SINGLE-LAYER HOP-PMLP: COMBINING PMLP WITH INFERENCE-PHASE HOPGNN FOR SPARSE HETEROPHILIC GRAPHS

Gabriela Kirejczyk, Alexandros Kyriakopoulos, Sachin Srivathsa Satish Kumar

University of Copenhagen, Department of Science

ABSTRACT

In this paper, we propose a novel architecture combining the strengths of Propagational-MLP (PMLP) and Hop Graph Neural Network (HopGNN) to improve performance on heterophilic, sparse datasets while trying to keep the computational efficiency of a PMLP. Our model, Hop-PMLP, incorporates a single layer of HopGNN during training to capture the initial graph structure, while fully using HopGNN's message-passing capabilities only in the inference phase. We evaluate our model on various datasets demonstrating its effectiveness in handling a variety of graph sparsities, with a substantially improved performance when working on heterophilic graphs compared to a standard GNN or PMLP. Our repository can be found in: `https://github.com/AlexandrosKyr/Hop-PMLP`

***Index Terms*—** PMLP, HopGNN, heterophilic graphs, message passing, graph neural networks

1. INTRODUCTION

Graph Neural Networks (GNNs) have recently become popular due to their success in applications such as social networks [1] or recommendation systems [2]. They rely on message-passing mechanisms that enable nodes to aggregate information from their neighbours and update their representations. However, GNNs face significant challenges with scalability and over-smoothing [3]. As the number of layers increases, node interactions become computationally intensive, and node representations may converge to indistinguishable values, reducing performance, especially in heterophilic graphs where connected nodes have dissimilar features. Recent approaches have tried to tackle these issues, including a Propagational Multi-Layer Perceptron (PMLP) [4] and HopGNN [3].

The following work describes an architectural mix of those two models. The new architecture is evaluated on heterophilic and homophilic datasets, showing the performance on both types of graphs. The results are compared to the standard versions of PMLP and GNN in terms of their accuracy and training time.

1.1. Hypothesis formulation

We hypothesize that embedding a HopGNN layer in PMLP during training and leveraging full HopGNN for inference will boost accuracy on sparse, heterophilic graphs while maintaining computational efficiency.

2. BACKGROUND, RELATED WORK AND MODEL FORMULATION

2.1. Propagational Multi-Layer Perceptron

Propagational Multi-Layer Perceptron [4] is a model that combines a simple Multi-Layer Perceptron and Graph Neural Networks. MLP is the most basic form of a feed-forward neural network composed of fully connected layers. It can pass through a graph structure but processes each node independently without taking into consideration information from any neighbouring nodes, therefore completely ignoring the graph structure of the data. GNN, on the other hand, is designed to capture the relationships between the data points. Besides the fully connected layers from the MLP structure, GNN introduces message-passing layers, thanks to which each node can aggregate information from the neighbouring nodes. The model then captures both individual node features and the graph structure, making the model update each node based on both inputs.

PMLP's structure is a combination of both MLP and GNN. During training, it follows the structure of an MLP, ignoring the graph context of the nodes. However, during the testing phase, it introduces the message-passing layers as in the GNN structure, allowing the model to aggregate node information during the inference phase.

Thanks to the lack of message passing during training, the structure of PMLP makes it a computationally efficient alternative to GNN that achieves similarly good performance on heterophilic graphs.

Eq. 1 is the equation of MLP with a series of fully connected (FF) layers, and equations 2 (MP - message passing) and 3 (FF - feed-forward operations) a general formulation of a GNN, representing the inference phase of the PMLP [4]:

$$\hat{y}_u = \psi^{(L)} \cdots \psi^{(1)}(x_u) = \psi(x_u). \tag{1}$$

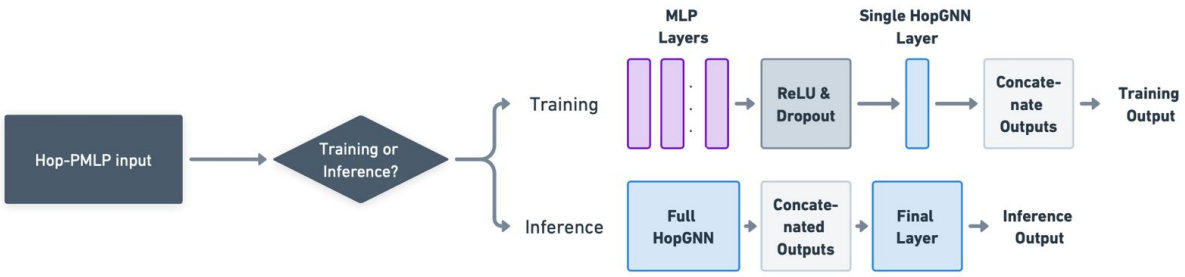


Fig. 1. Model architecture of Hop-PMLP

$$\text{(MP)}: \quad \tilde{h}_u^{(l-1)} = \sum_{v \in \mathcal{N}_u \cup \{u\}} a_G(u,v) \cdot h_v^{(l-1)}, \tag{2}$$

$$\text{(FF)}: \quad h_u^{(l)} = \psi^{(l)}\left(\tilde{h}_u^{(l-1)}\right), \tag{3}$$

2.2. HopGNN

The HopGNN framework's goal is to improve the traditional GNN in terms of scalability, efficiency, and over-smoothing. This is achieved by moving from node-to-node to hop-based relationships within each node. Instead of message-passing across nodes, HopGNN pre-computes multi-hop features, aggregating information from neighbours that are far away so that each node's embedding is improved by similar ones. The number of nodes grows exponentially with the number of hops, as it takes all the neighbours of the next nodes as well. The authors [3] define it with the following: given a graph $G = (V, E)$ with nodes N, adjacency matrix $A \in \mathbb{R}^{N \times N}$, and feature matrix $X \in \mathbb{R}^{N \times d}$, the multi-hop representation $\tilde{X}$ is formed as $\tilde{X} = [X, A^1X, \ldots, A^LX]$, where A is the normalized adjacency matrix and L is the number of hops. Each hop embedding is transformed through a linear layer f with an added learnable hop-order encoding E_{order} to produce $H = [f(X), f(A^1X), \ldots, f(A^LX)] + E_{\text{order}}$. Moreover, for non-linear interactions among hops, HopGNN utilizes a GNN-based interaction layer defined as $H^k = \text{GNN}(G_{\text{hop}}, H^{k-1}) + H^{k-1}$ for $k = 1, \ldots, K$, where G_{hop} is an internal fully connected hop-feature graph and H^k represents the processed multi-hop features at iteration k. The final representation Z is obtained via mean fusion of H^K, followed by a prediction layer $\hat{Y} = \text{softmax}(\text{linear}(Z))$, producing high scalability and improved performance across various graph datasets.

2.3. Bridging HopGNN and PMLP: Hop-PMLP

The Hop-PMLP Approach. Our model, Hop-PMLP (Fig. 1), combines the structure of Propagational MLP (PMLP) with the multi-hop message-passing abilities of HopGNN to effectively analyze sparse and heterophilic graphs. By adding a single HopGNN layer within PMLP during training, Hop-PMLP captures essential structural features. At inference, HopGNN's full message-passing layer activates, enabling multi-hop aggregation that improves robustness and accuracy. This unique hybrid structure is particularly well-suited for graphs where sparse connectivity and heterophily pose challenges to traditional GNNs, which may over-smooth or fail to leverage sufficient information diversity. Additionally, Hop-PMLP's architecture, with limited interaction in training but improved aggregation during inference, works as an adaptable model for graph data with diverse structural properties. This hybrid approach provides a targeted response to the limitations of GNNs on sparse, heterophilic data.

3. EXPERIMENTS

3.1. Implementation Details and Experimental Configurations

Initially, we experimented with a setup where HopGNN's message-passing layer was only active during the inference phase, while the model trained exclusively on the PMLP architecture. However, this approach led to poor generalization and inconsistent results, as the model was essentially trained and tested on different architectures, causing a mismatch in learned representations. Without message passing during training, the model failed to capture relational dependencies in the data that are crucial for effective inference. To address this, we introduced a single HopGNN message-passing layer during training, allowing the model to learn from node features and graph structure consistently across phases. This alignment between training and inference architectures significantly improved the model's performance.

In our implementation, we introduced the class `PMLP_HOPGNN`, which combines the functionalities of PMLP and HopGNN. The `PMLP_HOPGNN` class builds upon the structure of PMLP but includes key components of HopGNN. We implemented a customizable training and inference setup within `PMLP_HOPGNN` to allow `HopGNN`'s message-passing capabilities to be used conditionally. This enables the model to use the multi-hop attention-based features selectively, applying them only during inference if specified by a command-line flag.

Data loading, preprocessing, training loop and logging were taken from Yang et al.'s GitHub repository, while relevant code to construct the HopGNN features such as `hopgnn.py` and `graph_transform.py` were taken from Chen et al.'s GitHub repository.

We conducted a grid search on hyperparameters including learning rate, dropout rate, and hidden dimensions. The number of hops in HopGNN was adopted from Chen et al.[3] and set to 6 since they also use the same datasets to validate their results. For computational efficiency, the hyperparameter space for the search was limited to the best-performing values observed in single runs and the best values from Yang et al. [4] and Chen et al. [3], and consisted of the following:

- Learning Rate: $\{0.01, 0.05, 0.1\}$.
- Dropout Rate: $\{0.3, 0.5\}$.
- Number of Layers: $\{2, 4\}$
- Hidden Channels: $\{32, 64\}$
- Weight Decay: $\{0.00005, 0.0005\}$

3.2. Datasets

We evaluate our PMLP-HopGNN model on benchmark graph datasets, including Wisconsin, Texas, Cornell, Cora, and Pubmed. These datasets vary in heterophily, sparsity, and scale, providing a good way to validate our model's ability to handle different types of graph structures. Wisconsin, Texas, and Cornell are known for their heterophilic properties making them challenging for conventional GNNs. Meanwhile, Cora and Pubmed, though more homophilic, offer challenges like high node count and sparse connectivity, allowing us to assess the adaptability of our model.

	Wisconsin	**Texas**	**Cornell**	**Cora**	**Pubmed**
Nodes	251	183	183	2,708	19,717
Edges	515	325	293	10,556	88,651
Heterophily	High	High	High	Low	Low
Classes	5	5	5	7	3

Table 1. Dataset characteristics.

3.3. Results and Analysis

The results, enclosed in Tab. 2, show the accuracy and computational time of the Hop-PMLP, as well as the PMLP and traditional GNN as the performance benchmarks. Hop-PMLP consistently achieves the highest accuracy across all datasets, particularly outperforming by a significant margin GNN and PMLP on heterophilic graphs like Wisconsin, Texas and Cornell. On the Wisconsin dataset, the difference is the biggest, and Hop-PMLP reaches an accuracy of 81.96% compared to PMLP's 59.61% and GNN's 59.22%. The heterophily of the graph, characterized by neighbouring nodes having dissimilar features, emphasizes the architecture's ability to capture distant, non-local relationships through multi-hop aggregations. This approach gives the model a broader context and enhances its ability to make accurate predictions for individual heterophilic nodes.

In terms of the computational time, Hop-PMLP incurs a noticeable trade-off. Its enhanced accuracy comes with higher running times across all datasets due to the additional processing involved with its message-passing layer in training and multi-hop aggregations during inference. For instance, on Wisconsin, Hop-PMLP takes 33 seconds, compared to PMLP's 8 seconds and GNN's 10 seconds. This trend is consistent across datasets, with Hop-PMLP running 3 to 5 times slower than PMLP and GNN. It is particularly important with larger datasets like Pubmed, where it takes 150 seconds versus GNN's 38 seconds and PMLP's 25 seconds. This increased time reflects the computational cost of integrating a single HopGNN layer in training and using the full HopGNN inference phase, designed to capture complex node relationships effectively.

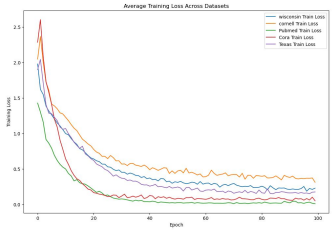
Tab. 2 also shows the accuracy results of the original HopGNN taken from Chen et al., 2023; the average time is not noted. The results show that Hop-PMLP and HopGNN perform well on heterophilic datasets such as Texas and Wisconsin thanks to their multi-hop abilities to capture non-local relationships and achieve higher accuracy. HopGNN, specifically designed for such graph properties, shows strong accuracy across all datasets, including Cornell, where Hop-PMLP also improves but less so compared to HopGNN. On more homophilic datasets like Cora and Pubmed, HopGNN's performance is notably strong, suggesting that its architec-

	Wisconsin	**Texas**	**Cornell**	**Cora**	**Pubmed**
Hop-PMLP **Av. time**	81.96 ± 6.99 33s	80.00 ± 10.57 34s	72.97 ± 5.06 23s	76.64 ± 2.31 65s	76.62 ± 0.66 150s
PMLP **Av. time**	59.61 ± 5.11 8s	63.24 ± 5.92 8s	55.14 ± 6.22 8s	73.96 ± 0.59 14s	76.00 ± 1.02 25s
GNN **Av. time**	59.22 ± 5.44 10s	59.46 ± 3.31 10s	56.22 ± 10.00 9s	74.36 ± 0.72 16s	75.80 ± 0.93 38s
HopGNN	84.96 ± 4.11	81.35 ± 4.31	83.70 ± 6.52	87.12 ± 1.35	89.98 ± 0.39

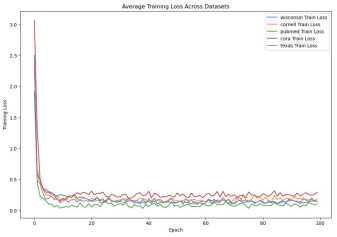
Table 2. Accuracy and average time (based on 5 runs) comparison on various datasets

ture may be more adaptable to varying levels of graph homophily. However, Hop-PMLP still performs on the same level as the original PMLP and GNN. On the other hand, while Hop-PMLP achieves competitive results, particularly on heterophilic data, it comes with increased computational time. While the average time is not shown, HopGNN is designed to also tackle the computational inefficiency of GNN [3], hence, it probably achieves better results faster, which is further described in the next section.

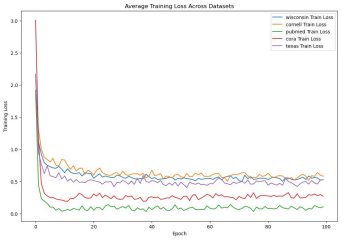
As shown in Fig. 2, Hop-PMLP converges steadily to a low loss. PMLP-GCN shows slower convergence with fluctuations, suggesting challenges in stabilizing on heterophilic graphs. The baseline GNN reduces loss quickly but stops at a higher loss, suggesting it may struggle to capture intricate relationships. The losses showcase Hop-PMLP's generalization potential, although it comes with some computational overhead.



(a) Avg. Training loss of Hop-PMLP



(b) Avg. Training loss of PMLP



(c) Avg. Training loss of GNN

Fig. 2. Average training loss across all runs for each model.

4. LIMITATIONS

While our Hop-PMLP model shows promising results on heterophilic, sparse graphs, there are a few limitations to consider. First, adding the HopGNN message-passing layer improves performance but introduces a higher computational cost. This trade-off suggests that future work could look into ways to optimize the model to maintain its performance boost without as much overhead.

Lastly, in our implementation of a Hop-PMLP, a crucial difference in how the nodes capture relationships from neighbours further away is that we don't pre-compute multi-hop node information in the model's preprocessing. HopGNN aggregates multi-hop information from the node's original feature vectors without requiring learnable parameters and is only computed once. In our implementation, hyperparameter tuning is important for the model to become expressive. Therefore this inter-node computation from multiple hops away takes place multiple times in the model, which adds a computational overhead. We did not directly compare our running time results to the HopGNN's setup, and although our approach seems to provide accuracy gains versus PMLP, it is difficult to assess whether there are any efficiency gains over HopGNN. We expect that HopGNN is more lightweight because of its preprocessing. Implementing a one-time computation of multi-hop neighbour relationships could potentially be a step which provides efficiency gains over both PMLP and HopGNN and substantially improve Hop-PMLP.

5. CONCLUSION

In this paper, we introduce Hop-PMLP, a hybrid model combining the efficiency of Propagational Multi-Layer Perceptron with the multi-hop capabilities of the Hop Graph Neural Network. Our Hop-PMLP model demonstrates substantial performance increases on heterophilic, sparse graphs, particularly in terms of accuracy gains over traditional GNNs and PMLP. By integrating a single HopGNN message-passing layer during training and fully utilizing HopGNN in inference, the model effectively captures complex relationships within graph structures. However, this approach comes with notable computational overhead, which partially offsets the efficiency benefits typically associated with PMLP. This makes only the first part of our hypothesis proven, as it did improve accuracy on heterophilic graphs but with increased computing time, making it disprove the initial assumption. Future work could focus on optimizing this hybrid model by exploring methods to reduce computational demands, such as one-time preprocessing for multi-hop neighbour relationships, without the need for learnable parameters. Such enhancements may allow the model to retain its performance boost while also achieving improved efficiency, making it more scalable and applicable to larger graph datasets.

6. REFERENCES

[1] J. Sun, Q. Jiang, and C. Lu, "Recursive social behavior graph for trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 657–666.

[2] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *arXiv preprint arXiv:2002.02126*. arXiv, 2020.

[3] J. Chen, Z. Li, Y. Zhu, J. Zhang, and J. Pu, "From node interaction to hop interaction: New effective and scalable graph learning paradigm," in *arXiv preprint arXiv:2211.11761*. arXiv, 2023.

[4] C. Yang, Q. Wu, J. Wang, and J. Yan, "Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps," in *arXiv preprint arXiv:2212.09034*. arXiv, 2023.