Research Proposal

Haolan Zuo

Problem Definition

PSE: Position and Structural Encoding

Graph Neural Networks (GNN) are the dominant paradigm in graph representation learning, of which applications are across a wide range of domains including biomedicine, molecule, social networks, documentation, transportation etc. Message-Passing Neural Network (MPNN) framework is the backbone of GNNs, which iteratively update a node' feature by aggregating messages from nodes in immediate neighborhood. However, though MPNN leverages well the local structural information in the graph, long range information and position information are not captured thus the expressiveness is upper bounded by 1-WL.

Recently, thanks to the rising of Transformer architecture in Natural Language Processing (NLP), more expressive Graph Transformers (GTs) are introduced to the graph area which naturally handle well long range information with global attention mechanism. However, unlike natural language with intrinsic sequential structure, general graphs lack a canonical node ordering and even in some graphs nodes are arbitrarily permutable. Therefore, Positional and Structural Encodings (PSEs) are in need to identify different nodes only based on their relative positions and local structures.

Thus far, while many different types of PSEs have been hand-crafted and used by various GT models, they are mostly domain-specific or task-specific which can hardly be generalized or transferred directly to other datasets. For example, a PSE method aligning well with a task may fail to work effectively in another.

Therefore, this research aims to solve the above problem by proposing a general PSE method suitable for arbitrary graphs which is not domain-specific or task-specific. To produce general PSEs, the model are trained not dependent on any node or edge features except input graphs. Furthermore, the general PSE model should be able to be pre-trained in a large amount of datasets from various fields. The general PSE model can be implemented on top of any graph models.

Since some PSEs like Laplacian based position encodings are low-dimensional, which can be easily transformed into vectors and concatenated with each other, quadratic or pair-wise features like shortest-path distances are hard to inject. As simple concatenation is naive but always effective, the primary work of this research is to implement some position-aware model to capture the shortest path encoding for each node and then generalize it to more informative PSEs.

Dataset

Training Dataset

For the training phase, this research proposes the utilization of the following datasets, each catering to different domains to ensure that the general PSE model is both generalizable and transferable:

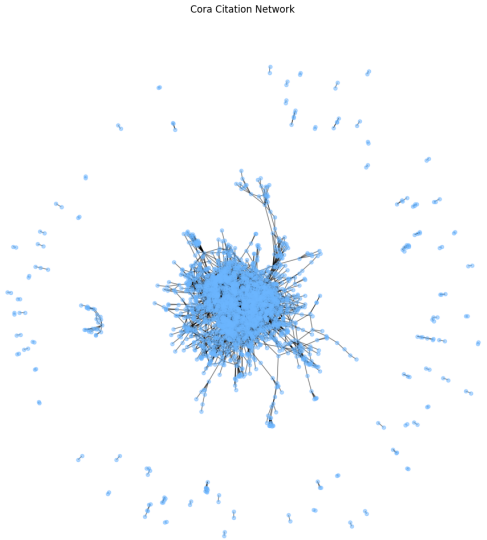
1. **Cora Dataset and Pubmed Dataset**: Citation network datasets that are widely used for node classification tasks in academia. Widely used in academia for benchmarking Graph Neural Networks. Ideal for understanding the citation linkages between papers and how information propagates through the academic community.
2. **Reddit Dataset**: A social network dataset that involves community prediction and other social network-related tasks. With its complex and large-scale social interactions, this dataset is apt for testing the scalability and effectiveness of the proposed PSE methods in social networks.
3. **ZINC Chemical Dataset**: A molecular graph dataset suitable for capturing the chemical structural attributes, which provides a variety of molecular structures, ideal for examining the model's capacity to generalize across chemical domains.
4. **Synthetic Graphs**: Randomly generated graphs with various topologies to ensure that the model is not overfitting to specific graph structures.

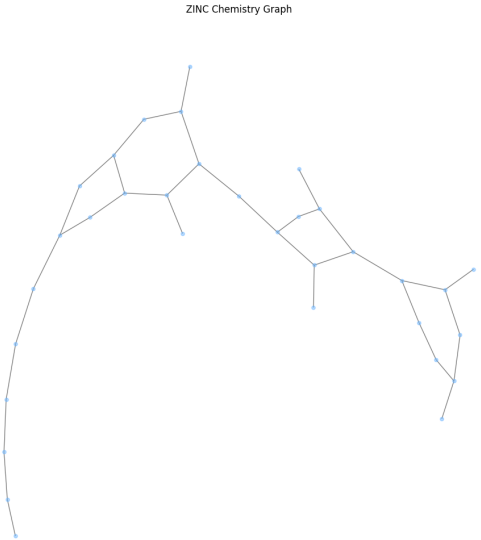
Downstream Evaluation Dataset

After pre-training on above datasets, the general PSE model is evaluated on downstream tasks. For the phase, the following datasets are proposed to gauge the model's performance in different application scenarios:

1. **CiteSeer Dataset**: Another citation network dataset similar to Cora and Pubmed but with distinct features and network structure. It serves as a good basis for evaluating how well the model generalizes across similar but not identical academic citation networks.
2. **PPI (Protein-Protein Interaction) Dataset**: A complex biological network dataset often used for multi-label node classification. The PPI dataset provides a robust test of the model's ability to generalize its learned PSE methods to biomedical applications.
3. **Facebook Pages Dataset**: A social network dataset involving community structures within Facebook pages, providing another dimension to evaluate the model's effectiveness in social network settings.

Dataset	Domain	#Nodes	#Edges	#Features	Tasks	Dataset Link
Cora	Citation Network	2,708	10,556	1,433	Node classification	PyG Cora
PubMed	Citation Network	19,717	88,648	500	Node classification	PyG PubMed
Reddit	Social Network	232,965	114,615,892	602	Community prediction	PyG Reddit
ZINC	Chemistry	~23.2	~49.8	1	Property prediction	PyG ZINC
Synthetic Graphs	N/A	Varies	Varies	None	Generalization tests	N/A
CiteSeer	Citation Network	3,327	9,104	3,703	Node classification	PyG CiteSeer
PPI	Biomedicine	~2,245.3	~61,318.4	50	Multi-label Classification	PyG PPI
Facebook Pages	Social Network	22,470	342,004	128	Community prediction	PyG FacebookPagePage





Description of related works

Positional and Structural Encodings (PSEs) have become a critical component in augmenting the expressiveness of Graph Neural Networks (GNNs). These encodings were initially inspired by the Transformer architecture in Natural Language Processing (NLP), where sinusoidal functions capture the ordinal positions of words [1]. Unlike natural language, which has a sequential structure, graphs lack a canonical node ordering. This discrepancy calls for specialized PSE methods to capture node positions and structures in graphs.

Positional Encodings

Various methods have emerged for positional encoding in graphs. One of the most prevalent approaches is the use of graph Laplacian eigenvectors, which serve as analogues to sinusoids in Euclidean space [2, 3, 4, 5, 6]. Other notable techniques involve electrostatic potential encodings [4] and shortest-path distances [7].

Structural Encodings

Structural encodings focus on capturing local and global connectivity patterns in graphs. The random walk encoding has gained considerable attention for its ability to distinguish between r-regular graphs [8]. This encoding method has shown excellent performance when integrated with Graph Transformers (GTs), particularly in molecular graph benchmarks [3, 2, 9]. Additional approaches in structural encodings include heat kernel methods [4, 10] and node degree centralities [7].

Unified PSE

While numerous PSE methods have been developed, they often suffer from domain-specific or task-specific limitations. Further, it remains poorly explored how to effectively combine multiple types of PSEs to enhance GNN performance [11, 12]. While there is a recent study that casts light on the unified PSE task, the study suffers from lacking of convincingness of the non-position-aware GNN architecture and the design of objective function [13].

This research aims to address these challenges by proposing a generalized PSE method that has well designed position-aware encoder and refined objective function that learns how to embed non low-dimensional features like shortest path.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[2] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. AAAI Workshop on Deep Learning on Graphs: Methods and Applications, 2021.

[3] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable graph transformer. Advances in Neural Information Processing Systems, 35:14501–14515, 2022.

[4] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. Advances in Neural Information Processing Systems, 34:21618–21629, 2021.

[5] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of ViT/MLP-Mixer to graphs. arXiv:2212.13350, 2022.

[6] Daniel Spielman. Spectral graph theory, volume 18. CRC Press Boca Raton, Florida, 2012.

[7] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., ... & Liu, T. Y. (2021). Do transformers really perform badly for graph representation?. *Advances in Neural Information Processing Systems*, *34*, 28877-28888.

[8] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. Advances in Neural Information Processing Systems, 33:4465–4478, 2020.

[9] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In International Conference on Learning Representations, 2022.

[10] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. GraphiT: Encoding graph structure in transformers. CoRR, abs/2106.05667, 2021.

[11] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. arXiv preprint arXiv:2202.13013, 2022.

[12] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In International Conference on Machine Learning, pages 3469–3489. PMLR, 2022.

[13] Liu, R., Cantürk, S., Lapointe-Gagné, O., Létourneau, V., Wolf, G., Beaini, D., & Rampášek, L. Graph Positional and Structural Encoder. arXiv:2307.07107, 2023.

Proposed Approaches

Hitherto, although there are other non-low-dimensional PSEs, such as subgraph isomorphism, this study has focused primarily on the shortest path. Also as mentioned, the architecture needs to be position-aware and structure-aware. Thus far, two methods are considered worth investigating. One is use a Graph Transformer itself to learn and output position encoding that captures quadratic shortest path information. The other is use shortest-path anchor-based GNN, which is also known as Position-aware GNN (P-GNN).

For the downstream task, while other MPNNs can also be applicable, GTs are put placed at the forefront of this research because of their natural need of position encodings.

Shortest Path Distance Encoder

Fig 1 shows the pipeline of the aforementioned Shortest Path Distance (SPD) encoder. The input of encoder are graphs without any features because what is aimed to capture is only the structural and positional information of graphs. During preprocess stage, SPD matrix is calculated as learning targets. The initial positional encoding is provided if Graph Transformer architecture is used as the following encoder, which can be calculated using any low-dimensional common positional encoding methods.

Notably, pair-wise SPD information acts as self-supervisor in the loss function. This learning process can be achieved adopting ideas from contrastive learning. Specifically, a similarity function is used to measure the distance of output encodings of a pair of nodes, with supervision on the actual SPD of the nodes. More detailed loss function is described in subsequent Evaluation Metrics section.

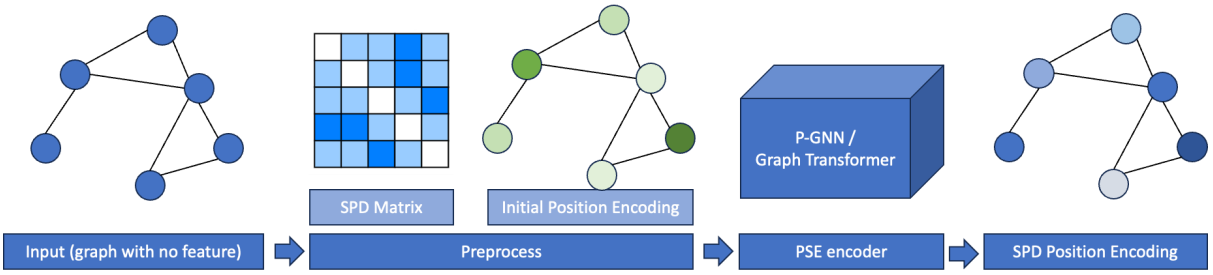


Fig 1 SPD position encoder

SPD/diameter

SPD/max distance

More importantly, this training process can be done simultaneously on various datasets mentioned before. As a result, a general encoder can be pre-trained independent on specific datasets or tasks and can be easily plugged into any downstream tasks.

Downstream Graph Transformer

With pre-trained encoder, the low-dimension SPD position encoding can also be concatenated with other common used SPEs. The ultimate SPEs are passed to any MPNNs as node features or to any Transformers architectures as positional encodings.

As the SPD PE is pre-trained on a large scale of datasets, it is thought to performs well on capturing shortest path information in unseen datasets with good transferability and generalizability.

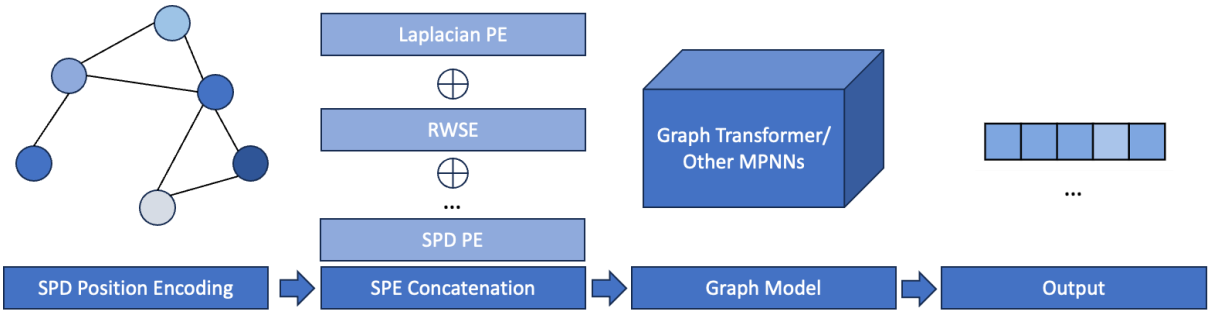


Fig 2 Downstream pipeline

Evaluation Metrics

As is mentioned before, there are two stages for PSEs encoder, which are the pre-training stage and the downstream stage. Accordingly, two different evaluation methods are applied to evaluate the performance of the PSE model. Thereinto, one is designed to test whether the PSE encoder can capture non low-dimension position features like shortest path given graphs without additional features. The other one is designed to test whether the gained PSEs can help to improve the performance of GTs (or other graph models) handling node-level/edge-level/graph-level tasks.

Self-Supervised Evaluation (Loss Function)

As mentioned earlier, this section will discuss how to evaluate the encoding effectiveness for the shortest path during the pre-training phase.

Actually, the effectiveness can be described well by the loss function. In the loss function, a similarity function is designed to measure the similarity of any two nodes.

Given a pair of nodes i and j. Let $\mathbf{z}_i$ and $\mathbf{z}_j$ be the encoding of shortest path respectively, and $SP_{i,j}$ be the shortest path between them. Specifically, Radial Basis Function Kernel (RBF) can be used to as the similarity function, which can be formularized as:

$$K\left(\mathbf{z}_i, \mathbf{z}_j\right)=\exp\left(-\frac{\left\|\mathbf{z}_i-\mathbf{z}_j\right\|^2}{2 \sigma^2}\right) \in(0,1]$$

For simplicity of training, replacement can be made that $\gamma=\frac{1}{2 \sigma^2}$. To approximate shortest path, the global loss function can be written as

$$\mathcal{L}=\frac{1}{\binom{N}{2}} \sum_{i=1}^N \sum_{j=i+1}^N\left(K\left(\mathbf{z}_i, \mathbf{z}_j\right)-S P_{i, j}^{-1}\right)^2$$

Downstream Task Evaluation

When it comes to down stream task, the evaluation methods should rely on specific tasks. Take node classification task as an example. Cutting edge GTs and GNNs will be implemented for the task and comparative experiments will be held to test whether pre-trained PSE module could make positive impact.

As for node classification tasks, assuming we have k labels. Let y_i be the label and $\hat{y}_i$ be the prediction. The Cross-Entropy loss function can be written as:

$$\mathcal{L}_{\mathrm{CE}}=-\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^k y_{i, j} \log \left(\hat{y}_{i, j}\right)$$

Commonly used evaluation metrics in multi-classification are as follows.

1. **Accuracy**:
   - Accuracy is the ratio of the number of correct predictions to the total number of predictions. It is given by:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. **Precision**:
   - Precision is the ratio of the number of true positives to the sum of true positives and false positives. It is given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Recall**:
   - Recall is the ratio of the number of true positives to the sum of true positives and false negatives. It is given by:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. **F1 Score**:
   - F1 Score is the harmonic mean of Precision and Recall, and is given by:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Timeline

Roughly, there are two months for the project. A primary timeline plan per week is as follows:

- Week 1: Deep dive into the relevant research area, conduct extensive literature review, and finalize the framework for the proposed model.
- Week 2: Start implementing the initial version of the model, focusing on the core components and functionality.

- Week 3: Continue implementing the model while simultaneously reading relevant papers to gather more insights and ideas for improvement.
- Week 4: Refine and optimize the implemented model based on initial evaluation results. Identify areas for enhancement and plan for further iterations.
- Week 5: Conduct thorough evaluation of the model's performance, gather quantitative and qualitative results, and analyze the findings.
- Week 6: Based on the evaluation results, make necessary improvements to the model, including fine-tuning hyperparameters and adjusting the architecture if needed.
- Week 7: Perform code refactorization to enhance the code quality, readability, and maintainability of the model implementation.
- Week 8: Summarize the work done throughout the research project, write the final report detailing the methodology, results, and conclusions. Prepare for the submission of the research proposal.

The timeline is a rough estimate and may be adjusted based on the progress made and any unforeseen challenges encountered during the project. To be more general, there are total three stages for the project: (1) Implement the proposed model while keeping reading; (2) Improve the model and refactorize existing codes; (3) Final evaluation and write reports.