Graph Neural Networks are Inherently Good Generalizers: Insights by Bridging GNNs and MLPs

Chenxiao Yang, Qitian Wu, Jiahua Wang, Junchi Yan
Shanghai Jiao Tong University

ICLR

Background and Model Formulation

- MLP v.s. GNN : Architectural Connection

GNN formulation in a general form that explicitly disentangles each layer into two operations, namely a *Message-Passing **(MP)*** operation and then a *Feed-Forwarding **(FF)*** operation. After removing all the MP operations, GNN models become an MLP with a series of FF layers.

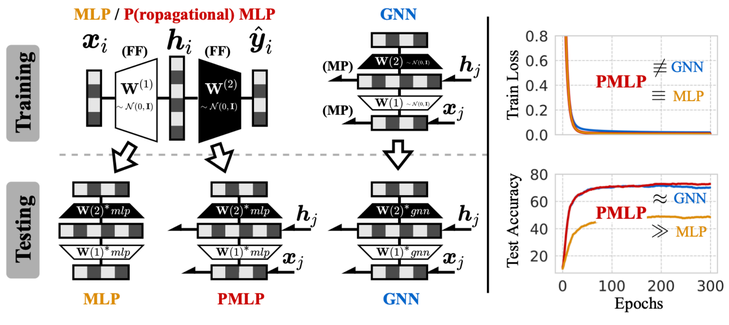
GNN **(MP):** $\tilde{\mathbf{h}}_u^{(l-1)} = \sum_{v \in \mathcal{N}_u \cup \{u\}} a_G(u,v) \cdot \mathbf{h}_u^{(l-1)}$, **(FF):** $\mathbf{h}_u^{(l)} = \psi^{(l)}\left(\tilde{\mathbf{h}}_u^{(l-1)}\right)$

MLP **(MP):** $\tilde{\mathbf{h}}_u^{(l-1)} = \sum_{v \in \mathcal{N}_u \cup \{u\}} a_G(u,v) \cdot \mathbf{h}_u^{(l-1)}$, **(FF):** $\mathbf{h}_u^{(l)} = \psi^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$

- MLP - (?) - GNN : Propagational MLP (PMLP)

PMLP is an intermediate class of models between MLP and GNN. It adopts the MLP architecture during training, and then uses the GNN architecture for inference.

(PMLP = MLP with inference-time **MP** = GNN w/o training-time **MP**)



- Instantiations of PMLP

The instantiation of PMLP *depends on the architecture of its GNN counterpart* and allows the training-time architecture to be other models (such as MLP + residual connection). See examples in the table below.

PMLP can be implemented in many different ways with a lot of flexibility, and the simplest version *only requires one line of code*. Check codes with quick guide by scanning the following QR code or entering https://github.com/chr26195/PMLP

(Codes with Quick Guide)

Model	Train and Valid	Inference
MLP	MLP: $\hat{y}_u = \psi(\mathbf{x}_u)$	MLP
PMLP_{GCN}		GCN: $\psi^{(l)}(\text{MP}(\{\mathbf{h}_v^{(l-1)}\}_{v \in \mathcal{N}_u \cup \{u\}}))$
PMLP_{SGC}		SGC: $\psi(\text{Multi-MP}(\{\mathbf{x}_v\}_{v \in \mathcal{V}}))$
PMLP_{APPNP}		APPNP: $\text{Multi-MP}(\psi(\{\mathbf{x}_v\}_{v \in \mathcal{V}}))$
PMLP_{GCNII}	ResNet	GCNII
PMLP_{JKNet}	MLP+JK	JKNet

Empirical Evaluation

We compare PMLP with MLP and GNN in inductive node classification tasks.

	Dataset #Nodes	Cora 2,708	Citeseer 3,327	Pubmed 19,717	A-Photo 7,650	A-Computer 13,752	Coauthor-CS 18,333	Coauthor-Physics 34,493
GNNs	GCN	74.82 ± 1.09	67.60 ± 0.96	76.56 ± 0.85	89.69 ± 0.87	78.79 ± 1.62	91.79 ± 0.35	91.22 ± 0.18
	SGC	73.96 ± 0.59	67.34 ± 0.54	76.00 ± 0.59	83.42 ± 2.47	77.10 ± 2.54	91.24 ± 0.59	89.18 ± 0.46
	APPNP	75.02 ± 2.17	66.58 ± 0.77	76.48 ± 0.49	89.51 ± 0.86	78.29 ± 0.55	91.64 ± 0.34	91.80 ± 0.77
MLPs	MLP	55.30 ± 0.58	56.20 ± 1.27	70.76 ± 0.78	75.61 ± 0.63	63.07 ± 1.67	87.51 ± 0.51	85.09 ± 4.11
	PMLP_{GCN}	75.86 ± 0.93	68.00 ± 0.70	76.06 ± 0.55	89.10 ± 0.88	78.05 ± 1.21	91.76 ± 0.27	91.35 ± 0.82
	Δ_{GNN}	+1.39%	+0.59%	−0.65%	−0.66%	−0.94%	−0.03%	+0.14%
	Δ_{MLP}	+37.18%	+21.00%	+7.49%	+17.84%	+23.75%	+4.86%	+7.36%
	PMLP_{SGC}	75.04 ± 0.95	67.66 ± 0.64	76.02 ± 0.57	86.50 ± 1.40	74.72 ± 3.86	91.09 ± 0.50	89.34 ± 1.40
	Δ_{GNN}	+1.46%	+0.48%	+0.03%	+3.69%	−3.09%	−0.16%	+0.18%
	Δ_{MLP}	+35.70%	+20.39%	+7.43%	+14.40%	+18.47%	+4.09%	+4.99%
	PMLP_{APP}	75.84 ± 1.36	67.52 ± 0.82	76.30 ± 1.44	88.47 ± 1.64	78.07 ± 2.10	91.64 ± 0.46	91.96 ± 0.51
	Δ_{GNN}	+1.09%	+1.41%	−0.24%	−1.16%	−0.28%	+0.00%	+0.17%
	Δ_{MLP}	+37.14%	+20.14%	+7.83%	+17.01%	+23.78%	+4.72%	+8.07%

- Phenomenon 1: PMLP significantly outperforms MLP

PMLP shares the same trained weights with a vanilla MLP, but generalizes better and thereby outperforms MLP by a large margin. This observation suggests that :

message passing layers in GNNs inherently improve model's generalization capability for handling unseen samples.

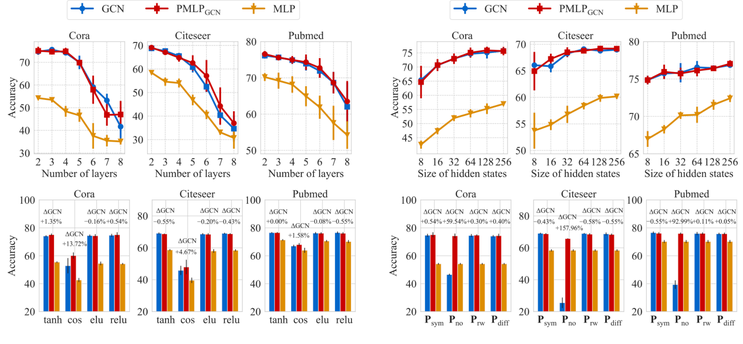
- Phenomenon 2: PMLP performs on par with GNNs.

PMLP achieves close testing performance to its GNN counterpart in *inductive node classification tasks*, and can even outperform GNN by a large margin in some cases. This observation suggests that

the major source of GNNs' success in node classification stems from their inherent generalization capability.

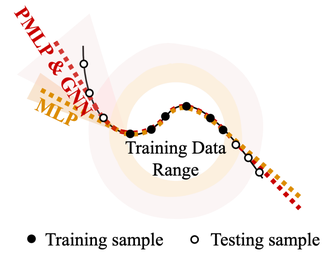
- Additional Discussions

We have also discussed: ① model depth, ② model width, ③ FF implementation, ④ MP implementation, ⑤ graph sparsity, ⑥ noisy structure, ⑦ over-smoothing, ⑧ residual connection, ⑨ heterophily …



Theoretical Analysis

Q: Why GNNs are Inherently Good Generalizers?



- Insight (informal)

Due to their architectures used in inference, both GNN and PMLP can better extrapolate out-of-distribution testing nodes for node-level tasks.

- NTK perspective on MLP, PMLP and GNN

NTK perspective allows us to conveniently study the inherent effects of model architectures due to disentanglement of weights and kernel feature map. From this perspective, *PMLP corresponds to transforming the kernel feature map of MLP to that of GNN, while fixing trained MLP weights*.

$$f_{mlp}(\mathbf{x}) = \mathbf{w}_{mlp}^{*\top}\, \phi_{mlp}(\mathbf{x}) \qquad f_{pmlp}(\mathbf{x}) = \mathbf{w}_{mlp}^{*\top}\, \phi_{gnn}(\mathbf{x}) \qquad f_{gnn}(\mathbf{x}) = \mathbf{w}_{gnn}^{*\top}\, \phi_{gnn}(\mathbf{x})$$

- Effects of Model Architectures in Extrapolation

Theorem 4. *Suppose $f_{pmlp}(\boldsymbol{x})$ is an infinitely-wide two-layer MLP with ReLU activation trained using squared loss, and adds average message passing layer before each feed-forward layer in the testing phase. For any direction $\boldsymbol{v} \in \mathbb{R}^d$ and step size $\Delta t > 0$, let $\boldsymbol{x}_0 = t\boldsymbol{v}$, and as $t \to \infty$, we have*

$$\left(f_{pmlp}(\boldsymbol{x}_0 + \Delta t \boldsymbol{v}) - f_{pmlp}(\boldsymbol{x}_0)\right)/\Delta t \quad \to \quad c_{\boldsymbol{v}} \sum_{i \in \mathcal{N}_0 \cup \{0\}} (\tilde{d} \cdot \tilde{d}_i)^{-1}. \tag{9}$$

where $c_{\boldsymbol{v}}$ is the same constant as in Theorem 3, $\tilde{d}_0 = \tilde{d}$ is the node degree (with self-connection) of $\boldsymbol{x}_0$, and $\tilde{d}_i$ is the node degree of its neighbors.

Interpretation: Similar to MLP, both PMLP and GNN (with sum/mean pooling) converge to linear functions when testing samples are far away from the training data.

Theorem 5. *Suppose all node features are normalized, and the cosine similarity of node $\boldsymbol{x}_i$ and the average of its neighbors is deonoted as $\alpha_i \in [0,1]$. Then, the convergence rate for $f_{pmlp}(\boldsymbol{x})$ is*

$$\left| \frac{\left(f_{pmlp}(\boldsymbol{x}_0 + \Delta t \boldsymbol{v}) - f_{pmlp}(\boldsymbol{x}_0)\right)/\Delta t}{c_{\boldsymbol{v}} \sum_{i \in \mathcal{N}_0 \cup \{0\}} (\tilde{d} \cdot \tilde{d}_i)^{-1}} - 1 \right| = O\left(\frac{1 + (\tilde{d}_{max} - 1)\sqrt{1 - \alpha_{min}^2}}{t} \right). \tag{10}$$

where $\alpha_{min} = \min\{\alpha_i\}_{i \in \mathcal{N}_0 \cup \{0\}} \in [0,1]$, and $\tilde{d}_{max} \geq 1$ denotes the maximum node degree in the testing node $\boldsymbol{x}_0$'s neighbors (including itself).

Interpretation: PMLP and GNN's convergence rates (to linear models) are smaller than MLP due to message passing at each layer. This indicates they are less more vulnerable to linearalization, and prone to generalize better on testing samples near the training data.