Project Report: Application of Path Integral based Graph Neural Networks and GraphNorm on Molecular Graph Classification

Jong Ha Lee
Stanford University
jongha98@stanford.edu

Hugo Vergnes
Stanford University
hvergnes@stanford.edu

Abstract

In this report we detail our implementation of a novel Path Integral Based Graph Neural Network (PAN) [4] and GraphNorm [2] for graph classification on the molecules dataset `ogb-molhiv` [3]. PAN algorithm leverages analogies between information transfer in quantum mechanics and message passing in Graph Neural Networks to improve expressiveness. However, we see that vanilla PAN is prone to over-fitting, an issue we tackled through GraphNorm. We achieve state of the art performance with significantly lower number of parameters than models on the OGB leaderboard. Our code is publicly available on: https://github.com/hugovergnes/drug-PAN

This algorithm wasn't present in the original leaderboard.

1 Dataset

1.1 General Statistics

The `ogbg-molhiv` dataset contains 41,127 undirected graphs, each representing a molecule where nodes are atoms and edges are chemical bonds [3]. Example molecule graphs are listed in Figure 1 - representing molecules inhibiting HIV replication, and Figure 2 - representing molecules which do not inhibit HIV replication. Each node has a 9-dimensional feature vector representing features about the atom (i.e. atom number, chirality), and each edge has a 3-dimensional feature vector representing features about the chemical bonds (i.e. bond type, bond stereo). To the best of our knowledge, **PAN's** algorithm wasn't implemented on the OGB dataset.

1.2 Prediction Task

The task at hand is graph classification, where we aim to predict whether the molecule (as represented by a graph) inhibits HIV virus replication, represented as a binary label. The dataset is heavily skewed with only 3.5% of molecules that inhibit HIV virus replication (positive label). The evaluation metric required by OGB is ROC-AUC, which we will primarily use.

Finally, we note that the dataset is split based on a *scaffold splitting* procedure which splits molecules into structurally different subsets to better replicate real experimental settings.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

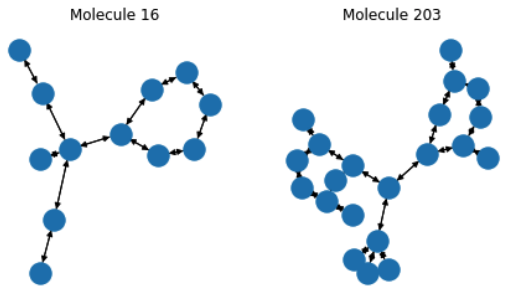

Figure 1: Sample molecules inhibiting HIV replication (positive label)

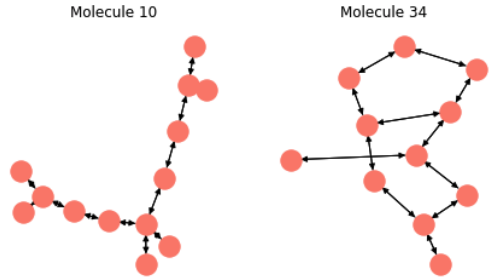

Figure 2: Sample molecules not inhibiting HIV replication (negative label)

2 Details of PANs

2.1 Motivation of PANs

Our motivation using PANs comes from the fact that PANs are derived from fundamental theories in physics. Richard Feynman introduced a generalization of the action principle in classical mechanics. It replaces the notion of a unique trajectory and computes a quantum amplitude in any given point of the space by summing the achievable possible trajectories of the wave function. [5].

In connection to Graph Neural Networks, Feynman's path integral formulations computing the probabilistic amplitude influenced by all connected paths is essentially a convolution on contributions from all possible paths in the space [4]. In other words, all possible trajectories of the information contributes to the information available in at a certain point. The discrete version of the path integral formulation defines our convolution layer of the GNN.

Empirically, the PAN original paper demonstrates high efficacy in biology and chemistry datasets, with significant improvements over state-of-the-art algorithms in graph classification on proteins graph dataset PROTEINS [1]; chemical compounds dataset on non-small cell lung cancer and ovariant cancer cell lines NCI1 [7]; and molecular compounds dataset on activities against HIV AIDS [6]. Thus, this efficacy naturally leads to an application of the PAN algorithm to our `ogb-molhiv` dataset, which should see similar improvements, while taking into account that the `ogb-molhiv` dataset is imbalanced.

2.2 PAN Convolution

In quantum mechanics, the information is propagated by a wave function over the entire space. The result observed at a given point is the summation of all possible trajectories that this wave function took. The PAN paper uses this analogy to infer a new updating function for the GNN convolution layer:

$$\phi(x,t) = \frac{1}{Z} \int \phi_0(x(t)) \int D(x) e^{iS(x,\hat{x})} \tag{1}$$

This is approximated by:

$$\phi_i = \frac{1}{Z_i} \sum_{j=1}^{N} \phi_j \sum_{\{\mathbf{l}|l_0=i, l_{|\mathbf{l}|}=j\}} e^{-\frac{E[\mathbf{l}]}{T}}, \tag{2}$$

In the analogy above, S denotes the Euclidean action. This second integral can then be interpreted as a weight. Indeed each node at time step t in node u will contribute to the information held by the node v at the time t+1 according to that weight. Analogously in a GNN we can approximate this using the Path Integral; the contribution of each node is more important if the node has many possible paths with the target node. The weight of each path (the exponential term) is learned, and it depends solely on the length of the path.

Finally after running through the algorithm, we have the following matrix formulation with Z being a diagonal matrix serving as a normalization coefficient and M being the propagation matrix, also known as the Maximum Entropy Transition (MET) matrix.

$$M = Z^{-1} \sum_{n=0}^{L} e^{-\frac{E(n)}{T}} A^n, \tag{3}$$

The corresponding update rule between layer h and $h+1$ is then:

$$X^{(h+1)} = M^{(h)} X^{(h)} W^{(h)}, \tag{4}$$

2.3 PAN Pooling

The MET matrix developed from the convolution layer can also conveniently be used for the pooling layer. Specifically, the diagonal elements of the MET matrix is a generalized version of the sub-graph centrality with different, learnable weights. Note that mathematically, sub-graph centrality can be written as $\sum_{n=0}^{\infty}(A^k)_{ii}/k!$ for node i, and thus the weight for each path length is simply $1/k!$. In contrast, the MET matrix truncates the path length at L, and the weights based on the differing path lengths (and therefore information as represented by $E(n)$) are learnable. As a result, the diagonal elements of the MET matrix M_{ii} essentially measures a sense of importance for node i. This generalized, data-driven version of sub-graph centrality captures the local graph structures to measure importance of a given node i. Finally, in combination with a node's global importance as represented by the a projection of the input signal X, a "pooling score" is determined which is ranked to choose the top K nodes with the highest "scores" to pool over. This is known as **Hybrid PANPool**, which balances both global and local importance. The equation to compute the Pooling score is the following:

$$score = \alpha Xp + \beta diag(M)$$

α, β and p are learned parameters. These coefficients allow us to compute an average mean between the projection of the node features and the sub-graph centrality. In practice, α and β are great ways to monitor over-fitting.

2.4 GraphNorm

To reduce over-fitting in our algorithm we chose to implement a Graph Normalization layer, using GraphNorm [2]. Empirically, normalization on graph is challenging because it is very task-dependent. it proposes a learnable shift in the equation:

$$norm(\hat{h}_{i,j}) = \gamma_j \frac{\hat{h}_{i,j} - \alpha_j \mu_j}{\hat{\sigma}_j} + \beta_j \tag{5}$$

$i \in [1, n]$ is a node index and $j \in [1, d]$ is a node feature index. Here μ is the mean of the hidden state and $sigma$ is its standard deviation. $\gamma_j and \beta_j$ are standard normalization parameters learned during

training. The novelty is in the parameter α_j. It is essentially a feature importance vector allowing the algorithm to learn how much each feature should be kept in the mean. Empirically, we see that GraphNorm successfully reduced Heavy Batch Noise (which was significant in our dataset since it was imbalanced) and retains information by not degrading the performance.

3 Ablation Study

3.1 Initial Approach

We mainly used OGB to split the dataset using scaffolding splitting, which is relevant for the molecular dataset. Our initial approach consisted of using the best model described in the original PAN paper while keeping the training time under two hours on a CPU. The original PAN paper demonstrated that the weighted average PAN pooling provided the best result on average.

We initially implemented 3 PANConv layers with a weighted average pooling layer (as indicated in the original paper) and dropout before finishing with a fully connected layer to extract the encoded information. The initial performance of this Vanilla model was poor with only a ROC-AUC score of 64% on the test set. Upon closer inspection of the model, the ROC-AUC score of the training set converged at only 62%, mainly due to the pooling operation where at least one of the learned parameters converged to zero (usually the weight α attributed to the matrix X). This was clearly a sign of over-fitting and that the pooling operation reduced our model to only the eigen-decomposition of the matrix M. Training results are shown in Figure 3.

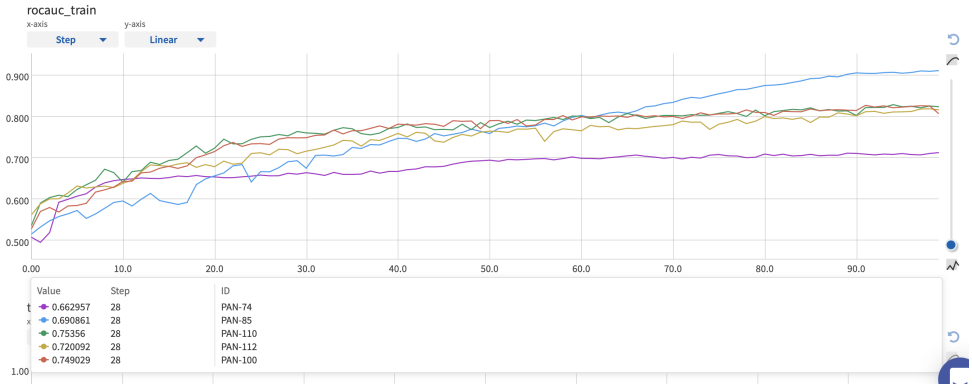

Figure 3: ROC-AUC on train dataset. The One Policy Scheduler clearly leads to over-fitting in blue, while removing GraphNorm clearly leads to under-fitting in purple. The removal of the dropout layer reduces the speed of convergence, shown in yellow.

In order to reduce overfitting, we implemented GraphNorm in Pytorch which demonstrated significant performance improvement. Additionally, the AtomEncoder which encoded the atomic information on each node allowed our first convolution layer to focus specifically on the message passing step instead of attempting to learn weights for both encoding of the node features as well as message passing. This allowed us to increase performance while reducing the number of epochs by 20% Summary of improved results are shown in Figure 4.

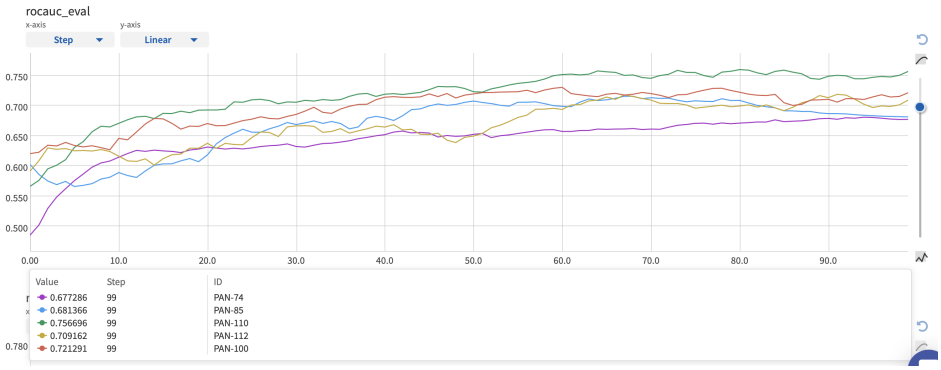

Figure 4: Results of our ablation study, characterized by ROC-AUC on the validation set. **Green** indicates our final model. **Blue** indicates a model with a One Policy Scheduler to test learning rates. **Purple** indicates a model with no GraphNorm between layers. **Yellow** indicates a model without dropout. **Red** indicates a model with a wider receptive field in the second conv layer.

3.2 Hyperparameter Tuning

Overall, more than 40 experiments have been recorded in Neptune logger. Best results are shown in Figure 5. They were designed in an ablation study fashion to zero-in on the effectiveness of each parameters. Most notably we concluded that:

- Our algorithm is learning-rate dependant. Adding a scheduler, will probably lead to over-fitting.
- The maximum size of the path taken into consideration (receptive field) has a tremendous effect on the outcome of the result. As in the original paper, we believe this parameter is really task dependent but can be reduced with the size of the graph after pooling operations.
- We noted that in the original implementation of the PAN paper, no activation functions were added between convolution layers. Adding them proved to reduce expressiveness in some scenarios.

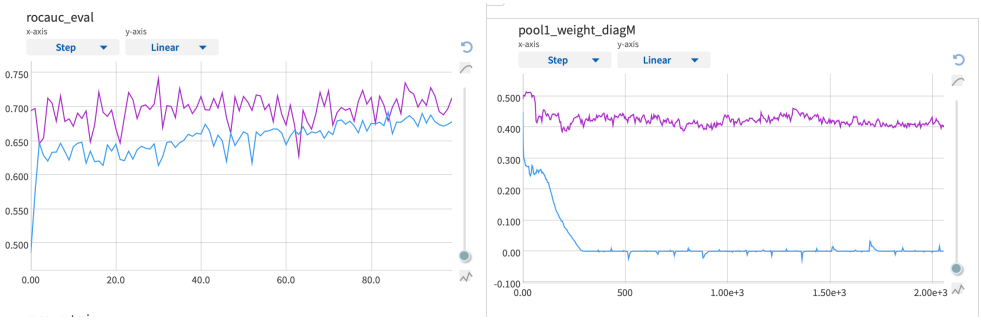

Figure 5: ROC-AUC results and PANPool layer weights on the best performing model.

4 Code and Performance

We chose to implement our model using the Pytorch-geometric module dedicated to Graph Deep Learning. Additionally, we chose to implement it again in Pytorch Lightning to use Neptune logger to track experiments more precisely. Additionally, we implemented GraphNorm from scratch in Pytorch. All of our code is available in our GitHub repo: https://github.com/hugovergnes/drug-PAN

Concerning performance, we achieved state-of-the-art with ROC-AUC score on the test set of 75.48% for our best model. Even if this performance is still technically at the very end of the OGB leaderboard,

we note that our model was only trained on a CPU with very limited number of parameters. Our experiments contained between 19k to 27k parameters, which is significantly less than models on the leaderboard which are beyond 500k parameters. This combined with the fact that we are not leveraging edge features yet, we are confident that adding more layers (up to 5), and increasing the size of the hidden space to 300 will experience significant improvements. In comparison, a GCN+topK from the OGB official repository with all hyper-parameters kept equal (Hidden dimension, number of layers, learning rate) gives a ROC-AUC score of around 70% or 71% depending on the training seed.

5 Critique of the Algorithm and Future Work

5.1 Learned Parameters

The new maximal entropy transition matrix only requires to learn the exponential term $e^{-\frac{E(n)}{T}}$ besides the usual weights. Since we typically compute a maximum path size of 5 we only add 5 learnable weights. Perhaps a vector parametrization or even a matrix parametrization as $M = Z^{-1}\sum_{n=0}^{L} E_n A^n$ might increase the expressiveness of the algorithm significantly. However, we also note that adding a matrix parametrization of size n*n will reduce significantly reduce the scalability of the model and will basically be a generalized GIN convolution.

Another point to bring to the attention of the reader is the way the pooling scores are computed. In the original paper there is only one learned parameter in the pooling layer: the weight attributed to the diagonal of M. However, adding another parameter α serves as a multiplicative coefficient for the projection score Xp and significantly boosts performance. Indeed we allow for an additional degree of freedom and simplifies the learning. So the pooling score is computed as followed in the code $score = \alpha Xp + \beta diag(M)$. Experiment showed that running the pooling score with only the parameter β led to a small standard-deviation in the pooling score, thereby creating more noise in the nodes sampled.

5.2 Edge Features

Our current algorithm doesn't take into account edge features. Usually in GCN, edge features are added to the node features in the aggregation procedure in the following way:

$$\mathbf{x}_i' = h_{\mathbf{\Theta}}\left((1+\epsilon)\cdot \mathbf{x}_i + \sum_{j\in\mathcal{N}(i)} \mathrm{ReLU}(\mathbf{x}_j + \mathbf{e}_{j,i})\right)$$

Where e_j, i represents the edge features of the edge linking j and i. But here, the information transfer is no longer incremental and we would have to take into account edges that are several hops away from the target node. However, edges in molecular structure carry a lot of information so we expect that including edge features will help gain significant predictive capability. This task might be difficult with the current structure of the algorithm, since the assumption that the density of energy is only a function of the path length will no longer be true. Including edge features this way will require a lot of extra computation since we can no longer leverage the power of the adjacency matrix to pre-compute the MET matrix.

5.3 Numerical Stability

The pooling mechanism described above proved to perform well on our dataset, but there were some numerical stability issues. Depending on the training seed, the coefficient for the pooling score would sometimes get stuck in local minima at zero. Then the subgraph centrality component usually became dominant, reducing performance. Monitoring those weights allowed us to control this issue, but further investigation would be needed to determine the root-cause behind this numerical instability.

To submit an official prediction on the leaderboard, we would need to run 10 different training with a random seed. We may use a GPU to do that. We also noticed that the seed can effect our result with the worse results being around 73% on the ROC-AUC curve.

References

[1] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21, 2005.

[2] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training, 2021.

[3] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, , and Jure Leskovec. Strategies for pre-training graph neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

[4] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16421–16433. Curran Associates, Inc., 2020.

[5] Miscellaneous. Relation between schrödinger's equation and the path integral formulation of quantum mechanics.

[6] Kaspar Riesen and Horst Bunke. Iam: graph database repository for graph based pattern recognition and machine learning. *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR)*, pages 287–297, 2008.

[7] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.