ADAPTIVE HETEROGENEOUS GRAPH REPRESENTATION LEARNING USING KNN-AUGMENTED GRAPH MAMBA NETWORKS (KA-GMN)

Eishkaran Singh
Thapar Institute of Engineering and Technology
esingh3_be21@thapar.edu

ABSTRACT

Graph representation learning for heterogeneous networks presents challenges in structural preservation and computational tractability. We present KA-GMN (KNN-Augmented Graph Mamba Networks), integrating k-nearest neighbor selection with state space models for graph representation learning. The architecture implements: (1) KNN-based state transitions for type-specific node representation, (2) compatibility functions for structural graph adaptation, and (3) type-aware feature transformations to prevent representation degradation. KA-GMN processes multi-typed relationships through selective message passing and state space modeling, maintaining graph structure through learned neighborhood functions. The theoretical framework establishes a foundation for heterogeneous graph representation through the synthesis of KNN-based topology and state space dynamics.

1 INTRODUCTION

Graph representation learning is crucial for processing relational data in artificial intelligence Hamilton et al. (2018). However, contemporary neural architectures face limitations when handling complex real-world graphs, primarily due to over-smoothing Li et al. (2019), over-squashing Alon & Yahav (2021), and computational complexity in attention-based transformers Dwivedi & Bresson (2021). While Graph Mamba Networks (GMNs) partially address these issues through State Space Models (SSMs) Gu & Dao (2024), they struggle with heterogeneous graph elements and structural variations. This work introduces KNN-Augmented Graph Mamba Networks (KA-GMN), advancing graph representation learning through three key contributions:

1. **Enhanced heterogeneous representation**: KA-GMN integrates K-nearest neighbor metrics within the state space model, enabling type-aware representation of heterogeneous nodes and edges. This approach preserves node discriminability and mitigates over-smoothing Li et al. (2019) in complex graph structures.
2. **Dynamic neighborhood adjustment**: Learned compatibility functions adapt to structural graph variations, addressing over-squashing by facilitating efficient long-range dependency propagation. This mechanism flexibly adjusts to evolving graph structures, ensuring effective information capture regardless of complexity.
3. **Type-specific feature transformation**: Novel operators maintain semantic relationships while preventing representation collapse, preserving the heterogeneous nature of complex graphs throughout the learning process.

This architecture enables efficient processing of multi-typed relationships, scalable global dependency capture, and balanced local structure preservation, advancing the field of graph representation learning for complex, heterogeneous graph structures.

2 BACKGROUND

The field of graph representation learning has evolved through sequential methodological advancements. Kipf & Welling (2017b) established Graph Convolutional Networks (GCNs) using spectral graph convolutions for node classification, demonstrating superior performance over manual feature engineering through neighbor aggregation.

Veličković et al. (2018) extended this paradigm with Graph Attention Networks (GATs), introducing dynamic attention weights for heterogeneous node relations. Subsequent analysis by Li et al.

(2018) revealed the over-smoothing phenomenon in deep GNN architectures, where repeated message passing layers cause node representation collapse.

The introduction of Graph Transformers by Yun et al. (2020) addressed locality constraints through self-attention mechanisms, though computational complexity limited scalability. Dwivedi & Bresson (2021) proposed sparse attention variants to mitigate these costs. Shirzad et al. (2023) advanced this through exponential sparsification patterns in EXPHORMER, though attention-based computation persisted as a constraint. The KNN-Augmented Graph Mamba Network (KA-GMN) synthesizes SSM efficiency with KNN-based neighborhood adaptation. This architecture diverges from EXPHORMER's fixed sparsification by implementing selective state transitions and type-aware neighbor selection, addressing heterogeneity while maintaining O(n) complexity.

Recent innovations in State Space Models (SSMs) by Gu & Dao (2024) provided linear-time alternatives for sequence modeling, prompting adaptations to graph structures. For heterogeneous graphs, Wang et al. (2021) developed hierarchical attention mechanisms in HAN, though scalability remained challenge due to attention dependencies. Geometric deep learning frameworks (Fey & Lenssen (2019)) and simplified GCN variants (Chen et al. (2020)) expanded methodological diversity.

3 METHODOLOGY

Drawing from the foundational work in graph neural networks Kipf & Welling (2017a) and state space modeling Gu & Dao (2024), we present a theoretical framework for heterogeneous graph processing that combines selective k-nearest neighbors with state space models.

3.1 GRAPH REPRESENTATION AND STATE SPACE MAPPING

We define a heterogeneous graph $G = (V, E)$, where V represents the node set and E the edge set. Each node $v \in V$ is characterized by a feature vector $x_v \in \mathbb{R}^d$ and a node type $t_v \in T_t$. Edges $e \in E$ connect nodes v_i and v_j with an edge type $r_{ij} \in R$. The graph structure is encoded in two primary matrices. 1)Node Feature Matrix: $X \in \mathbb{R}^{|V| \times d}$, where each row x_v corresponds to the features of node v. 2) Edge Feature Matrix: $E \in \mathbb{R}^{|E| \times e_d}$, capturing edge features.

3.2 K-NEAREST NEIGHBORS (KNN) INTEGRATION AND ADJUSTMENT

Building on the theoretical framework of Dong et al. (2011), we compute the K-nearest neighbors for each node $v \in V$ based on its feature vector x_v. The selective mechanism extends this base KNN approach:

$$N_K^{\text{selective}}(v) = u \in N_K(v) \mid \gamma(u, v) > \tau(v)$$

where $\gamma(u, v)$ is a learned compatibility function:

$$\gamma(u, v) = \text{MLP}([h_u; h_v; |h_u - h_v|; h_u \odot h_v; r_{\text{type}}(u, v)])$$

Following spectral graph theory, we construct the dynamic adjacency matrix:

$$A_K(i, j) = \begin{cases} 1 & \text{if } j \in N_K^{\text{selective}}(i) \text{ or } i \in N_K^{\text{selective}}(j), \\ 0 & \text{otherwise.} \end{cases}$$

The degree matrix D_K and normalized adjacency matrix $\tilde{A}_K$ are defined as:

$$D_K = \text{diag}\left(\sum_j A_K(i, j)\right)$$

$$\tilde{A}_K = D_K^{-1/2} A_K D_K^{-1/2}$$

The adaptive choice of K follows:

$$K_v = \min\left(\max\left(\lceil \log(\deg(v)) \rceil, K_{\min}\right), K_{\max}\right)$$

3.3 ENHANCED NEIGHBORHOOD WEIGHTS

We introduce two theoretically grounded formulations for neighborhood weights:

MLP-based weights:

$$W_K(i, j) = \text{softmax}j(\text{MLP}\theta([x_i; x_j; |x_i - x_j|; x_i \odot x_j]))$$

Similarity-based weights:

$$W_K(i,j) = \exp(-\alpha \cdot d_{ij} + \beta \cdot \text{TypeSim}(t_i, t_j) + \gamma \cdot \text{EdgeSim}(r_{ij}))$$

These weights incorporate d_{ij} is the Euclidean distance between x_i and x_j, $\text{TypeSim}(t_i, t_j)$ measures similarity between node types, $\text{EdgeSim}(r_{ij})$ evaluates similarity of edge types, α, β, γ are learned parameters.

3.4 KNN-Enhanced Message Passing Mechanism

Building upon standard message passing Gilmer et al. (2017), our Graph Mamba Network (GMN) introduces a KNN-based adaptive aggregation:

$$m_v^{(l)} = \sum_{u \in N_K(v)} W_K(v,u) \cdot (\tilde{A}_K(v,u) \cdot h_u^{(l-1)})$$

where $W_K(v,u)$ and $\tilde{A}_K(v,u)$ refine local message computation by dynamically adjusting neighborhood importance. The hidden state update follows a selective state-space approach inspired by Gu et al. (2022):

$$h_v^{(l)} = \text{S4}(h_v^{(l-1)}, m_v^{(l)})$$

but differs by integrating KNN-enhanced message passing, allowing for more adaptive and structured information flow.

3.5 Enhanced State Space Model Integration

The selective message passing mechanism is defined as:

$$m_v^{(l)} = \sum_{u \in N_K^{\text{selective}}(v)} W_K(v,u) \cdot (\tilde{A}K(v,u) \cdot h_u^{(l-1)} \cdot w\text{type}(t_v, t_u))$$

Building on Gu & Dao (2024), the state evolution follows modified Mamba SSM equations:

$$s'(t) = (\bar{A} \odot D(x_t))s(t) + (\bar{B} \odot E(x_t))x_t$$
$$y(t) = (\bar{C} \odot F(x_t))s(t)$$

with selective masks as:

$$D(x_t) = \sigma(\text{MLP}_D([x_t; \text{AggN}(x_t)]))$$
$$E(x_t) = \sigma(\text{MLP}_E([x_t; \text{AggE}(x_t)]))$$
$$F(x_t) = \sigma(\text{MLP}_F([x_t; \text{AggV}(x_t)]))$$

3.6 Long-Range Dependencies Handling

Inspired by attention mechanisms Vaswani et al. (2023), we capture long-range dependencies:

$$h_{\text{long},v} = \sum_{u \in D_v} \beta_u \cdot h_u \cdot \phi(d_{vu})$$

where β_u represents the attention coefficient and is computed using a softmax function applied to the output of an MLP, which takes as input the concatenation of node features h_v and h_u Similarly, $\phi(d_{vu})$ is defined using a softmax operation applied to a combination of a distance-based term and an MLP transformation of node features. The local and long-range features combine as:

The local and long-range features combine as:

$$h_{\text{final},v} = z \cdot h_{\text{local}} + (1 - z) \cdot h_{\text{long}}$$

where: z represents $\sigma(W_z \cdot [h_{\text{local}} \parallel h_{\text{long}}] + b_z)$

3.7 Graph-to-Sequence Conversion with KNN-aware Ordering

The conversion process follows a structured approach to encode node representations and establish an order based on local neighborhood information. The encoding of node features follows prior methodologies, while the ordering mechanism extends existing work by incorporating a KNN-aware priority function:

$$\text{Order}(v) = \text{Priority}(h_{\text{final},v}, N_K^{\text{selective}}(v))$$

ensuring that nodes are prioritized based on both their final representations and their selectively chosen KNN neighbors, which enhances context preservation in the sequence.

3.8 MAMBA PROCESSING OF ENHANCED GRAPH SEQUENCE

Following established sequence processing methods, the Mamba model processes the transformed sequence by embedding it and applying state-space modeling. We retain the fundamental Mamba architecture but modify the timestep operations to introduce a selective multi-layer perceptron (MLP)-based split mechanism with timestep operations as

$$\begin{aligned} \Delta, B, C &= \text{Split}(\text{MLP}(x_t)) \\ y_t &= \Delta \odot (Ax_t) + B \odot x_t \\ z_t &= C \odot \sigma(y_t) \end{aligned}$$

This adjustment ensures that feature interactions at each timestep are adaptively partitioned, allowing for more expressive transformations within the Mamba model.

3.9 HETEROGENEITY ALIGNMENT

To align heterogeneous node representations effectively, we introduce type-specific transformations and multi-scale type integration.

$$h_{\text{type}}(v) = \sum_{i=1}^{L} w_i \cdot \text{TypeConv}i(v, N_K^{\text{selective}}(v))$$

This formulation ensures that different type transformations contribute variably based on learned importance weights. Furthermore, type-aware attention aggregation is introduced to refine neighborhood message passing:

$$h_{\text{aligned},v} = \sum_u \alpha_u \cdot h_u$$

where α_u represents $\text{softmax}u(a^T \cdot \text{LeakyReLU}(W \cdot [h_v \parallel h_u \parallel e_{vu}]))$

3.10 KNN-AWARE READOUT AND TASK-SPECIFIC LAYERS

For graph-level readout, we introduce a selective softmax-based aggregation mechanism:

$$h_G = \sum_v \text{softmax}v(\text{MLPreadout}(h_{\text{final},v})) \cdot h_{\text{final},v}$$

This ensures that node contributions are adaptively weighted based on task-specific importance. For node-level tasks, the output is computed as: $y_v = \text{MLPtask}(h\text{final}, v)$ ensuring effective downstream adaptation, maintaining consistency with prior architectures while allowing for enhanced feature interaction.

3.11 TRAINING OBJECTIVE AND REGULARIZATION

The overall learning process is governed by a composite loss function that balances multiple objectives to enhance model performance and generalization:

$$L = L_{\text{task}} + \lambda_1 L_{\text{KNN}} + \lambda_2 L_{\text{struct}} + \lambda_3 L_{\text{type}} + \lambda_4 L_{\text{reg}}$$

where:

$$\begin{aligned} L_{\text{KNN}} &= \sum_{i,j} |h_{\text{final},i} - h_{\text{final},j}|^2 \cdot A_K(i,j) \\ L_{\text{struct}} &= |\tilde{A}_K - \text{softmax}(H_{\text{final}} H_{\text{final}}^T)|_F^2 \\ L_{\text{type}} &= \sum_{t \in T_t} |H_t - \text{Center}(H_t)|_F^2 \\ L_{\text{reg}} &= \lambda_{\text{decay}} \sum_i |\theta_i|^2 \end{aligned}$$

4 CONCLUSION

This work introduces KNN-Augmented Graph Mamba Networks (KA-GMN), establishing a framework for heterogeneous graph representation learning through the integration of selective state space models with KNN-based topology. By integrating type-specific node representations, structural graph adaptation, and selective message passing mechanisms, KA-GMN addresses critical challenges in graph neural networks. The framework advances the theoretical foundations for processing heterogeneous graphs by bridging local structural preservation with global dependency capture, enabling efficient representation learning on complex networked systems. Through its structured approach to heterogeneous graph modeling, KA-GMN establishes a foundation for developing scalable graph learning systems and opening new directions for investigating the convergence of state space models and graph neural architectures.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications, 2021. URL https://arxiv.org/abs/2006.05205.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks, 2020. URL https://arxiv.org/abs/2007.02133.

Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pp. 577–586. Association for Computing Machinery, 2011. doi: 10.1145/1963405.1963487. URL https://doi.org/10.1145/1963405.1963487.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL https://arxiv.org/abs/1903.02428.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017. URL https://arxiv.org/abs/1704.01212.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications, 2018. URL https://arxiv.org/abs/1709.05584.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017a. URL https://arxiv.org/abs/1609.02907.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017b. URL https://arxiv.org/abs/1609.02907.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns?, 2019. URL https://arxiv.org/abs/1904.03751.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, 01 2018. doi: 10.1609/aaai.v32i1.11604.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs, 2023. URL https://arxiv.org/abs/2303.06147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. Heterogeneous graph attention network, 2021. URL https://arxiv.org/abs/1903.07293.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks, 2020. URL https://arxiv.org/abs/1911.06455.