



# **MLPInit**: Embarrassingly Simple GNN Training Acceleration with MLP Initialization

Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, Neil Shah Texas A&M University, Snap Inc., Rice University



#### Graph Context Empower Graph Learning



#### GNN empowers graph learning via message passing.

#### GNNs vs. MLPs





Effectiveness (for graph)

Efficiency

Worse performance

Superior performance

Computationally efficient

Computationally cost

# GNNs are powerful for graph while MLPs are computationally efficient.

#### Begin with an Intriguing Phenomenon

MLP:  $\mathbf{H}^{l} = \sigma(\mathbf{H}^{l-1}\mathbf{W}_{mlp}^{l})$  GNN:  $\mathbf{H}^{l} = \sigma(\mathbf{A}\mathbf{H}^{l-1}\mathbf{W}_{gnn}^{l})$ 

GNN and MLP have the same trainable weight.

- If the dimensions of the hidden layers are the same
- we refer to that MLP as a PeerMLP

## What will happen if we directly adopt the weights of a converged PeerMLP to GNN?

#### Transfer Weights from MLP to GNN



	PeerMLP	GCN w/ $w_{\text{peermlp}}$	Improv.	GCN
Cora	58.50	77.60	$\uparrow 32.64\%$	82.60
CiteSeer	60.50	69.70	$\uparrow 15.20\%$	71.60
PubMed	73.60	78.10	$\uparrow 6.11\%$	79.80

Weights from a fully-trained PeerMLP make GNN performs very well.

### **Further Investigation**

PeerMLP f<sub>mlp</sub>(**X**; w<sub>mlp</sub>); GNN f<sub>gnn</sub>(**X**, **A**; w<sub>mlp</sub>)
w<sub>mlp</sub> is only trained by PeerMLP



The loss curve decreases while the accuracy curve are increas.

The GNN can be optimized by updating its PeerMLP.

## MLPInit

For a target GNN,

- 1. Construct its PeerMLP
- 2. Train PeerMLP to converge  $\rightarrow w^*_{mlp}$
- 3. Initialize GNN with  $w_{mlp}^*$
- 4. Fine tune the GNN

```
# f_gnn: graph neural network model
# f_mlp: PeerMLP of f_gnn
# Train PeerMLP for N epochs
for X, Y in dataloader_mlp:
   P = f_mlp(X)
   loss = nn.CrossEntropyLoss(P, Y)
   loss.backward()
   optimizer_mlp.step()
# Initialize GNN with MLPInit
torch.save(f_mlp.state_dict(), "w_mlp.pt")
f_gnn.load_state_dict("w_mlp.pt")
# Train GNN for n epochs
for X, A, Y in dataloader_gnn:
   P = f_{gnn}(X, A)
   loss = nn.CrossEntropyLoss(P, Y)
   loss.backward()
   optimizer gnn.step()
```

Why "Embarrassingly Simple"? Construct a PeerMLP and train it.

#### At a Glance: Faster and Better



- 1. MLPInit can accelerate GNN training by providing a better initialization of GNN.
- 2. MLPInit obtain better accuracy, gain performance improvement.

#### How Fast MLPInit Accelerate GNN?

Methods	Flickr	Yelp	Reddit	Reddit2	A-products	OGB-arXiv	OGB-products	Avg.
Random(★) MLPInit (★) Improv.	45.6 39.9 1.14×	44.7 20.3 2.20×	36.0 7.3 4.93×	48.0 7.7 6.23×	48.9 40.8 1.20×	46.7 22.7 2.06×	43.0 2.9 14.83×	44.7 20.22 2.21×
Random MLPInit Improv.	31.0 14.1 2.20×	35.8 0.0	40.6 21.8 1.86×	28.3 6.1 4.64×	50.0 9.1 5.49×	48.3 19.5 2.48×	44.9 16.9 2.66×	40.51 14.58 2.77×
Random MLPInit Improv.	15.7 7.3 2.15×	40.3 18.0 2.24×	46.2 12.8 3.61×	47.0 17.0 2.76×	37.4 1.0 37.40×	42.9 10.9 3.94×	42.8 15.0 2.85×	38.9 11.7 3.32×
Random MLPInit Improv.	46.4 30.5 1.52×	44.5 23.3 1.91×	42.4 0.0	2.4 0.0	47.7 0.0	46.7 24.5 1.91×	43.8 1.3 33.69×	45.35 19.9 2.27×
	Methods Random(*) MLPInit(*) Improv. Random MLPInit Improv. Random MLPInit Improv. Random	MethodsFlickrRandom( $\bigstar$ )45.6MLPInit( $\bigstar$ )39.9Improv.1.14×Random31.0MLPInit14.1Improv.2.20×Random15.7MLPInit7.3Improv.2.15×Random46.4MLPInit30.5Improv.1.52×	Methods       Flickr       Yelp         Random()       45.6       44.7         MLPInit()       39.9       20.3         Improv.       1.14×       2.20×         Random       31.0       35.8         MLPInit       14.1       0.0         Improv.       2.20×          Random       15.7       40.3         MLPInit       7.3       18.0         Improv.       2.15×       2.24×         Random       46.4       44.5         MLPInit       30.5       23.3         Improv.       1.52×       1.91×	MethodsFlickrYelpRedditRandom( $\bigstar$ )45.644.736.0MLPInit( $\bigstar$ )39.920.37.3Improv.1.14×2.20×4.93×Random31.035.840.6MLPInit14.10.021.8Improv.2.20×1.86×Random15.740.346.2MLPInit7.318.012.8Improv.2.15×2.24×3.61×Random46.444.542.4MLPInit30.523.30.0Improv.1.52×1.91×	MethodsFlickrYelpRedditReddit2Random( $\bigstar$ )45.644.736.048.0MLPInit( $\bigstar$ )39.920.37.37.7Improv.1.14×2.20×4.93×6.23×Random31.035.840.628.3MLPInit14.10.021.86.1Improv.2.20×1.86×4.64×Random15.740.346.247.0MLPInit7.318.012.817.0Improv.2.15×2.24×3.61×2.76×Random46.444.542.42.4MLPInit30.523.30.00.0Improv.1.52×1.91×	MethodsFlickrYelpRedditReddit2A-productsRandom( $\bigstar$ )45.644.736.048.048.9MLPInit( $\bigstar$ )39.920.37.37.740.8Improv.1.14×2.20×4.93×6.23×1.20×Random31.035.840.628.350.0MLPInit14.10.021.86.19.1Improv.2.20×1.86×4.64×5.49×Random15.740.346.247.037.4MLPInit7.318.012.817.01.0Improv.2.15×2.24×3.61×2.76×37.40×Random46.444.542.42.447.7MLPInit30.523.30.00.00.0Improv.1.52×1.91×	MethodsFlickrYelpRedditReddit2A-productsOGB-arXivRandom( $\bigstar$ )45.644.736.048.048.946.7MLPInit( $\bigstar$ )39.920.37.37.740.822.7Improv.1.14×2.20×4.93×6.23×1.20×2.06×Random31.035.840.628.350.048.3MLPInit14.10.021.86.19.119.5Improv.2.20×1.86×4.64×5.49×2.48×Random15.740.346.247.037.442.9MLPInit7.318.012.817.01.010.9Improv.2.15×2.24×3.61×2.76×37.40×3.94×Random46.444.542.42.447.746.7MLPInit30.523.30.00.00.024.5Improv.1.52×1.91×1.91×	MethodsFlickrYelpRedditReddit2A-productsOGB-arXivOGB-productsRandom( $\bigstar$ )45.644.736.048.048.946.743.0MLPInit ( $\bigstar$ )39.920.37.37.740.822.72.9Improv.1.14×2.20×4.93×6.23×1.20×2.06×14.83×Random31.035.840.628.350.048.344.9MLPInit14.10.021.86.19.119.516.9Improv.2.20×1.86×4.64×5.49×2.48×2.66×Random15.740.346.247.037.442.942.8MLPInit7.318.012.817.01.010.915.0Improv.2.15×2.24×3.61×2.76×37.40×3.94×2.85×Random46.444.542.42.447.746.743.8MLPInit30.523.30.00.00.024.51.3Improv.1.52×1.91×1.91×33.69×

## MLPInit can significantly reduce the training time of GNNs.

#### How Well does MLPInit Perform?

						10		<u> </u>	
	Methods	Flickr	Yelp	Reddit	Reddit2	A-products	OGB-arXiv	OGB-products	Avg.
SAGE	Random MLPInit Improv.	$53.72{\scriptstyle\pm0.16} \\ 53.82{\scriptstyle\pm0.13} \\ \uparrow 0.19\%$	$\begin{array}{c} 63.03 {\scriptstyle \pm 0.20} \\ 63.93 {\scriptstyle \pm 0.23} \\ \uparrow 1.43\% \end{array}$	$\begin{array}{c} 96.50 {\scriptstyle \pm 0.03} \\ 96.66 {\scriptstyle \pm 0.04} \\ \uparrow 0.16\% \end{array}$	$51.76{\scriptstyle\pm2.53} \\ 89.60{\scriptstyle\pm1.60} \\ \uparrow 73.09\%$	$\begin{array}{c} 77.58 {\scriptstyle \pm 0.05} \\ 77.74 {\scriptstyle \pm 0.06} \\ \uparrow 0.21 \% \end{array}$	$\begin{array}{c} 72.00{\scriptstyle\pm0.16} \\ 72.25{\scriptstyle\pm0.30} \\ \uparrow 0.36\% \end{array}$	$\begin{array}{c} 80.05 {\scriptstyle \pm 0.35} \\ 80.04 {\scriptstyle \pm 0.62} \\ \scriptstyle \downarrow 0.01\% \end{array}$	70.66 76.29 ↑ 7.97%
SAINT	Random MLPInit Improv.	$51.37{\scriptstyle\pm 0.21}\\51.35{\scriptstyle\pm 0.10}\\{\scriptstyle\downarrow 0.05\%}$	$\begin{array}{c} 29.42{\scriptstyle\pm1.32} \\ 43.10{\scriptstyle\pm1.13} \\ \uparrow 46.47\% \end{array}$	$\begin{array}{c} 95.58 {\scriptstyle \pm 0.07} \\ 95.64 {\scriptstyle \pm 0.06} \\ \uparrow 0.06\% \end{array}$	$\begin{array}{c} 36.45 {\scriptstyle \pm 4.09} \\ 41.71 {\scriptstyle \pm 1.25} \\ \uparrow 14.45\% \end{array}$	$59.31{\scriptstyle\pm 0.12} \\ 68.24{\scriptstyle\pm 0.17} \\ \uparrow 15.06\%$	$\begin{array}{c} 67.95 {\scriptstyle \pm 0.24} \\ 68.80 {\scriptstyle \pm 0.20} \\ \uparrow 1.25\% \end{array}$	$73.80{\scriptstyle\pm 0.58} \\ 74.02{\scriptstyle\pm 0.19} \\ \uparrow 0.30\%$	59.12 63.26 ↑ 7.00%
C-GCN	Random MLPInit Improv.	$\begin{array}{c} 49.95 \pm 0.15 \\ 49.96 \pm 0.20 \\ \uparrow 0.02\% \end{array}$	$\begin{array}{c} 56.39 {\scriptstyle \pm 0.64} \\ 58.05 {\scriptstyle \pm 0.56} \\ \uparrow 2.94 \% \end{array}$	$\begin{array}{c} 95.70 {\scriptstyle \pm 0.06} \\ 96.02 {\scriptstyle \pm 0.04} \\ \uparrow 0.33 \% \end{array}$	$53.79{\scriptstyle\pm2.48} \\ 77.77{\scriptstyle\pm1.93} \\ \uparrow 44.60\%$	$52.74{\scriptstyle\pm0.28}\atop 55.61{\scriptstyle\pm0.17}\\ \uparrow 5.45\%$	$\begin{array}{c} 68.00 {\scriptstyle \pm 0.59} \\ 69.53 {\scriptstyle \pm 0.50} \\ \uparrow 2.26\% \end{array}$	$78.71{\scriptstyle\pm 0.59} \\78.48{\scriptstyle\pm 0.64} \\\downarrow 0.30\%$	65.04 69.34 ↑ 6.61%
GCN	Random MLPInit Improv.	$50.90{\scriptstyle\pm 0.12} \\ 51.16{\scriptstyle\pm 0.20} \\ \uparrow 0.51\%$	$\begin{array}{c} 40.08 \pm 0.15 \\ 40.83 \pm 0.27 \\ \uparrow 1.87\% \end{array}$	$92.78 {\scriptstyle \pm 0.11} \\ 91.40 {\scriptstyle \pm 0.20} \\ {\scriptstyle \downarrow} 1.49\%$	$\begin{array}{c} 27.87 \pm 3.45 \\ 80.37 \pm 2.61 \\ \uparrow 188.42\% \end{array}$	$\begin{array}{c} 36.35 {\scriptstyle \pm 0.15} \\ 39.70 {\scriptstyle \pm 0.11} \\ \uparrow 9.22\% \end{array}$	$70.25{\scriptstyle\pm0.22}\atop70.35{\scriptstyle\pm0.34}\\\uparrow0.14\%$	$77.08{\scriptstyle\pm 0.26} \\ 76.85{\scriptstyle\pm 0.34} \\ \downarrow 0.29\%$	$56.47 \\ 64.38 \\ \uparrow 14.00\%$

MLPInit improves the prediction performance for node classification.

### How Well does MLPInit Perform?

	Methods	AUC	AP	Hits@10	Hits@20	Hits@50	Hits@100
PubMed	MLP <sub>random</sub> GNN <sub>random</sub> GNN <sub>mlpinit</sub> Improvement	$\begin{array}{c} 94.76 \pm 0.30 \\ 96.66 \pm 0.29 \\ 97.31 \pm 0.19 \\ \uparrow 0.68\% \end{array}$	$\begin{array}{c} 94.28 \pm 0.36 \\ 96.78 \pm 0.31 \\ 97.53 \pm 0.21 \\ \uparrow 0.77\% \end{array}$	$\begin{array}{c} 14.68 {\scriptstyle \pm 2.60} \\ 28.38 {\scriptstyle \pm 6.11} \\ 37.58 {\scriptstyle \pm 7.52} \\ \uparrow 32.43 \% \end{array}$	$\begin{array}{c} 24.01 \pm 3.04 \\ 42.55 \pm 4.83 \\ 51.83 \pm 7.62 \\ \uparrow 21.80\% \end{array}$	$\begin{array}{c} 40.02 \pm 2.75 \\ 60.62 \pm 4.29 \\ 70.57 \pm 3.12 \\ \uparrow 16.42\% \end{array}$	$54.85{\scriptstyle\pm2.03} \\ 75.14{\scriptstyle\pm3.00} \\ 81.42{\scriptstyle\pm1.52} \\ \uparrow 8.36\%$
DBLP	MLP <sub>random</sub> GNN <sub>random</sub> GNN <sub>mlpinit</sub> Improvement	$\begin{array}{c} 95.20{\scriptstyle\pm 0.18}\\ 96.29{\scriptstyle\pm 0.20}\\ 96.67{\scriptstyle\pm 0.13}\\ \uparrow 0.39\% \end{array}$	$\begin{array}{c} 95.53 {\scriptstyle \pm 0.25} \\ 96.64 {\scriptstyle \pm 0.23} \\ 97.09 {\scriptstyle \pm 0.14} \\ \uparrow 0.47\% \end{array}$	$\begin{array}{c} 28.70 \pm 3.73 \\ 36.55 \pm 4.08 \\ 40.84 \pm 7.34 \\ \uparrow 11.73\% \end{array}$	$\begin{array}{c} 39.22{\scriptstyle\pm4.13} \\ 43.13{\scriptstyle\pm2.85} \\ 53.72{\scriptstyle\pm4.25} \\ \uparrow 24.57\% \end{array}$	$53.36{\scriptstyle\pm3.81}\\59.98{\scriptstyle\pm2.43}\\67.99{\scriptstyle\pm2.85}\\\uparrow13.34\%$	$\begin{array}{c} 64.83 \pm 1.95 \\ 71.57 \pm 1.00 \\ 77.76 \pm 1.20 \\ \uparrow 8.65\% \end{array}$
A-Photo	MLP <sub>random</sub> GNN <sub>random</sub> GNN <sub>mlpinit</sub> Improvement	$\begin{array}{c} 86.18 \pm 1.41 \\ 92.07 \pm 2.14 \\ 93.99 \pm 0.58 \\ \uparrow 2.08\% \end{array}$	$\begin{array}{c} 85.37 \pm 1.24 \\ 91.52 \pm 2.08 \\ 93.32 \pm 0.60 \\ \uparrow 1.97\% \end{array}$	$\begin{array}{c} 4.36 \pm 1.14 \\ 9.63 \pm 1.58 \\ 9.17 \pm 2.12 \\ \downarrow 4.75\% \end{array}$	$\begin{array}{c} 6.96 {\scriptstyle \pm 1.28} \\ 12.82 {\scriptstyle \pm 1.72} \\ 13.12 {\scriptstyle \pm 2.11} \\ \uparrow 2.28\% \end{array}$	$\begin{array}{c} 12.20{\scriptstyle\pm1.24}\\ 20.90{\scriptstyle\pm1.90}\\ 22.93{\scriptstyle\pm2.56}\\ \uparrow 9.73\% \end{array}$	$\begin{array}{c} 17.91 \pm 1.26 \\ 29.08 \pm 2.53 \\ 32.37 \pm 1.89 \\ \uparrow 11.32\% \end{array}$
Physics	MLP <sub>random</sub> GNN <sub>random</sub> GNN <sub>mlpinit</sub> Improvement	$\begin{array}{c} 96.26 \scriptstyle{\pm 0.11} \\ 95.84 \scriptstyle{\pm 0.13} \\ 96.89 \scriptstyle{\pm 0.07} \\ \uparrow 1.10\% \end{array}$	$95.63{\scriptstyle\pm 0.15} \\ 95.38{\scriptstyle\pm 0.15} \\ 96.55{\scriptstyle\pm 0.11} \\ \uparrow 1.22\%$	$5.38{\scriptstyle\pm1.32}\\6.62{\scriptstyle\pm1.00}\\8.05{\scriptstyle\pm1.44}\\\uparrow21.63\%$	$\begin{array}{c} 8.76 \pm 1.37 \\ 10.39 \pm 1.04 \\ 13.06 \pm 1.94 \\ \uparrow 25.76\% \end{array}$	$15.86 \pm 0.81 \\ 18.55 \pm 1.60 \\ 22.38 \pm 1.94 \\ \uparrow 20.63\%$	$\begin{array}{c} 24.70 \pm 1.11 \\ 26.88 \pm 1.95 \\ 32.31 \pm 1.43 \\ \uparrow 20.20\% \end{array}$
	Avg.	$\uparrow 1.05\%$	$\uparrow 1.10\%$	↑ 17.81%	↑ 20.97%	$\uparrow 14.88\%$	$\uparrow 10.46\%$

MLPInit improves the prediction performance for link prediction task.

#### Why Perform Well?

• Loss Landscape:



#### MLPInit helps find better local minima for GNNs.

### Why Perform Well?

Weight distribution



#### MLPInit produces more high-magnitude weights, indicating better optimization of GNN.





#### Thank you!

Xiaotian Han Texas A&M University <u>han@tamu.edu</u> <u>https://ahxt.github.io</u>





MLPInit: Embarrassingly Simple GNN Training Acceleration with MLP Initialization Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, Neil Shah Paper: <u>https://openreview.net/forum?id=P8YIphWNEGO</u> Code: <u>https://github.com/snap-research/MLPInit-for-GNNs</u> Slides: <u>https://ahxt.github.io/files/mlpinit\_slides.pdf</u>