Adaptive Multi-Order Node Embeddings

Amro Asali

June 2025

Abstract

Standard node embeddings of a fixed size in Graph Neural Networks (GNNs) often fail to generalize across diverse downstream tasks, as the structural requirements for node classification and link prediction are frequently in conflict. This work investigates architectural solutions to this problem by evaluating established adaptive models, namely Jumping Knowledge Networks (JKN) and MixHop, against a standard GCN baseline. Furthermore, we propose a novel hybrid architecture that integrates GCN and MixHop layers under a Jumping Knowledge aggregation framework to balance feature extraction across multiple scales with an adaptive receptive field. We conduct a rigorous empirical study on the Cora and Amazon Photo datasets, encompassing settings for single task learning, cross task transfer, and multi task learning. Our results starkly highlight the core challenge, showing that embeddings trained for one task exhibit a significant performance drop when transferred to another. We find that JKN provides robust and balanced performance, while our proposed hybrid model achieves leading results on several combinations of tasks and datasets, particularly for link prediction. Overall, our findings validate that architectures with adaptive aggregation are essential for effective graph representation learning for multiple tasks and demonstrate the potential of hybrid designs to reconcile competing structural objectives.

1 Introduction

Traditional graph representation learning maps each node to a fixed size embedding, an approach that is highly effective when optimized for a single prediction task like node classification. However, a fundamental limitation arises when these embeddings are applied to a diverse set of tasks. Their performance often deteriorates on higher order tasks, such as link prediction, due to conflicting structural and symmetry requirements.

The core of this conflict lies in two opposing failure modes. On one hand, embeddings optimized for node classification must preserve fine grained local distinctions to differentiate node roles. This can make them too rigid and localized, failing to capture the broader relational context needed to model pairwise

compatibility for link prediction. Conversely, embeddings trained for link prediction tend to *oversmooth* node representations. In learning to model structural similarity, they often average out the unique characteristics of individual nodes, degrading the discriminative power essential for node level tasks.

These issues arise because a single, fixed size vector has limited capacity to simultaneously satisfy these competing objectives. This motivates the move toward adaptive or structured embedding such as dynamically sized vectors or subgraph based representations that can break these symmetries and generalize effectively across different prediction orders. By design, these advanced methods can capture more expressive structural properties, enabling a single model to succeed at both distinguishing individual node roles and modeling complex network relationships.

2 Background

2.1 Jumping Knowledge Networks

Jumping Knowledge (JK) Networks [1] enable flexible aggregation of multi-hop neighborhood information by combining representations from different GNN layers. Instead of relying on a fixed receptive field (e.g., 2-hop), JK networks allow each node to adaptively select relevant neighborhood ranges.

The choice of aggregation function in Jumping Knowledge (JK) Networks dictates its node adaptivity. We explore adaptive strategies like **Max-Pooling**, which implicitly selects the most salient features from different layers on a per-node, parameter-free basis, and **LSTM-attention**, which explicitly learns per-node weights for each layer. These contrast with the non-adaptive **Concatenation** method, which applies a single, globally shared transformation to all nodes, thereby learning an average combination rule for the entire dataset rather than a node-specific one.

Given a stack of L GNN layers producing intermediate representations $\{h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(L)}\}$ for node v, JK networks apply a layer aggregation function:

$$h_v^{\mathrm{JK}} = \mathrm{Aggregate}\left(h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(L)}\right)$$

MaxPool: A simple non-learned strategy where the final representation is the element-wise maximum over all layers:

$$h_v^{\mathrm{JK}} = \max_{l=1,\ldots,L} h_v^{(l)}$$

This allows each dimension to be chosen from the most informative layer, breaking uniformity across nodes.

LSTM: A learned aggregation strategy where an LSTM processes the sequence $\{h_v^{(1)}, \ldots, h_v^{(L)}\}$ and the final hidden state is used as h_v^{JK}. This captures layer-wise dependencies and adaptively fuses multi-hop information:

$$h_v^{\mathrm{JK}} = \mathrm{LSTM}(h_v^{(1)}, \ldots, h_v^{(L)})$$

JK networks produce *structured* and optionally *dynamic* multi-order embeddings. When using fixed strategies like MaxPool or concatenation, the embeddings are structured but static. When using LSTM or attention, the embeddings become dynamic, allowing per-node adaptation across different receptive fields.

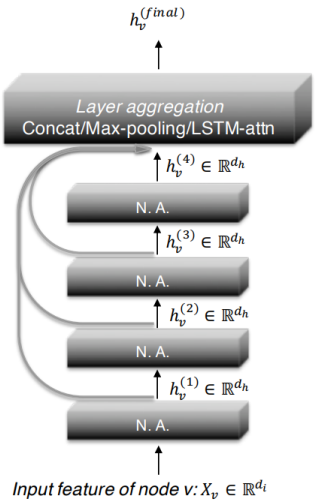

Figure 4. A 4-layer Jumping Knowledge Network (JK-Net). N.A. stands for neighborhood aggregation.

Figure 1: JKnet

MixHop MixHop is a multi-hop message passing framework that captures neighborhood information at different hop distances within a single GNN layer [2]. Unlike standard GNNs that aggregate only 1-hop neighbors per layer, MixHop simultaneously mixes features from multiple powers of the adjacency matrix. Formally, each layer computes:

$$h^{(l)} = \Big\|_{p \in \mathcal{P}} \sigma\left(\mathbf{A}^p x \mathbf{W}^{(l,p)}\right)$$

where $\mathcal{P}$ is a predefined set of hop distances (e.g., $\{1, 2, 3\}$), $\mathbf{A}^p$ denotes the p-th power of the normalized adjacency matrix, and $\mathbf{W}^{(l,p)}$ are separate learnable weights for each hop. This hop-wise mixing is concatenated and passed to the next layer. MixHop produces *structured*, *multi-scale* embeddings by encoding multiple neighborhood depths in parallel. However, unlike Jumping Knowledge Networks, MixHop does not adaptively select hops per node, it relies on fixed hop patterns defined globally for the model.

2.2 Symmetry-Breaking Techniques

To enhance the expressiveness of GNNs and distinguish structurally similar nodes, several symmetry-breaking strategies have been proposed. These include

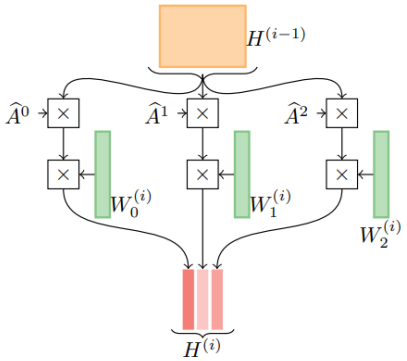

Figure 2: MixHop Layer

adding structural node identifiers such as Weisfeiler-Lehman (WL) subtree labels, shortest-path distances to anchor nodes, or simple structural features like node degree. These techniques introduce node-specific signals that can disambiguate otherwise isomorphic neighborhoods, improving the ability of GNNs to model roles and structural positions.

In our experiments, we investigated two such approaches: appending WL subtree labels and node degree as additional features to the node embeddings. However, these enhancements did not yield significant performance improvements on the Cora dataset across node classification and link prediction tasks thus the results for these experiments are not included in this report. This suggests that while theoretically beneficial, structural encodings may require more sophisticated integration or may offer limited gain in already well structured datasets like Cora. Hence, we shifted our focus toward architectural adaptations like Jumping Knowledge and MixHop, which provide a more flexible and learnable way of handling multi order information.

2.3 Datasets

2.3.1 Cora

The *Cora* dataset is a citation network commonly used for benchmarking graph neural networks. It consists of 2,708 scientific publications classified into seven categories. Each node represents a paper, and edges denote citation links between them. The node features are represented as a 1,433-dimensional binary vector indicating the presence of specific words from a predefined dictionary. The task is typically node classification, where the goal is to predict the category of each paper based on its features and citation links. To evaluate both node classification and link prediction, we use a dual data splitting strategy. For node classification, we use the default node masks of the dataset. For link prediction, we generate a new split, partitioning the edges into an 85% training, 5% validation, and 10% test set, with negative samples automatically added for

the binary classification task.

2.4 Amazon Photo

The *Amazon Photo* dataset is a subset of the Amazon co-purchase graph, where nodes represent products and edges indicate that two products are frequently bought together. Each node is associated with a feature vector extracted from product reviews, and labeled with the product category it belongs to. This dataset naturally supports node classification, where the task is to predict the product category, and link prediction, where the goal is to infer co-purchase relationships between products. It comprises 7,487 nodes and 119,043 edges, randomly split into 60-20-20% for training validation and test partitions respectively .

3 Identifying a solid architecture for both tasks

In this experiment we tried to identify the best architecture, that can handle both node classification and link prediction tasks, all hyperparameters were chosen empirically, beginning with the same hyperparameters specified in [1].

3.0.1 Node Classification Experiments

We conducted a comprehensive comparison of several multi-layer graph neural network (GNN) architectures on the Cora dataset for the node classification task. All models were implemented using PyTorch Geometric and trained using cross-entropy loss with early stopping based on validation accuracy.

- **GCN:** Following the findings in [1], we implemented a standard 2-layer Graph Convolutional Network (GCN) [3] with hidden dimension 32, ReLU activation, and dropout 0.5 to serve as a baseline GNN.
- **JKNet (Max Pool):** The Jumping Knowledge Network (JKNet) [1] with max pooling aggregates outputs from all GCN layers and applies a linear classifier. We used 6 GCN layers, each with hidden dimension 32, followed by a max-pooling readout and dropout 0.5 replicating the top performing model from [1] on cora dataset.
- **MixHop:** The MixHop encoder [2] combines multiple powers of the adjacency matrix at each layer, enabling explicit mixing of features across different hop distances, followed by a linear readout layer. We used two layers with hop powers $\{0, 1, 2\}$ and hidden dimensions 32 per hop.
- **Hybrid-JKN:** We designed a hybrid model that alternates between GCN and MixHop layers, using Jumping Knowledge (max pooling) for layer aggregation and a final linear classifier.

  Resulting in a $JKnet(\text{GCN}_1 \rightarrow \text{MixHop}_1 \rightarrow \text{GCN}_2)$ all with hidden dimension 32 and dropout 0.4, the motivation for this design is to create a

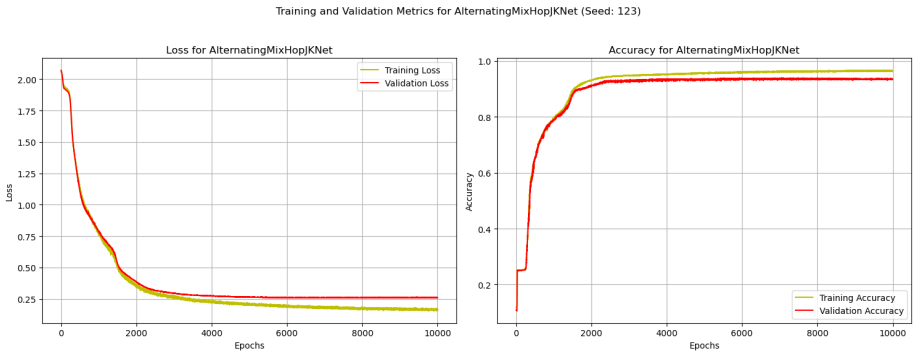

Figure 3: proposed system loss and accuracy for node classification task on Amazon Photo dataset

more diverse and expressive set of candidate representations for the final Jumping Knowledge (JK) aggregation. Unlike a standard JKNet which uses a homogeneous stack of identical layers, our hybrid design alternates between standard `GCN` layers and a `MixHop` layer. This heterogeneous stack provides the JK aggregator with fundamentally different types of embeddings to choose from: a *low-order* representation from the initial GCN layer, a rich *mixed-order* representation from the MixHop layer, and a *high-order* representation from the final GCN layer. Consequently, the model can learn to adaptively select not just the optimal receptive field *depth* for each node, but also the most suitable representational *texture*, leading to a more robust and task-appropriate final embedding.

For all architectures, we normalized node features, employed Cora/Amazon photo datasets. We tracked training and validation losses and accuracies for all runs. Model selection was based on the best validation accuracy.

Model	**Cora Accuracy**	**Amazon Photo Accuracy**
GCN	0.7630	0.9059
JKNet-Max	0.7640	0.9307
MixHop	0.7880	0.8719
Hybrid-JKN:	0.7870	**0.9379**

Table 1: Comparison of mean test accuracy for different GNN architectures on Cora and Amazon Photo datasets.

3.0.2 Link Prediction Experiments

We evaluated several GNN-based encoders on the Cora/Amazon photo datasets for the link prediction task. All models produce fixed dimensional node embed-

dings and use a shared MLP decoder for edge scoring. Evaluation is done using ROC AUC.

- **GCN:** A 2-layer Graph Convolutional Network with hidden dimension 32, ReLU activation, and dropout 0.5.
- **JKNet (Max Pool):** A 6-layer GCN backbone with hidden dimension 32 and max-pooling Jumping Knowledge aggregation [1].
- **MixHop:** A single-layer MixHop encoder [2] with hidden dimension 32 and hop powers $\{0, 1, 2\}$, resulting in a 96-dimensional output vector.
- **Hybrid-JKN:** A hybrid encoder alternating between GCN and MixHop layers (3 layers total), each with hidden dimension 32 and dropout 0.4, using max Jumping Knowledge aggregation.

Model	Cora AUC	Amazon Photo AUC
GCN	0.7944	0.9448
JKNet-Max	0.7996	0.9640
MixHop	0.7116	0.8284
Hybrid-JKN:	**0.8357**	**0.9647**

Table 2: Mean test AUC for link prediction on Cora and Amazon Photo datasets using different GNN architectures.

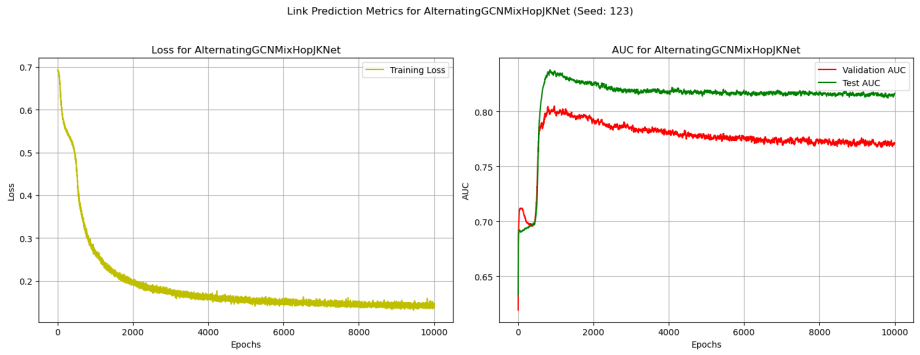

Figure 4: proposed system loss and AUC plots for link prediction task on Cora dataset

Findings. In the node classification task, **MixHop** and **Hybrid-JKN** achieved the highest test accuracies (0.7880 and 0.7870), outperforming both GCN and JKNet-Max. This highlights the benefit of integrating multi-hop neighborhood information either explicitly through MixHop's adjacency powers or structurally

via hybrid architectures. However, the trend differed in the link prediction task: the **Hybrid-JKN** model obtained the best AUC (0.8357), while **GCN** and **JKNet-Max** followed with 0.7944 and 0.7996, respectively. Notably, MixHop performed poorly on link prediction (AUC 0.7116), suggesting that fixed hop combinations may be less effective when modeling pairwise relationships. These results indicate that while MixHop is advantageous for local classification, hybrid or adaptively aggregated architectures generalize better across tasks particularly in capturing structural similarity for link prediction.

3.0.3 Cross-Task Evaluation: Link Prediction to Node Classification

To assess the generality of learned node embeddings, we performed a cross-task evaluation. Each model was first trained on the link prediction task, and then its learned node embeddings were evaluated by training a linear classifier for node classification, without fine-tuning the encoder. The results are summarized in Table **??**.

Table 3: Cross-task evaluation: models trained on link prediction, evaluated on node classification (test accuracy).

Model	**Cora**	**Amazon Photo**
GCN	0.2380	0.4444
JKNetMax	**0.3020**	0.4092
MixHop	0.2860	0.2569
Hybrid-JKN	0.2340	**0.4458**

Findings. While all models achieved reasonably high AUC on the link prediction task, the transfer of embeddings to node classification resulted in significantly lower accuracy compared to direct supervised training. The JKNetMax showed the best transfer performance on the Cora dataset, while GCN exhibited weaker generalization across tasks on Cora dataset, while Hybrid-JKN achieved the best transfer accuracy on Amazon photo. These results suggest that embeddings optimized solely for link prediction are suboptimal for node classification, highlighting the challenge of learning universally useful representations in multi-task graph settings.

3.1 Multi-Task Model Architecture

We evaluate four GNN encoders under a unified multi-task learning framework that jointly optimizes for node classification and link prediction on the Cora dataset. For all models, we set the hidden dimension to 32 and apply dropout between layers. Each encoder is followed by two separate heads: one for node classification and one for binary link prediction, we made sure our proposed system's number of parameters is similar to the baseline's ($\tilde{5}$0K).

- **GCN:** A standard Graph Convolutional Network with 2 layers. Both layers use hidden dimension 32 and ReLU activations. The output from the final layer is passed directly to the task heads.
- **JKNet (Max / LSTM):** A 2-layer GCN backbone where outputs from each layer are aggregated using Jumping Knowledge [1]. We experiment with both max pooling and LSTM-based aggregation. Each GCN layer has hidden dimension 32.
- **MixHop:** A single MixHopConv layer with hop powers $\{0, 1, 2\}$, each using 32 hidden units. The resulting 96-dimensional output is used for both downstream tasks.
- **Hybrid-JKN** A 2-layer hybrid architecture with alternating structure: `GCN → MixHop` . The GCN layer uses 32 hidden units. The MixHopConv layer applies hop powers $\{0, 1, 2\}$ with 32 hidden units per hop, followed by a linear projection to 32 dimensions. Jumping Knowledge (max) aggregates outputs from both layers.

For training, we adopt a randomized alternating strategy [4] where at each step, either the node classification or link prediction loss is sampled and optimized. Final evaluation is based on validation accuracy (node classification) and AUC (link prediction).

Model	**Cora**		**Amazon Photo**	
	Accuracy	AUC	Accuracy	AUC
GCN	0.7710	**0.9088**	0.9124	**0.9680**
JKNetMax	**0.8000**	0.9042	0.9150	0.9613
JKNetLSTM	0.7810	0.9026	0.9092	0.9580
MixHop	0.7430	0.8683	0.8641	0.9624
Hybrid-JKN	0.7430	0.8836	**0.9183**	0.9584

Table 4: Multi-task node classification and link prediction performance on Cora and Amazon Photo datasets (Seed 123).

Findings Jumping Knowledge Networks (JKNet) effectively address the adaptive multi-order embedding challenge by aggregating features from multiple GNN layers, allowing nodes to flexibly utilize information from different neighborhood depths. This adaptability supports both node classification and link prediction by reconciling task-specific symmetry requirements. In our experiments, JKNet-Max achieved the highest node classification accuracy on the Cora dataset (**0.8000**) and competitive AUC (**0.9042**), while the simple GCN baseline yielded the best AUC (**0.9088**). On the Amazon Photo dataset, the Hybrid-JKN model achieved the best test accuracy (**0.9183**), while GCN again

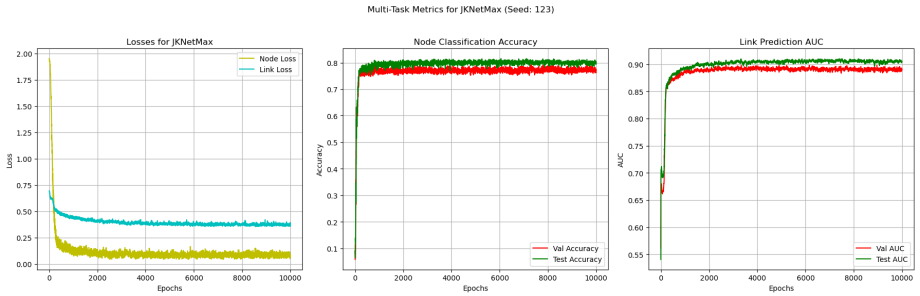

Figure 5: JKnet with maxpooling, loss and AUC plots for link prediction task

attained the highest AUC (**0.9680**). These results reinforce that JKNet generalizes well across datasets and tasks, while GCN remains a strong baseline for link prediction. Although the Hybrid alternating architecture showed strong performance, its added complexity did not consistently outperform simpler models across both tasks and datasets.

4 Conclusion

In this work, we addressed the challenge of learning adaptive multi-order node embeddings by evaluating several architectural strategies under both single-task and multi-task settings across Cora and Amazon Photo datasets. Our experiments demonstrate that Jumping Knowledge Networks (JKNet), which adaptively aggregate multi-hop information via max pooling or LSTM, offer a robust mechanism for reconciling the differing structural demands of node classification and link prediction. We also proposed a novel hybrid architecture that alternates between GCN and MixHop layers before applying JK-based aggregation, which achieved the best accuracy on Amazon Photo. Furthermore, we experimented with explicit symmetry-breaking techniques such as Weisfeiler-Lehman (WL) labels and node degrees, but these did not significantly improve performance. Overall, our findings validate the benefit of adaptive multi-order aggregation strategies like JKNet for multi-task learning, supporting the core motivation of the challenge and generalizing effectively across datasets.

References

[1] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5453–5462.

[2] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, and G. Ver Steeg, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research. PMLR, 2019, pp. 52–61.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016, published as a conference paper at ICLR 2017.

[4] A. Asali, Y. Ben-Shimol, and I. Lapidot, "Atmm-saga: Alternating training for multi-module with score-aware gated attention sasv system," 2025. [Online]. Available: https://arxiv.org/abs/2505.18273