectures (Chen et al., 2022; 2025). Tokenized Graph Transformers (GTs) convert the local neighborhood of each node into a sequence of tokens, enabling the use of attention mechanisms to capture dependencies and global structure. However, a core bottleneck in tokenized GTs is the construction of these sequences. Existing approaches typically rely on deterministic heuristics such as top-k similarity or fixed-hop neighborhood sampling (Chen et al., 2025). These methods are limited in both diversity and adaptivity—they fail to model the stochasticity and semantic richness of real-world neighborhoods, and cannot adapt to new or evolving graphs without retraining or redesign. Moreover, the deterministic nature of these strategies makes them brittle under structural perturbations.

However, such perturbations are prevalent in many real-world applications. Graph structures often evolve dynamically or are subject to corruption due to noise, missing links, or adversarial perturbations (Bojchevski & Günnemann, 2019; Chi et al., 2024). For example, social and recommendation

[*†]Work done during an internship at Meta.

graphs may have missing or spurious connections caused by incomplete user interactions, while biomedical interaction networks may contain erroneous links from experimental uncertainty. These imperfections challenge models that rely on precise structural connectivity and highlight the need for approaches that are resilient to topological noise at test time.

To address these challenges, we propose SemWalk, a generative framework that models the distribution over semantic neighbor sequences using discrete diffusion models. The key idea behind SemWalk is to treat neighborhood generation as a semantic sequence modeling task. Instead of selecting neighbors heuristically, SemWalk learns to generate sequences of discrete semantic tokens that capture the contextual semantics of a node in the graph.

The framework is structured into three fundamental stages. First, a Residual Quantized Variational Autoencoder (RQ-VAE) (Lee et al., 2022) is trained to transform continuous node embeddings into compact semantic token IDs. This allows nodes to be represented in a discrete latent space suitable for sequence modeling. Second, an order-agnostic autoregressive diffusion model (OA-ARDM) (Hoogeboom et al., 2021) is trained on tokenized neighbor sequences. Given a source token, the model learns to generate semantically coherent neighbor tokens by modeling the distribution over permutations and prediction orders. Third, at inference time, SemWalk generates neighbor token sequences from a single source token, and reconstructs their embeddings via the RQ-VAE decoder.

SemWalk offers several notable advantages. First, it is *inductive by design*—the learned tokenizer and diffusion model can generalize to unseen nodes and graphs without retraining. Second, by modeling distributions over neighborhoods instead of relying on fixed rules, SemWalk produces richer and more diverse semantic contexts. Third, and most importantly, SemWalk demonstrates strong robustness to topological noise at test time. Since it generates neighbors in a semantic space learned from clean data, the model is less sensitive to missing or corrupted edges in the observed graph. This is particularly beneficial in domains where graph connectivity is noisy or dynamic. Empirical evaluations show that SemWalk generates high-quality node representations, is resilient to structure corruption, and outperforms baselines such as Personalized PageRank (PPR) (Page et al., 1999), VCR-Graphormer (Fu et al., 2024), and SwapGT in downstream tasks.

In summary, we make the following contributions in this work:

- We propose SemWalk, a generative framework that produces semantic neighbor sequences using a discrete diffusion model trained in a tokenized latent space.
- We design a novel training paradigm that integrates residual quantization and order-agnostic diffusion modeling, enabling diversity in token generation and generalization to unseen graphs.
- We demonstrate that SemWalk yields high-quality node representations and exhibits strong robustness to test-time structural noise, outperforming strong baselines across multiple datasets.

2 Related Work

2.1 Graph Diffusion Models

Recent work in graph representation learning has explored the use of diffusion-based propagation as a method to enhance the expressiveness of graph neural networks. These approaches treat diffusion as a mechanism to aggregate information from increasingly distant nodes, often addressing key limitations such as oversmoothing and limited receptive fields. For example, GREAD (Choi et al., 2023) introduces a reaction–diffusion framework that mitigates oversmoothing by incorporating learnable reaction terms. Dan et al. (Dan et al., 2023) propose a variational functional framework, combining total variation regularization with adaptive propagation, achieving deeper GNNs with edge-aware diffusion. AGDN (Sun et al., 2020) proposes a learnable hop-specific attention mechanism over graph diffusion, enabling the model to adaptively weigh different diffusion steps (i.e., neighborhood hops). This helps overcome limitations of fixed-hop aggregation in GNNs, allowing deeper and more expressive architectures that adapt to different graph structures. ARROW-Diff (Bernecker et al., 2024) proposes a random walk-based discrete diffusion framework tailored for large-scale graph generation. Their model employs a token masking process and reverse denoising to iteratively reconstruct full graphs from sampled walks, optimizing both efficiency and generation quality. While conceptually related, ARROW-Diff focuses on unconditional graph gen-

eration, whereas our method tackles conditional neighborhood generation with semantic grounding via residual quantization. Our work also emphasizes robustness and inductive generalization at the node level, distinguishing it from full-graph diffusion approaches.

2.2 NEIGHBOR DISCOVERY AND SAMPLING

Neighbor discovery plays a vital role in building scalable and expressive GNNs. Personalized PageRank (PPR)(Page et al., 1999) serves as a foundational method for ranking node relevance via diffusion. PPRGo(Bojchevski et al., 2020) leverages an efficient approximation of PPR to precompute sparse neighbor sets, enabling fast and accurate large-scale inference. Other early approaches such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) employ random walks to sample co-occurring nodes, treating the resulting sequences as context windows for representation learning. More recently, PathNN (Michel et al., 2023) aggregates over shortest-path neighborhoods to improve expressiveness. VCR-Graphormer (Fu et al., 2024) introduced PPR-based fixed-length tokenization along with structure-aware and content-aware "virtual tokens" that encode global or heterophilous context. These tokens ensure semantic consistency and improve generalization across varying graph topologies. Similarly, SwapGT (Chen et al., 2025) augments token sequences via feature-based and topology-based neighbor lists and proposes a token-swapping mechanism to inject diversity. A center alignment loss enforces consistency across representations derived from different neighborhood views. Both methods emphasize semantic richness and robustness in neighborhood modeling—principles aligned with SemWalk. Our framework goes further by learning a diffusion-based generative model over semantic tokens, enabling both diversity and inductive consistency without relying on deterministic neighbor heuristics.

2.3 DISCRETE TOKENIZATION

Discrete representation learning bridges graph data and generative modeling by converting continuous inputs into symbolic tokens. VQ-VAE (Van Den Oord et al., 2017) introduced a learned codebook-based quantization mechanism, and RQ-VAE (Lee et al., 2022) improved this via residual encoding. Recent graph-specific tokenization methods have leveraged these advances: MoleBERT (Xia et al., 2023) learns atom-level tokens via context-aware VQ-VAE, while SimSGT (Liu et al., 2023) and VQGraph (Yang et al., 2023) discretize subgraphs or node neighborhoods to produce compact, structure-aware representations for pretraining and distillation. These works demonstrate that discrete tokens enhance both performance and interpretability. On the generative side, discrete denoising diffusion models have emerged as powerful tools. DiGress (Vignac et al., 2022) adapts DDPMs to graph generation via discrete corruption steps, while DISCO (Xu et al., 2024) introduces continuous-time Markov diffusion over graph structures. OA-ARDM (Hoogeboom et al., 2021) enables flexible token-level generation via random order autoregressive denoising. Our method unifies these directions by using RQ-VAE to tokenize graph embeddings and OA-ARDM to learn a discrete, order-agnostic diffusion process over neighborhoods, enabling diverse, inductive, and robust neighbor generation.

3 METHOD

3.1 OVERVIEW

In this work, we propose a three-stage pipeline for generating node neighbors via discrete diffusion models. The overview of our pipeline is illustrated in Figure 1. The pipeline consists of: (1) training a Residual Quantized Variational Autoencoder (RQ-VAE) to tokenize node embeddings into a compact, discrete semantic space, (2) training an order-agnostic autoregressive diffusion model on sequences constructed from the semantic tokens, and (3) inferring neighbors by generating new token sequences, which are subsequently decoded back to continuous node embeddings. In the following, we elaborate on each component of the system in detail.

3.2 SEMANTIC TOKENIZER

In the first stage, we build a semantic tokenizer using a Residual Quantized Variational Autoencoder (RQ-VAE) (Lee et al., 2022). For each node v in the graph, a continuous embedding $\mathbf{h}_v \in \mathbb{R}^d$

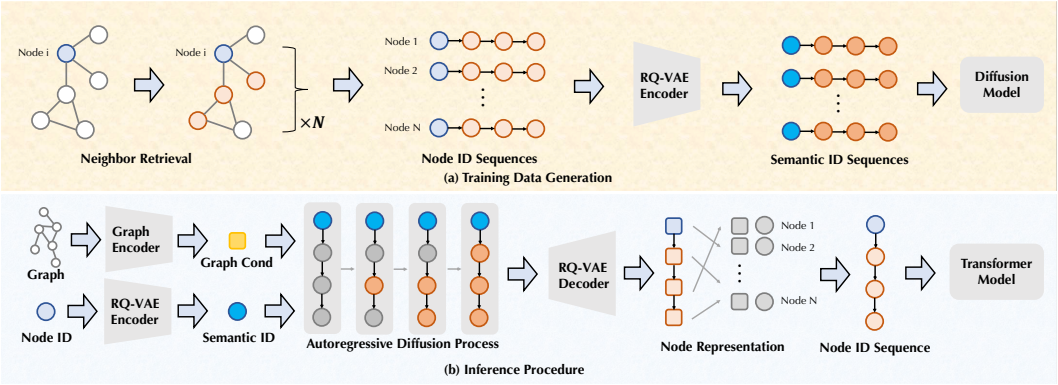


Figure 1: Overview of our method. **(a) Training:** PPR is used to sample neighbor sequences, which are encoded into discrete semantic tokens by a pre-trained RQ-VAE to train a discrete diffusion model. **(b) Inference:** Conditioned on a source node token and a graph-level embedding, the diffusion model generates a token sequence that is decoded, matched to real nodes, and fed to a Transformer for node-level prediction.

is obtained directly from the dataset and normalized using the training set statistics via standard score normalization (i.e., subtracting the mean and dividing by the standard deviation). The RQ-VAE encoder maps $\mathbf{h}_v$ to a sequence of discrete code indices $[a_1, a_2, \ldots, a_n]$, where each $a_i \in \{0, \ldots, \text{card} - 1\}$ corresponds to the index of the closest vector in a learned codebook at level i.

Residual quantization is performed in a multi-stage fashion. At level i, the encoder outputs a residual vector r_i, which is quantized to the nearest codebook vector $e_{a_i}^{(i)}$, and the next level quantizes the residual of the previous quantization:

$$r_1 = \mathbf{h}_v, \quad r_{i+1} = r_i - e_{a_i}^{(i)}, \quad \text{for } i = 1, \ldots, n.$$

The final quantized embedding is the sum of quantizations from all levels:

$$z_q = \sum_{i=1}^{n} e_{a_i}^{(i)}.$$

The decoder reconstructs $\hat{\mathbf{h}}_v$ from z_q, and the model is trained to minimize the reconstruction loss

$$\mathcal{L}_{\text{rec}} = \left\| \mathbf{h}_v - \hat{\mathbf{h}}_v \right\|_2^2.$$

To stabilize training and encourage effective codebook learning, RQ-VAE minimizes a reconstruction loss along with a quantization loss consisting of commitment and codebook terms:

$$\mathcal{L}_{\text{RQ-VAE}} = \mathcal{L}_{\text{rec}} + \underbrace{\sum_{i=1}^{n} \|\text{sg}[r_i] - e_{a_i}\|_2^2}_{\text{codebook}} + \underbrace{\beta \sum_{i=1}^{n} \|r_i - \text{sg}[e_{a_i}]\|_2^2}_{\text{commitment}},$$

where r_i is the residual input to quantization level i, e_{a_i} is the selected codeword, $\text{sg}[\cdot]$ denotes the stop-gradient operator, and β is a hyperparameter controlling the strength of the commitment loss.

To encourage equal utilization of all quantization levels and reduce codebook collapse, we apply the *rotation trick* (Fifty et al., 2024), where the order of residual levels is cyclically permuted across training batches. This ensures that each quantization level receives direct supervision during training.

Finally, to convert multi-level discrete codes into a single scalar token for downstream modeling, we define a surjective mapping:

$$\text{semantic\_id}([a_1, \ldots, a_n]) = \sum_{i=1}^{n} \text{card}^{n-i} \cdot a_i.$$

This scalar ID serves as the input to our discrete diffusion model in the second stage.

3.3 DIFFUSION MODEL TRAINING

Once the semantic tokenizer is trained, we proceed to generate training data for the diffusion model. For each source node in the graph, we use Personalized PageRank (PPR) to sample a sequence of D neighboring node IDs that captures the local graph structure. Each node ID in the sequence is then converted into its corresponding semantic token via the pre-trained RQ-VAE encoder. In this way, the training dataset consists of sequences of discrete tokens of length D.

We adopt the Order-Agnostic AutoRegressive Diffusion Model (OA-ARDM) framework (Hoogeboom et al., 2021) to learn the generative process over these token sequences. The diffusion model is designed to learn the reverse process of a forward noising (or masking) schedule. In our setting, the forward process gradually replaces tokens with an absorbing state $a = (N + 1)$, where N is the size of the vocabulary. More specifically, for a given token sequence $x = (x_1, x_2, \ldots, x_D)$, the forward process randomly masks tokens according to a schedule that is governed by a randomly selected timestep $t \in \{1, \ldots, D\}$.

The training procedure is as follows. First, a timestep t is drawn uniformly from $\{1, \ldots, D\}$. In parallel, a random permutation $\sigma \in S_D$ is sampled, where S_D denotes the set of all permutations of $\{1, \ldots, D\}$. This permutation determines the order in which tokens are to be revealed. A Boolean mask is then defined as

$$m = \mathbb{I}_{\{\sigma(i)<t\}} \quad \text{for } i = 1, \ldots, D,$$

where $\mathbb{I}_{\{\cdot\}}$ is the indicator function. The input of the diffusion model is constructed as

$$i = m \odot x + (1 - m) \odot a,$$

where $\odot$ denotes elementwise multiplication and a is a D-dimensional vector filled with the absorbing token value. The model, parameterized by a neural network f, is tasked with predicting the original tokens at the masked positions given the unmasked context.

To incorporate global structure information, we additionally condition the diffusion model on a graph-level embedding g, extracted from the original graph using a pre-trained graph encoder such as a GCN or GAT. This embedding is shared across all sequences from the same graph and is fused with the timestep embedding to guide the reverse process.

The model's prediction is formulated through a categorical distribution over the vocabulary for each masked token. The training objective is to maximize the average log-likelihood of the correct tokens at the masked positions. Formally, at each timestep t, the loss is given by

$$\mathcal{L}_t = \frac{1}{D - t + 1} \sum_{k \in \sigma(\geq t)} \log p(x_k \mid x_{\sigma(<t)}, g),$$

where $x_{\sigma(<t)}$ denotes the set of tokens that have not been masked according to the permutation σ. The final training loss is computed by averaging $\mathcal{L}_t$ over all choices of t and over all sampled permutations. The full training procedure for OA-ARDM is detailed in Algorithm 1.

In practice, the neural network f is implemented using a U-Net architecture, which takes token embeddings, timestep embeddings, and the graph-level conditioning vector as input. The network is trained end-to-end to predict the conditional probabilities for the masked tokens, effectively learning to reverse the forward noising process. This order-agnostic training scheme permits the model to generate tokens in a flexible, non-sequential manner, while the structural conditioning ensures coherence with the underlying graph topology.

3.4 INFERENCE AND DECODING

At inference time, we first encode a source node v into a scalar semantic ID using the pre-trained RQ-VAE. Starting from this token, the OA-ARDM generates a sequence of D semantic tokens by progressively predicting masked positions. To prevent repetitive generations, we set the sampling probability of previously generated tokens to zero at each step. We also apply softmax temperature $\tau > 0$ to control sampling entropy:

$$p_{\text{adjusted}}(x_i) \propto \exp\left(\frac{\log p(x_i)}{\tau}\right), \quad p(x_i) = 0 \text{ if } x_i \text{ already sampled.} \tag{1}$$

After generating the sequence of semantic IDs, we decode each token back into a continuous node embedding. Specifically, for each generated token, we reconstruct its embedding using the RQ-VAE decoder and then match it to the closest node in the dataset using k-nearest neighbor (KNN) search in the normalized feature space. This ensures that each generated neighbor corresponds to a real node in the graph:

$$\hat{\mathbf{h}}_i = \text{Decoder}(z_q^i), \quad v_i = \arg\min_u \left\| \hat{\mathbf{h}}_i - \mathbf{h}_u \right\|_2 .$$

The resulting node sequence forms the semantic neighborhood of the source node and can be used for downstream tasks such as classification, link prediction, or graph transformers.

Overall, our method combines the benefits of discrete diffusion modeling and robust quantization via the RQ-VAE framework to generate high-quality, semantically coherent neighborhoods in graphs.

4 EXPERIMENTS

In this section, we present comprehensive experimental results demonstrating the effectiveness and robustness of SemWalk. We show that: 1) SemWalk achieves performance on par with state-of-the-art methods in clean graph settings; 2) SemWalk demonstrates strong resilience to test-time graph perturbations, maintaining stable performance where baselines significantly degrade; 3) SemWalk can benefit further from longer random walk sequences and carefully calibrated decoding temperatures, enabling it to better capture semantic context; and 4) both graph conditioning and decoding are crucial for optimal performance, as removing either leads to clear declines in accuracy.

4.1 EXPERIMENT SETUP

Datasets. We evaluate on two large-scale multi-graph benchmarks, **PascalVOC-SP** and **COCO-SP**, from the LRGB suite (Dwivedi et al., 2022), which are well suited for testing generalization to unseen graphs. In both datasets, nodes correspond to image superpixels with 14-dimensional features (RGB statistics and 2D center coordinates), and edges connect spatially adjacent superpixels with weighted similarity. **PascalVOC-SP** contains 11,355 graphs with an average of 479.40 nodes, while **COCO-SP** includes 123,286 graphs averaging 476.88 nodes. We use the full PascalVOC-SP dataset and the first 10,000/2,000/2,000 COCO-SP graphs for training/validation/testing. Dataset statistics are summarized in Table 4.

Baselines. We compare SemWalk with five neighbor generation baselines. **Node Feature** uses only the source node's features (length 1). **PPR** (Page et al., 1999) selects top-h neighbors by proximity. **GALClean** (Chi et al., 2024) performs graph purification; we apply PPR on the purified graph to obtain sequences. **VCR-Graphormer** (Fu et al., 2024) augments PPR with structural and content tokens, evenly splitting the walk length. **SwapGT** (Chen et al., 2025) generates multiple neighborhood sequences via different similarity metrics and token swapping. All methods are treated as plug-in neighborhood generators, omitting method-specific losses.

Evaluation Protocol. All methods generate node sequences that are fed into a shared transformer classifier for node-level prediction. **Node Feature** uses sequences of length 1, while all other methods produce sequences of length $D = 1 + h$, where h denotes the number of neighbors obtained via PPR or generative sampling. Each node is classified based on its associated sequence, using the same transformer architecture across methods. To evaluate robustness, we perturb the test-time graph topology by randomly flipping edges with probability $p \in \{0.1, 0.2, 0.3\}$, while preserving graph connectivity. The perturbed graph is used for neighbor sampling when applicable, with node features and labels unchanged. This protocol assesses robustness to test-time structural noise. We report **classification accuracy**; additional details are provided in Appendix B.1.

4.2 MAIN RESULTS

SemWalk demonstrates strong resilience to test-time noise. In Table 1, we report the accuracy of node classification on different datasets. We can observe that under noisy settings ($p = 0.1$ to $p = 0.3$), SemWalk shows remarkable robustness on both PascalVOC-SP and COCO-SP. On

Table 1: Node classification accuracy under varying degrees of edge perturbation, where p denotes the probability of randomly flipping each edge during inference. Higher values of p correspond to noisier graph structures. The **bold** values indicate the best-performing method for each setting. Our method SemWalk consistently achieves the highest accuracy under noisy settings ($p = 0.1$ to $p = 0.3$), demonstrating strong robustness.

Method	PascalVOC-SP				COCO-SP			
	p=0	p=0.1	p=0.2	p=0.3	p=0	p=0.1	p=0.2	p=0.3
Node Feature	0.7161	0.7161	0.7161	0.7161	0.7074	0.7074	0.7074	0.7074
PPR	**0.7306**	0.6739	0.6719	0.6688	0.7143	0.6814	0.6795	0.6793
PPR + GALClean	**0.7306**	0.6833	0.6809	0.6778	**0.7145**	0.6867	0.6836	0.6844
VCR-Graphormer	0.7234	0.7116	0.7115	0.7112	0.7113	0.7027	0.7025	0.7024
SwapGT	0.7283	0.6839	0.6814	0.6794	0.7126	0.6895	0.6896	0.6903
SemWalk	0.7254	**0.7253**	**0.7253**	**0.7253**	0.7099	**0.7094**	**0.7093**	**0.7093**

both datasets, SemWalk consistently outperforms all baselines as noise increases. In contrast, PPR degrades sharply with increasing noise, dropping from 0.7306 to 0.6688 on PascalVOC-SP and from 0.7143 to 0.6793 on COCO-SP, reflecting its heavy reliance on local connectivity. VCR-Graphormer and SwapGT show similar degradation trends due to their dependence on handcrafted or structure-based neighborhood design. Applying GALClean before running PPR can slightly improve the performance, but it still underperforms SemWalk. Meanwhile, the Node Feature baseline remains flat, as it ignores graph structure entirely. We note that while our graph encoder leverages the graph structure, it remains robust due to its unsupervised training using the InfoNCE loss. These results validate the reliability of SemWalk in challenging conditions and its ability to generalize beyond fixed graph structures.

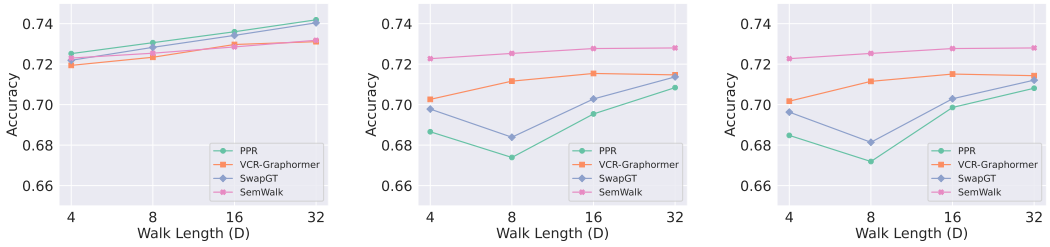
SemWalk rivals state-of-the-art methods on clean graphs. In the absence of graph perturbations ($p = 0$), SemWalk achieves comparable performance with the state-of-the-art methods, demonstrating SemWalk's capability of generalizing to unseen graphs. For instance, SemWalk achieves 72.54% accuracy on PascalVOC-SP, closely matching the best-performing baseline, Personalized PageRank (PPR), which reaches 73.06%. This demonstrates that SemWalk can generate informative and discriminative neighbor sequences purely from learned semantic priors, rivaling strong structure-based approaches that have full access to clean graph topology. Unlike PPR and token-engineered methods such as VCR-Graphormer and SwapGT, which depend heavily on explicit graph connectivity and handcrafted augmentations, SemWalk learns to model meaningful neighborhoods through a conditional discrete diffusion process. However, explicit graph connectivity and handcrafted augmentations provide advantages in noisy settings, allowing these methods, particularly VCR-Graphormer, which leverages node content information, to outperform PPR. The Node Feature baseline lags behind at 71.61%, confirming the importance of incorporating neighborhood context. These results show that even under ideal conditions for structure-based methods, SemWalk performs competitively while retaining greater flexibility and generalizability.

4.3 Ablation Studies

We conduct a series of ablation studies to better understand the design choices behind SemWalk and their impact on downstream node classification. Specifically, we investigate the effect of walk length, decoding temperature during sampling, and the role of key architectural components such as graph conditioning and semantic decoding. All ablations are performed on PascalVOC-SP unless otherwise specified.

Longer walk sequences improve classification performance. Figure 2 shows node classification accuracy on PascalVOC-SP as the walk length D varies under various noise levels. SemWalk and VCR-Graphormer consistently benefit from longer sequences across different noise levels. Notably, both SwapGT and PPR perform worst at $D = 8$ under noisy conditions, likely due to the inclusion of corrupted neighbors in the sampled sequences. This highlights SemWalk 's ability to leverage

extended context effectively, without overfitting or over-smoothing, unlike the baselines which are more susceptible to noise as sequence length increases.



(a) $p = 0$ (b) $p = 0.1$ (c) $p = 0.2$

Figure 2: Impact of walk length D on node classification accuracy on PascalVOC-SP under varying noise levels $p \in \{0, 0.1, 0.2\}$. SemWalk shows steady gains with increasing D, highlighting its ability to leverage extended context.

Table 2: Effect of sampling temperature τ on PascalVOC-SP under different noise levels p.

Temperature	p=0	p=0.1	p=0.2	p=0.3
$\tau = 0.1$	0.7244	0.7242	0.7241	0.7242
$\tau = 0.5$	0.7251	0.7252	0.7251	0.7253
$\tau = 1.0$	0.7254	0.7253	0.7253	0.7253
$\tau = 1.5$	0.7251	0.7242	0.7242	0.7242
$\tau = 2.0$	0.7258	0.7251	0.7251	0.7251

Table 3: Ablation study on PascalVOC-SP evaluating the role of decoding and graph conditioning under increasing noise. Removing either component leads to degraded accuracy.

Variant	p=0	p=0.1	p=0.2	p=0.3
SemWalk (Full)	0.7254	0.7253	0.7253	0.7253
w/o Decoding	0.7169	0.7165	0.7166	0.7165
w/o Graph Cond.	0.7223	0.7222	0.7222	0.7222

Proper temperature improves sampling quality. We evaluate different sampling temperatures $\tau \in \{0.1, 0.5, 1.0, 1.5, 2.0\}$ across various noise levels, with results shown in Table 2. On clean graphs, higher temperatures (e.g., $\tau = 2.0$) yield the best performance, with SemWalk achieving 0.7258. However, under noisy conditions, moderate temperatures ($\tau = 0.5$ or 1.0) perform best. This suggests that a controlled level of randomness during sampling helps mitigate noise by avoiding overfitting to corrupted inputs. Our use of temperature-controlled decoding effectively balances exploration and semantic consistency.

Graph conditioning and decoding are both essential. We assess the impact of removing either the graph conditioning or the decoding step. As shown in Table 3, removing decoding causes a performance drop to 0.7165, while omitting graph conditioning results in a moderate drop to 0.7222. The full version of SemWalk with both components achieves the highest accuracy at 0.7253. This highlights that both the global context from graph conditioning and the semantic mapping from discrete decoding are necessary for robust performance.

5 Conclusion

We introduced SemWalk, a generative framework for semantic neighbor discovery on graphs based on discrete diffusion modeling and residual quantization. Unlike prior heuristic or deterministic approaches, SemWalk learns to model the distribution over neighbor sequences in a permutation-invariant and semantically grounded space. By combining a Residual Quantized VAE tokenizer and an order-agnostic autoregressive diffusion model with graph-level conditioning, SemWalk achieves both diversity and robustness. Extensive experiments on two large-scale superpixel datasets show that SemWalk matches or outperforms strong baselines in clean settings and maintains superior accuracy under substantial test-time graph noise. We believe this work paves the way toward more principled and adaptive neighborhood generation for graph learning and opens new opportunities for generative modeling in structured domains.

REFERENCES

Tobias Bernecker, Ghalia Rehawi, Francesco Paolo Casale, Janine Knauer-Arloth, and Annalisa Marsico. Random walk diffusion for efficient large-scale graph generation. *arXiv preprint arXiv:2408.04461*, 2024.

Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International conference on machine learning*, pp. 695–704. PMLR, 2019.

Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2464–2473, 2020.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3438–3445, 2020.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. *arXiv preprint arXiv:2206.04910*, 2022.

Jinsong Chen, Chenyang Li, GaiChao Li, John E Hopcroft, and Kun He. Rethinking tokenized graph transformers for node classification. *arXiv preprint arXiv:2502.08101*, 2025.

Hongliang Chi, Cong Qi, Suhang Wang, and Yao Ma. Active learning for graphs with noisy structures. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pp. 262–270. SIAM, 2024.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, pp. 5722–5747. PMLR, 2023.

Tingting Dan, Jiaqi Ding, Ziquan Wei, Shahar Kovalsky, Minjeong Kim, Won Hwa Kim, and Guorong Wu. Re-think and re-design graph neural networks in spaces of continuous graph diffusion functionals. *Advances in Neural Information Processing Systems*, 36:59375–59387, 2023.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Christopher Fifty, Ronald G Junkins, Dennis Duan, Aniketh Iyengar, Jerry W Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. *arXiv preprint arXiv:2410.06424*, 2024.

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

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Rethinking tokenizer and decoder in masked graph modeling for molecules. *Advances in Neural Information Processing Systems*, 36:25854–25875, 2023.

Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *International Conference on Machine Learning*, pp. 24737–24755. PMLR, 2023.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Chuxiong Sun, Jie Hu, Hongming Gu, Jinpeng Chen, and Mingchuan Yang. Adaptive graph diffusion networks. *arXiv preprint arXiv:2012.15024*, 2020.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jevY-DtiZTR.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. *arXiv preprint arXiv:2405.11416*, 2024.

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

A Appendix

B Additional Details

B.1 Implementation Details.

We first train an RQ-VAE to quantize continuous node features into discrete semantic tokens. The tokenizer uses two quantization levels of size 32, producing a vocabulary of 1024 unique token IDs. Its encoder and decoder are implemented as 3-layer MLPs with hidden dimensions [18, 18] and an embedding size of 16. We train the graph encoder independently using a contrastive learning objective. Specifically, we use a two-layer GCN to encode graph-level embeddings from node features, where each graph is augmented twice via random edge dropout. The resulting embeddings are passed through a global mean pooling followed by a two-layer projection head. The encoder is trained using an InfoNCE loss that maximizes agreement between the two augmented views of the same graph. Training is done for 200 epochs using Adam with a learning rate of 1e-3 and weight decay of 1e-5.

The diffusion model follows the OA-ARDM framework and is implemented using a 1D U-Net. The model takes masked sequences of semantic tokens as input and is trained to recover the original tokens based on a randomly sampled timestep and permutation. To incorporate global structural context, the diffusion model is additionally conditioned on a graph-level embedding vector obtained from a separate graph encoder. The U-Net consists of contracting and expansive blocks with residual connections and uses 64-dimensional embeddings for both timestep and graph-level inputs.

Finally, for downstream evaluation, we train a Transformer classifier to predict node labels based on generated neighbor sequences. The Transformer takes as input a token sequence of length $D = 1+h$, where h is the hop size used in neighbor generation. It consists of one self-attention layer with 8 heads, a feed-forward dimension of 64, and a hidden size of 512. The model is trained using AdamW with polynomial decay scheduling, early stopping, and dropout applied to both feedforward and attention layers. All models are implemented in PyTorch and trained on NVIDIA RTX A6000 GPUs or NVIDIA GeForce RTX 3090 GPUs.

Algorithm 1 OA-ARDM Training

Require: Training set $\mathcal{D} = \{(x_1^{(i)}, \ldots, x_D^{(i)})\}_{i=1}^N$, absorbing token a, graph embeddings $\{g^{(i)}\}_{i=1}^N$

for each training step **do**
  Sample a training sequence $x = (x_1, \ldots, x_D)$ from $\mathcal{D}$
  Sample a random timestep $t \sim \text{Uniform}(1, D)$
  Sample a random permutation $\sigma \in S_D$
  Create a binary mask $m_i = \mathbb{I}_{\sigma(i)<t}$ for $i = 1, \ldots, D$
  $i \leftarrow m \odot x + (1 - m) \odot a$
  $\hat{x} \leftarrow f(i, g, t)$
  $\mathcal{L}_t \leftarrow \frac{1}{D-t+1} \sum_{k \in \sigma(\geq t)} \log p(x_k \mid x_{\sigma(<t)}, g)$
  Update model parameters with gradient step
end for

Table 4: Dataset statistics for PascalVOC-SP and COCO-SP. Each dataset contains graph representations of images, where nodes correspond to superpixels with 14-dimensional features, and edges indicate superpixel affinities.

Dataset	#Graphs	Avg. #Nodes	Feature Dim	Avg. Shortest Path	Avg. Diameter
PascalVOC-SP	11,355	479.40	14	10.74 ± 0.51	27.62 ± 2.13
COCO-SP	123,286	476.88	14	10.66 ± 0.55	27.39 ± 2.14

C Additional Results

To ensure the stability of our findings, we repeated the experiments in Table 1 on PascalVOC-SP for three independent runs and report the mean and standard deviation of the node classification accuracy in Table 5. For SemWalk, we use different temperatures each run: 0.5, 1.0, 1.5. We observe consistent trends across runs.

Table 5: Mean ± standard deviation of node classification accuracy on PascalVOC-SP under different edge perturbation probabilities p across three runs.

Method	p=0	p=0.1	p=0.2	p=0.3
Node Feature	0.7162 ± 0.0001	0.7162 ± 0.0001	0.7162 ± 0.0001	0.7162 ± 0.0001
PPR	**0.7307 ± 0.0001**	0.6773 ± 0.0024	0.6755 ± 0.0025	0.6725 ± 0.0026
VCR-Graphormer	0.7234 ± 0.0000	0.7116 ± 0.0000	0.7115 ± 0.0000	0.7112 ± 0.0000
SwapGT	0.7283 ± 0.0000	0.6839 ± 0.0000	0.6814 ± 0.0000	0.6794 ± 0.0000
SemWalk	0.7252 ± 0.0001	**0.7249 ± 0.0005**	**0.7249 ± 0.0005**	**0.7249 ± 0.0005**