Path Integral Based Convolution and Pooling for Heterogeneous Graph Neural Networks

Lingjie Kong
Department of Computer Science
Stanford University
Stanford, CA 94305
ljkong@stanford.edu

Yun Liao
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
yunliao@stanford.edu

Abstract

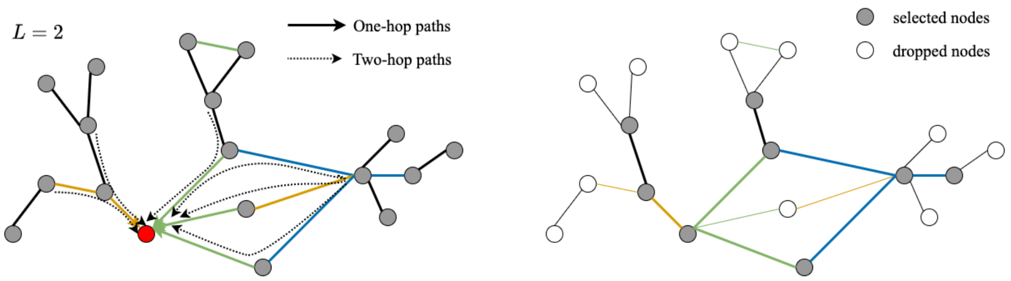
Graph neural networks (GNN) extends deep learning to graph-structure dataset. Similar to Convolutional Neural Networks (CNN) using on image prediction, convolutional and pooling layers are the foundation to success for GNN on graph prediction tasks. In the initial PAN paper [1] , it uses a path integral based graph neural networks for graph prediction. Specifically, it uses a convolution operation that involves every path linking the message sender and receiver with learnable weights depending on the path length, which corresponds to the maximal entropy random walk. It further generalizes such convolution operation to a new transition matrix called maximal entropy transition (MET). Because the diagonal entries of the MET matrix is directly related to the subgraph centrality, it provide a trial mechanism for pooling based on centrality score. While the initial PAN paper only considers node features. We further extends its capability to handle complex heterogeneous graph including both node and edge features.

1 Introduction

1.1 Background

With the success of applying Convolutional Neural Networks (CNN) on with fix-size 2D image dataset, researchers have dived deeper into how to apply deep learning on graph dataset. Graph neural networks (GNNs) especially Graph Convolutional Neural Networks (GCN) provides a great framework as baseline. An essential part of GCN is message passing. Message passing not only allow us to encoder richer node feature, but also enables tasks such as node, edge, or even graph prediction. One specific method is to use the graph Laplacian based methods relying on message passing between connected nodes with equal weights across edges. The idea of generic random walk (GRW) defined on graphs is at heart of many graph Laplacian based methods that essentially rely on message passing between directly connected nodes (eg. GCN [2], GraphSAGE [3], etc.). Although proved effective in many graph-based tasks, the GRW-based methods inherently suffer from information dilution as paths between nodes branch out. This can pose great difficulty on graph-level interpretation tasks especially when the multi-hop local structures matter as much as the global node attributes.

Ma, Xuan, et al. proposed a path-integral-based graph neural networks (PAN) approach in [1] that overcomes this drawback by considering every path linking the message sender and receiver as the elemental unit in message passing, which is analogous to Feynman's path integral formulation [4] extensively used in statistical mechanics and stochastic processes. Similar ideas have been shown effective in link prediction [5] and community detection [6] tasks. The popular graph attention mechanism [7] can also be viewed as a special case of PAN by restricting the maximal entropy transition (MET) matrix to a particular form. Another stream of related works are the GNN models



(a) PANConv: Aggregate messages from each path connected to the central node, where the path length is at most L. Different colors represent different edge types.

(b) PANPool: obtain the subgraph consisting of K out of N nodes with the highest scores and the edges connecting them.

Figure 1: Visual illustration of the PANConv and PANPool approaches.

using multi-scale information and/or higher-order adjacency matrix, such as LanczosNet [8], N-GCN [9] and SGC [10].

This project re-implements and improves upon the PAN approach and evaluates its performance on a molecular classification dataset named ogbg-molhiv.

1.2 Method: PAN and HPAN

The PAN approach consists two major modules: (1) a PAN convolution module (PANConv) that gives the message passing rule in the graph, and (2) a PAN pooling module (PANPool) that specifies the approach to extract higher-level features of the graph for the final graph-level tasks.

The PANConv module is illustrated in Fig. 1a. Each message passing step aggregates neighborhood information up to L hops away. The message from node A to node B is weighted according to the number of paths between A and B and the lengths of the paths. This can be considered as an extension to the idea of aggregating information from one-hop neighborhood in most of the GNN approaches.

The PANPool module computes the score of each node in the graph and selects a subset of nodes with the highest scores. The subgraph induced from the selected nodes is passed to the next layer. All the other nodes and edges are dropped. The PANPool module is illustrated in Fig. 1b.

The original PAN approach can only deal with node features, and no edge attributes are considered. As an extension to the PAN approach, this project proposes heterogeneous PAN (HPAN) to incorporate edge features in the message passing rule, so that the approach is able to handle heterogeneous graphs. In particular, we add a PanLump layer as shown in figure 3 to sum corresponding node embedding and feature embedding together during message passing.

1.3 Dataset

This project uses the ogbg-molhiv dataset, which is adopted from the MoleculeNet, to evaluate the performance of PanConv and hybrid PanPool approach. The ogbg-molhiv dataset contains 41,127 graphs. Each graph represents a molecule, where nodes are atoms, and edges are chemical bonds. The input node features are 9-dimensional, containing atomic number, chirality, etc. The edge features are 3-dimensional, representing bond type, bond stereochemistry, and conjugation, respectively. The task associated with this dataset is to predict whether a target molecule inhibits HIV virus replication or not, which is a binary classification task. ROC-AUC metric will be used for evaluation.

This dataset is selected to evaluate the PAN and HPAN algorithms for the following reasons: the key idea of PANConv is to smartly exploit the structural features of the graph in each layer of convolution, while on the other hand, PANPool provides a way to combine the node features with the local graph structure. This can be very useful in understanding graphs representing molecules, where the structural information can be essential. Moreover, the output of the selected algorithm can

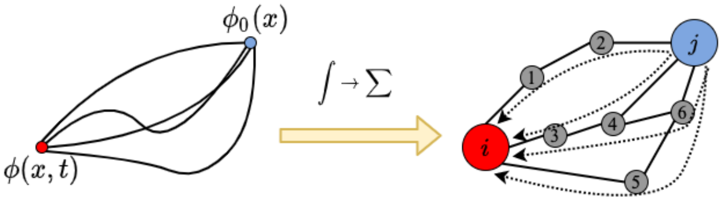


Figure 2: Schematic analogy between the original path integral formulation in continuous space (left) and the discrete version for a graph (right).

be naturally interpreted as a graph-level representation, which makes it convenient for a graph-level classification task.

2 Existing Method

This section details the PAN approach, which is **NOT YET in the OGB leaderboard**. The PAN approach consists two major components: PANConv as the convolution operator and PANPool as the pooling operator.

2.1 PANConv

The core of PANConv is a discrete analogy of Feynman's path integral formulation [4] that can be applied to a graph. The analogy is illustrated in Fig. 2. In the original path integral formulation, the probability amplitude $\phi(\boldsymbol{x}, t)$ is influenced by the surrounding field, where the contribution from $\phi_0(\boldsymbol{x})$ is computed by summing over the influences (denoted by $e^{iS[\boldsymbol{x},\dot{\boldsymbol{x}}]}$) from all paths connecting itself and $\phi(\boldsymbol{x}, t)$:

$$\phi(\boldsymbol{x}, t) = \frac{1}{Z}\int \phi_0(\boldsymbol{x}) \int e^{iS[\boldsymbol{x},\dot{\boldsymbol{x}}]}\mathcal{D}(\boldsymbol{x}), \tag{1}$$

where Z is the partition function.

A graph can be viewed as a discrete version of a continuous field. The path integral formulation is generalized to a graph setting in [1] by replacing the integral over paths to a discrete sum over all possible paths in the graph, and the integral of Lagrangian $e^{iS[\boldsymbol{x},\dot{\boldsymbol{x}}]}$ to a sum over Boltzmann's factor. The analogous path integral formulation on graph $G(V, E)$ then becomes

$$\phi_i = \frac{1}{Z_i}\sum_{j\in V}\phi_j \sum_{\{\boldsymbol{l}|l_0=i,l_{|\boldsymbol{l}|}=j\}} e^{-\frac{E[\boldsymbol{l}]}{T}} \overset{(*)}{=} \frac{1}{Z_i}\sum_{l=0}^{\infty} e^{-\frac{E(l)}{T}} \sum_{j\in V} g(i,j;l)\phi_j, \tag{2}$$

where ϕ_i denotes the feature/message at node i, $\boldsymbol{l} = (l_0, l_1, \dots, l_{|\boldsymbol{l}|})$ is a path connecting node i and node j, $E[\boldsymbol{l}]$ represents the fictitious energy w.r.t. path $\boldsymbol{l}$, and T is the fictitious energy. Z_i is the partition function for node i. Equation $(*)$ holds under the assumption that the fictitious energy only depends on the path length l, in which $g(i, j; l)$ denotes the number of length-l paths between nodes i and j. Presumably, the energy $E(l)$ is an increasing function of l. Note that $g(i, j; l)$ can be computed efficiently from the graph's adjacency matrix A as $(A^l)_{ij}$. To make the computation tractable, a cutoff maximal path length L is applied to (2), and the path integral formulation is simplified as

$$\phi_i = \frac{1}{Z_i}\sum_{l=0}^{L} e^{-\frac{E(l)}{T}} \sum_{j\in V}(A^l)_{ij}\phi_j. \tag{3}$$

The above expression can be written in a compact form by defining the MET matrix

$$M = Z^{-1}\sum_{l=0}^{L} e^{-\frac{E(l)}{T}} A^l, \tag{4}$$

where $Z = \operatorname{diag}(Z_i)$. The name comes from the fact that it realizes maximal entropy under the microcanonical ensemble. Based on the MET matrix, a convolutional layer can be defined as

$$X^{(h+1)} = M^{(h)} X^{(h)} W^{(h)}, \tag{5}$$

where h is the layer index and $W^{(h)}$ is the trainable weight. A PANConv module further improves over (5) by applying symmetric normalization instead of Z^{-1}. The final definition of PANConv is

$$X^{(h+1)} = Z^{-1/2} \sum_{l=0}^{L} e^{-\frac{E(l)}{T}} A^l Z^{-1/2} X^{(h)} W^{(h)}. \tag{6}$$

2.2 PANPool

The diagonal element $M_i i$ of the MET matrix resembles the subgraph centrality defined by $\sum_{l=1}^{\infty}(A^l)_{ii}$ for node i, so $\operatorname{diag}(M)$ provides a natural characterization of the nodes' importance. On the other hand, the global importance can be represented by, but not limited to, the strength of the message X itself. PANPool projects feature $X \in \mathbb{R}^{|V| \times d}$ by a trainable parameter vector $p \in \mathbb{R}^d$ and combines it with $\operatorname{diag}(M)$ to obtain a score vector

$$\text{score} = Xp + \beta \operatorname{diag}(M) \tag{7}$$

Here, $\beta \in \mathbb{R}$ controls the emphasis on these two potentially competing factors. PANPool then selects a fraction of the nodes ranked by this score (number denoted by K), and outputs the pooled feature array $\tilde{X} \in \mathbb{R}^{K \times d}$ and the corresponding adjacency matrix $\tilde{A} \in \mathbb{R}^{K \times K}$.

The feature array of the final layer of the stacked "PANConv + PANPool" structure is averaged and then fed into the task-oriented layers, which can be trained against any ground truth label through specific loss functions.

3 Method

This section elaborates the modifications over the original PAN approach that could bring improvements on the classification task in the dataset of interest.

3.1 HPAN with edge features

The original PAN approach can only handle homogeneous graph without edge attributes, and only node attributes are being propagated in the graph. However, the dataset of interest (ogbg-molhiv) includes edge attributes, direct application of the PAN approach to this dataset would simply ignore the edge attributes and treat the edges as if they are homogeneous, which would lose information and potentially result in inferior classification performance (as shown by the experimental results in Section 4.

Our method named HPAN proposes to incorporate the edge features in the PAN approach by introducing an additional module named PANLump.

The PANLump module is added before the stack of "PANConv + PANPool" layers as shown in Fig. 3. This module adds transformed edge features to the incident node features. The PANLump module works as follows. First, apply `atom_encoder` to all nodes and `edge_encoder` to all edges, where the embedding dimensions of both encoders match each other. Then, each edge (u, v) shares its embedding $\boldsymbol{h}(u, v)$ to both of its vertices, and the incident node embeddings $\boldsymbol{X}(u)$ (as well as $\boldsymbol{X}(v)$) update as

$$\boldsymbol{X}'(u) \leftarrow \mathbf{MLP}[(1+\epsilon)\boldsymbol{X}(u) + \mathbf{AGG}\{\boldsymbol{h}(u, v),\ \forall v \in \mathcal{N}(u)\})]. \tag{8}$$

The output of the PANLump module is the updated node features that incorporate its surrounding edge attributes. This output is then fed into the following "PANConv + PANPool" structure, which assumes homogeneous edges.

3.2 Loss function

The ogbg-molhiv dataset is highly skewed in the sense that it contains 39,684 samples (not inhabit HIV) and only 1,443 positive samples (inhibit HIV). To make sure that the model does not learn

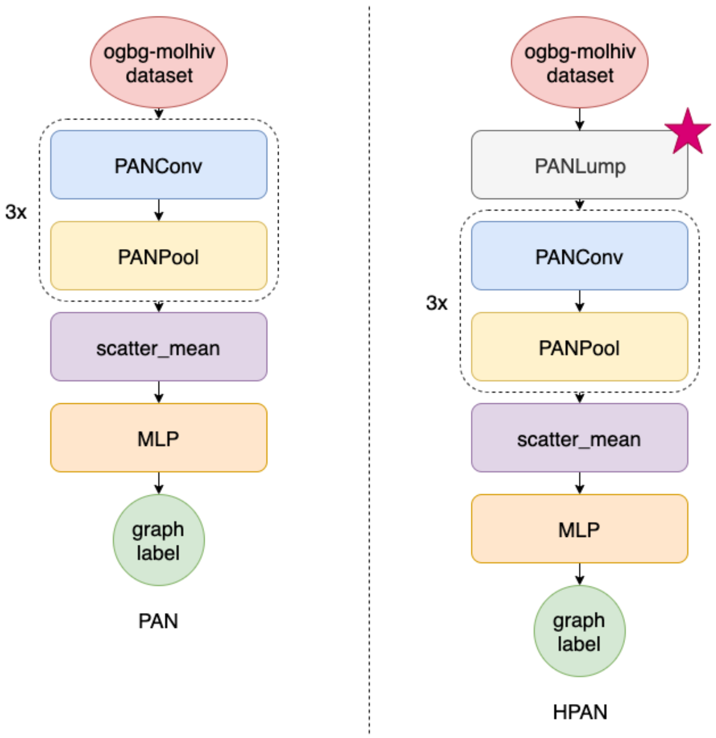


Figure 3: Comparison of a PAN structure and the corresponding HPAN structure.

Table 1: Comparison of ROC-AUC scores of ogbg-molhiv dataset

Model	Add. Feat.	Virtual Node	ROC-AUC Val	ROC-AUC Test	#Params
GCN	No	Yes	83.73 ± 0.78	74.18 ± 1.22	1,978,801
GCN	Yes	No	82.04 ± 1.41	76.06 ± 0.97	527,701
GCN	Yes	Yes	83.84 ± 0.91	75.99 ± 1.19	1,978,801
GIN	No	Yes	84.1 ± 1.05	75.2 ± 1.30	3,336,306
GIN	Yes	No	82.32 ± 0.90	75.58 ± 1.40	1,885,206
GIN	Yes	Yes	**84.79** ± 0.68	**77.07** ± 1.49	3,336,306
PAN	Yes	No	70.17 ± 0.29	73.06 ± 0.49	**26,843**
HPAN	Yes	No	82.27 ± 0.44	76.76 ± 0.41	**43,676**

trivially to always output 0, a weighted binary cross entropy loss is adopted in training. In particular, the loss function puts larger weight on positive sample than on negative ones:

$$loss = \frac{1}{|\mathcal{D}|}\sum_{n=1}^{|\mathcal{D}|}\left[\alpha y^{(n)}\log\left(\hat{y}^{(n)}\right) + \left(1-y^{(n)}\right)\log\left(1-\hat{y}^{(n)}\right)\right], \tag{9}$$

where $\hat{y}^{(n)}$ is the predicted probability of the n-th graph sample has positive label, and the weight $\alpha \geq 1$.

4 Experiments and Discussions

Both PAN and HPAN are implemented and evaluated on the OGB ogbg-molhiv dataset [11]. The evaluation metric is the ROC-AUC score. The performance is compared to the benchmarks given in [11]. In particular, the benchmark includes GCN and GIN with or without using additional features and virtual nodes. The results are shown in Table 1.

In the experiments, both PAN and HPAN use 3 layers of PANConv and PANPool followed by a mean pooling module to extract a graph-level representation. The output representation is then fed into a two-layer MLP of size $\{\text{emb\_dim}, \text{emb\_dim}/2, 1\}$ and ReLU activation. Cutoff length $L = 3$ for PANConv1, and $L = 2$ for PANConv2 and PANConv3. At each PANPool layer, 80% of the nodes are selected according to the score function, i.e., $|V|^{(h+1)} = K = 0.8|V|^{(h)}$. The MLP in the PANLump module has size $\{\text{emb\_dim}, \text{emb\_dim} * 2, \text{emb\_dim}\}$ with BatchNorm and ReLU activation. The sum aggregation is adopted as $\mathbf{AGG}(\cdot)$ in PANLump. The embedding dimension is set to 64, and the weight in loss function is set to $\alpha = 5.0$ in PAN and $\alpha = 10.0$ in HPAN. The ROC-AUC scores and variations are calculated based on 10 independent runs.

As shown in the table, the original PAN without considering edge features does not show very satisfying results. Instead, the modified version HPAN shows comparable performance to the benchmarks. The deviation, on the other hand, is significantly reduced compared to the benchmarks, which may be due to the consideration of higher order paths. Besides, since each PANConv layer considers information in up to L-hop neighborhoods, fewer layers are needed to extract the information from the entire graph. Therefore, PAN and HPAN can be extremely light-weighted. This is verified in Table 1 in the sense that the number of trainable parameters is several orders of magnitude smaller than the parameters needed for the benchmarks.

5 Conclusion

The PAN approach, which takes multi-hop message passing into account in each graph convolutional layer, was implemented and applied to the graph classification task for the ogbg-molhiv dataset. A HPAN method was proposed as an extension to the original PAN approach that enables the model to incorporate edge attributes. Experimental results showed that the original PAN approach gave inferior performance compared to the existing benchmarks due to the ignorance of edge features, while the modified HPAN method achieved comparable performance to the benchmarks with a extremely small model. Besides, PAN and HPAN also showed significantly smaller variance in performance across independent runs. In this perspective, PAN provides a more efficient and stable learning model for the graph classification task. Future works would include smarter ways to incorporate edge features in each layer of PANConv or PANPool and theoretical comparison between wide PANs (i.e., large L) and deep GNN.

References

[1] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. *arXiv preprint arXiv:2006.16811*, 2020.

[2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[3] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[4] Richard P Feynman, Albert R Hibbs, and Daniel F Styer. *Quantum mechanics and path integrals*. Courier Corporation, 2010.

[5] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1147–1156, 2011.

[6] JK Ochab and Zdzisław Burda. Maximal entropy random walk in community detection. *The European Physical Journal Special Topics*, 216(1):73–81, 2013.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[8] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019.

[9] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence*, pages 841–851. PMLR, 2020.

[10] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.