# GraphNorm:

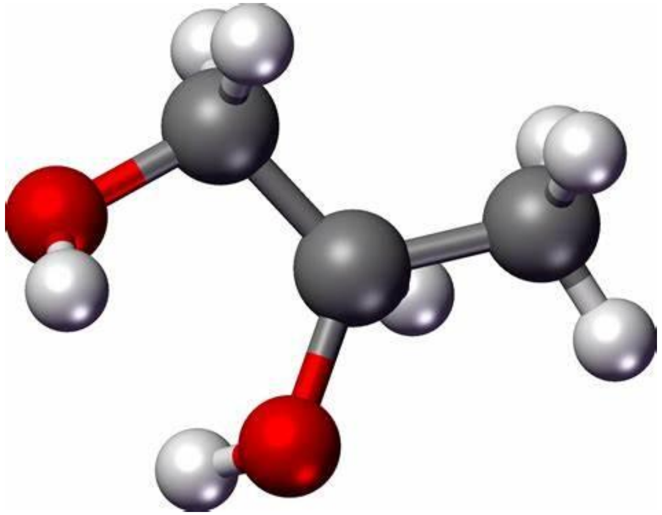
## A Principled Approach to Accelerating Graph Neural Network Training

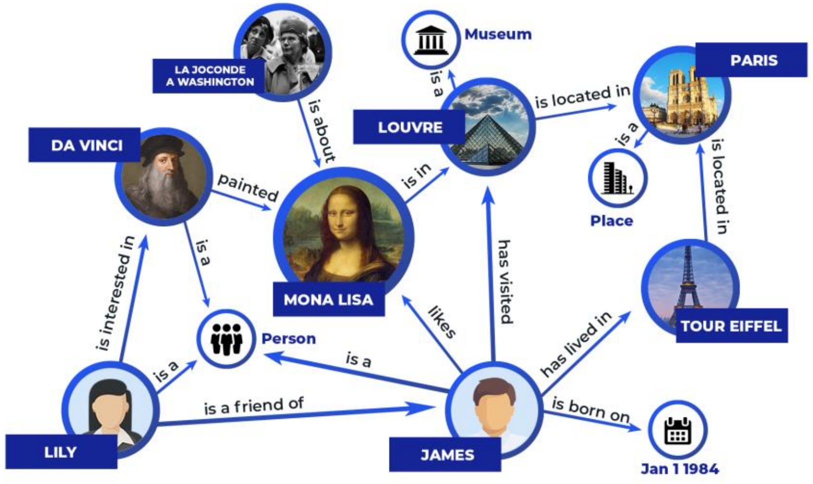
Tianle Cai*, Shengjie Luo*, Keyulu Xu, Di He, Tie-Yan Liu, Liwei Wang

https://github.com/lsj2408/GraphNorm

PRINCETON UNIVERSITY 北京大學 PEKING UNIVERSITY Massachusetts Institute of Technology Microsoft

# Learning with graph – a general form of data






**Drug Discovery**

(phys.org)

**Social Networks**

(acwits)

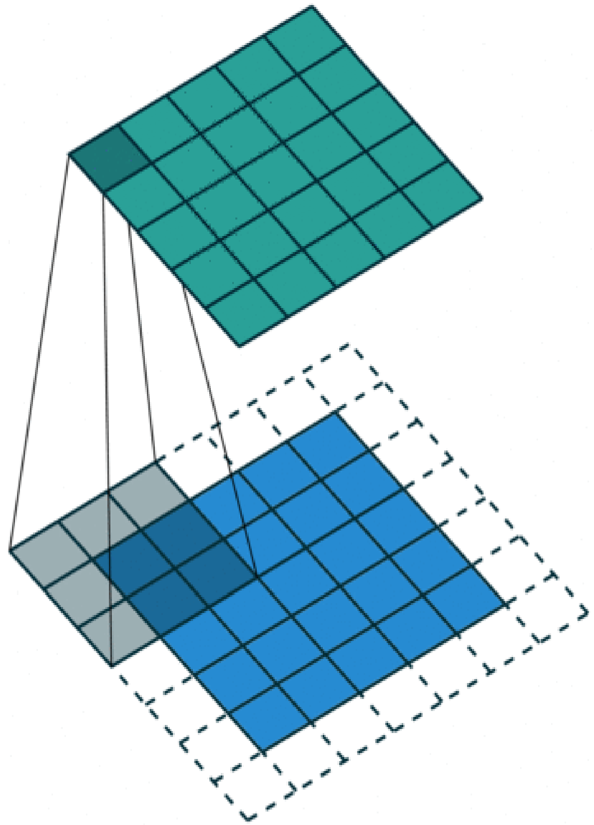
**Knowledge Graph**

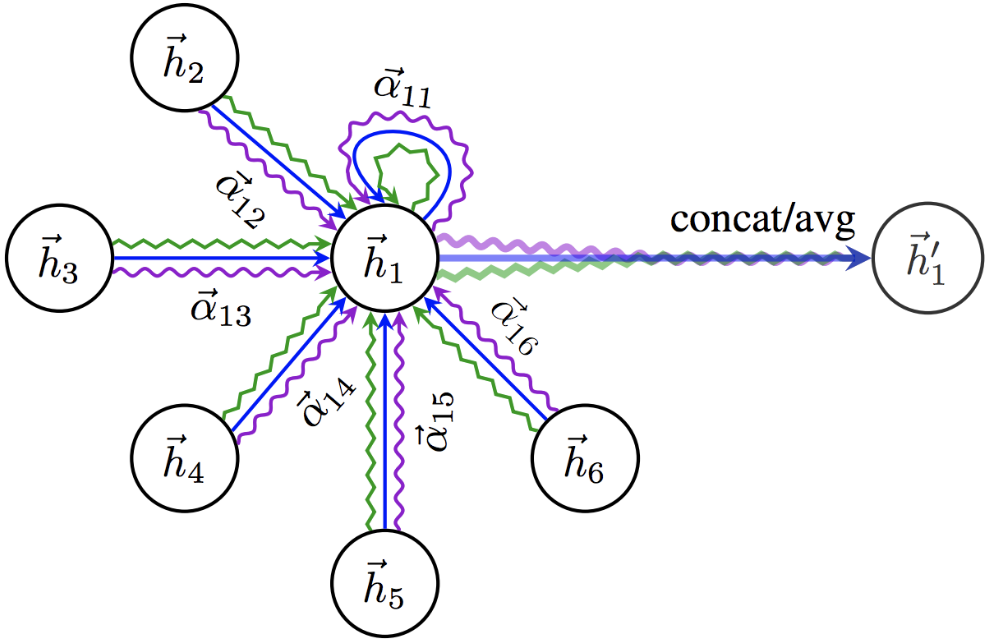
(yashuseth.blog)

# Graph Neural Networks

**Neighborhood Aggregation** a.k.a. Message Passing or Graph Convolution

Aggregate neighbor features with permutation invariance functions





# Graph Neural Networks

## Neighborhood Aggregation

$$h_i^{(k)} = AGGREGATE^{(k)}\left(h_i^{(k-1)}, \left\{h_j^{(k-1)}: v_j \in \mathcal{N}(v_i)\right\}\right)$$

*Feature of node $v_j$ in layer k-1.*

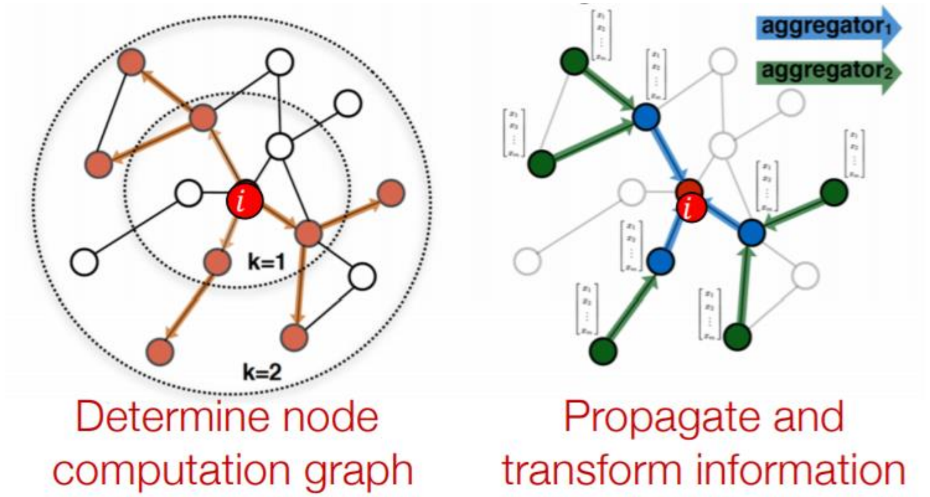
## Example -- GIN

$$h_i^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_i^{(k-1)} + \sum_{j\in\mathcal{N}(v_i)} h_j^{(k-1)}\right)$$

$\epsilon^{(k)}$ is a learnable parameter

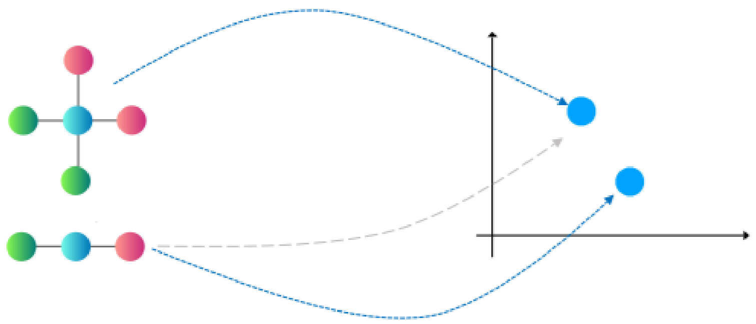
## Readout Function

$$h_G = READOUT(\{h_i^{(K)} | v_i \in V\})$$



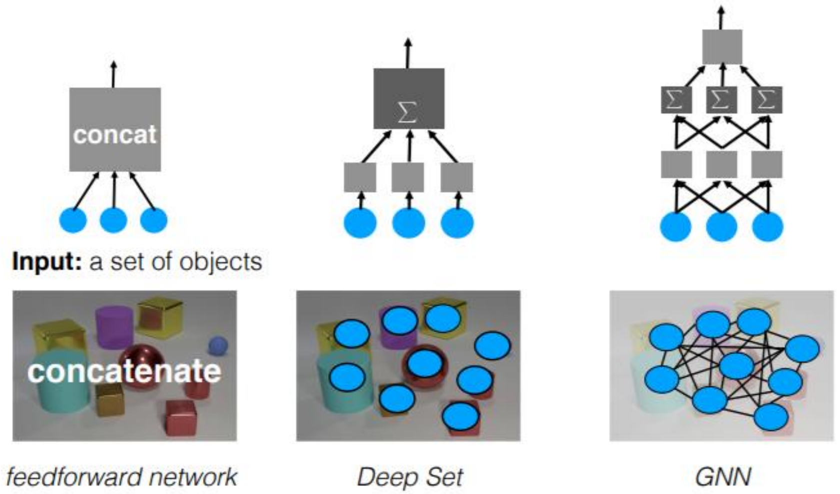
# Investigations on Graph Neural Network

***Expressive Power***



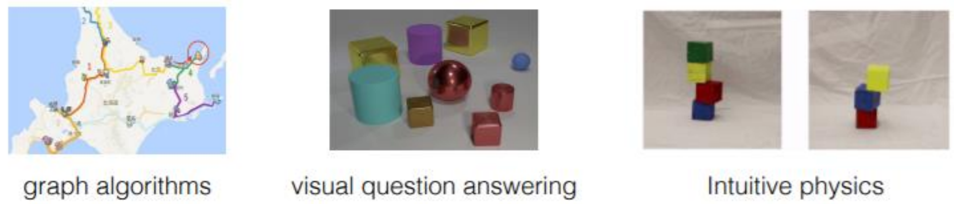
**Which graphs can a GNN distinguish?**

***Generalization***

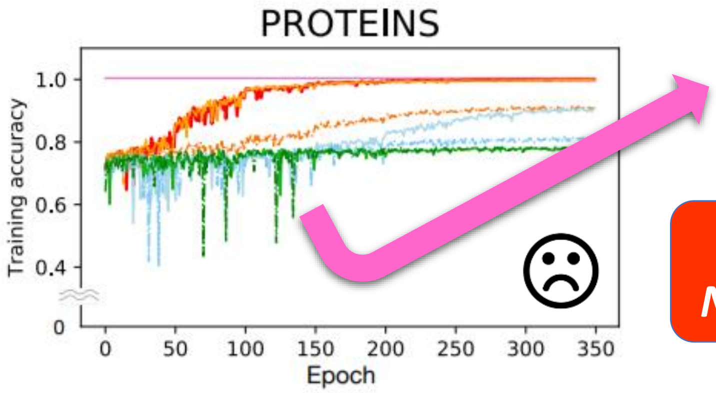



***Reasoning***

Reasoning tasks as dynamic programming (DP):



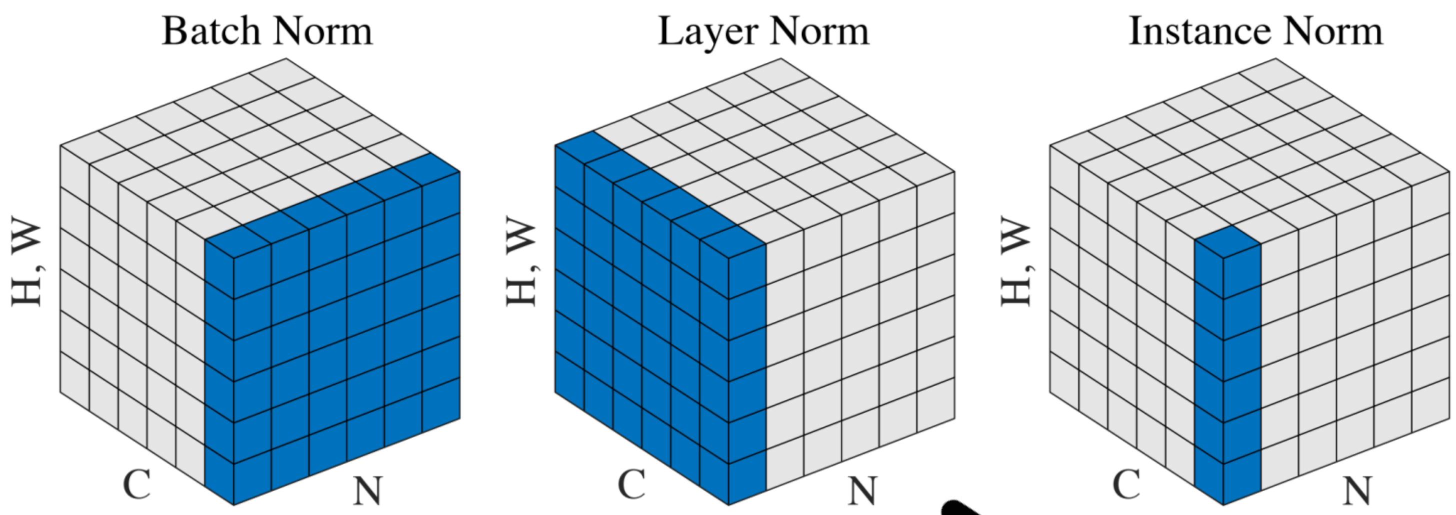

***Optimization?***

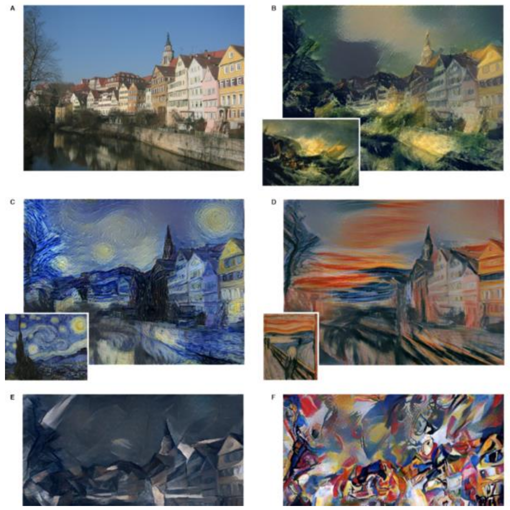



- **Training instability**
- **Slow convergence**

***Even with Normalization methods.***

# Normalization for Neural Networks



Style Transfer

Image Classification


NLP Tasks

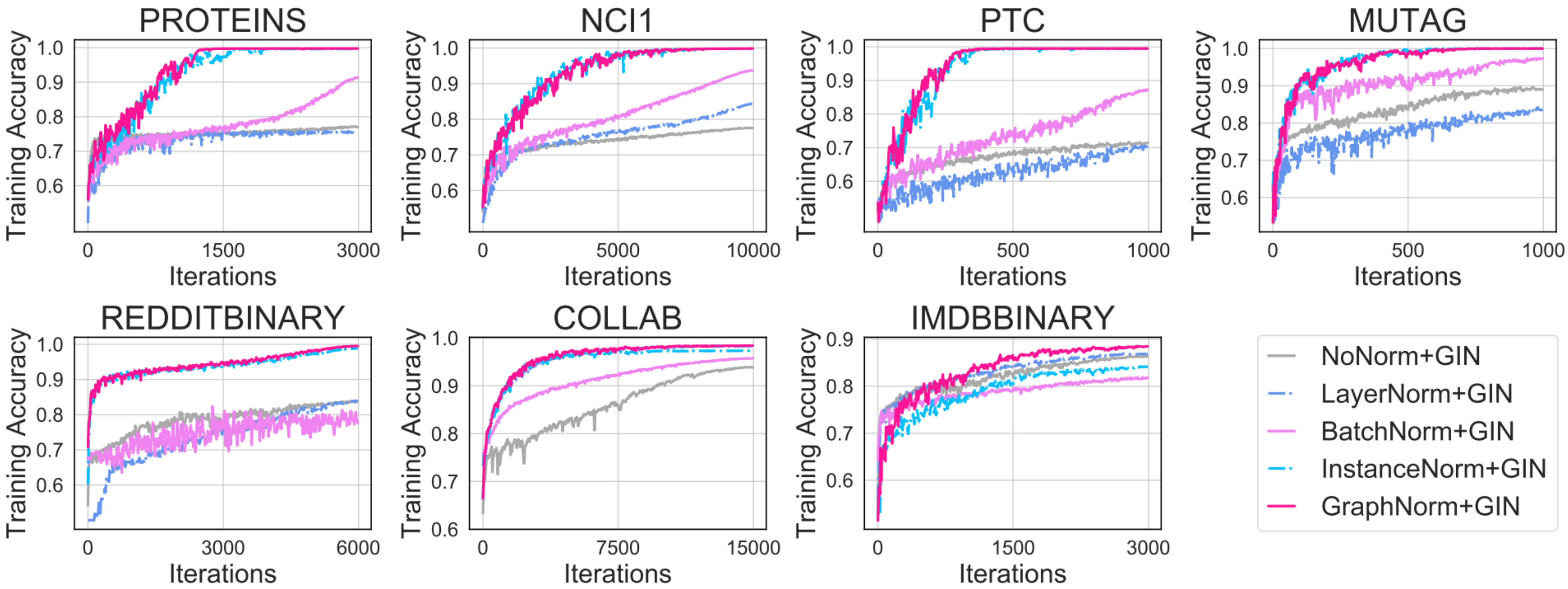
***For GNNs, BN and LN are simply adopted without further investigations.***

What normalization methods are effective for Graph Neural Networks?

# This paper

- Adapting and evaluating existing normalization methods to GNNs.
- Explaining the effectiveness of InstanceNorm over BatchNorm.
- Identifying an expressiveness degradation of InstanceNorm.
- Proposing GraphNorm which addresses the issue and converges faster.

# Evaluation of existing normalization methods



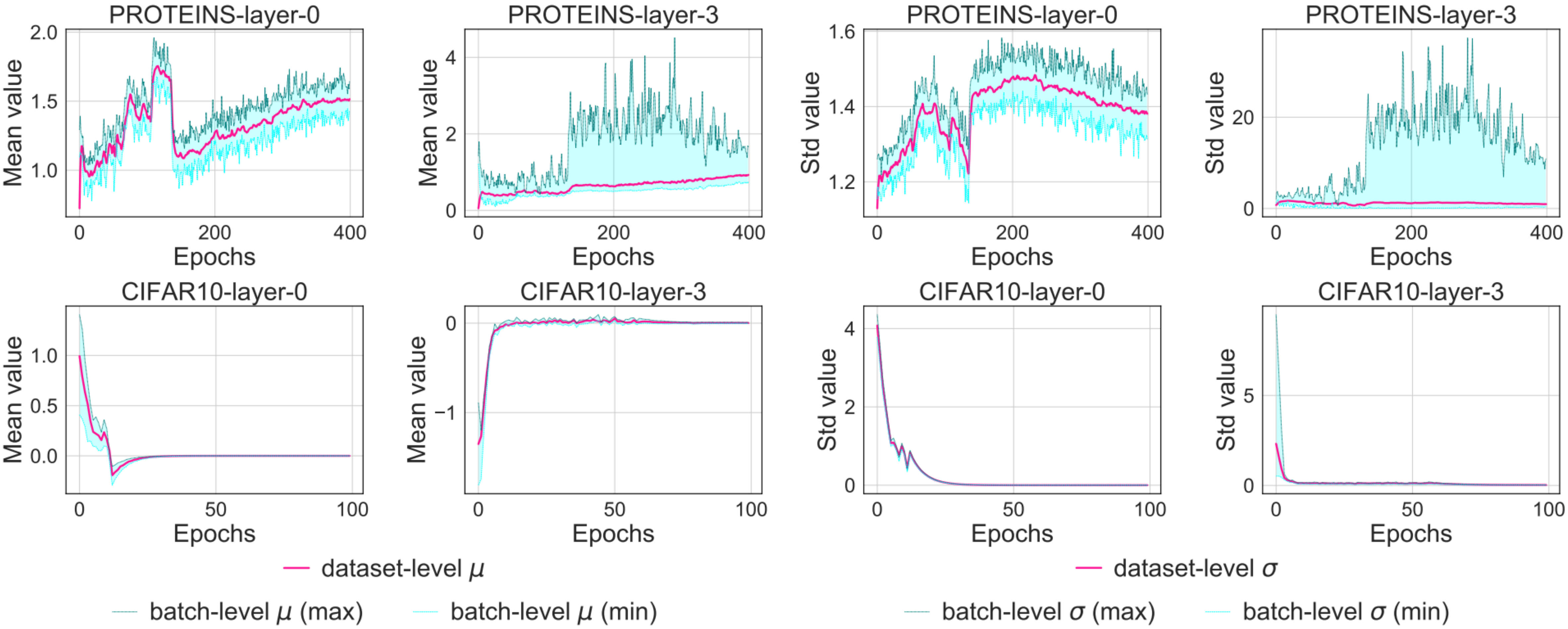
# Preconditioning effect of InstanceNorm

**Theorem 3.1** (Shift Serves as a Preconditioner of $Q$). *Let $Q, N$ be defined as in Eq. (6), $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$ be the singular values of $Q$. We have $\mu_n = 0$ is one of the singular values of $QN$, and let other singular values of $QN$ be $0 \leq \mu_1 \leq \mu_2 \leq \cdots \leq \mu_{n-1}$. Then we have*

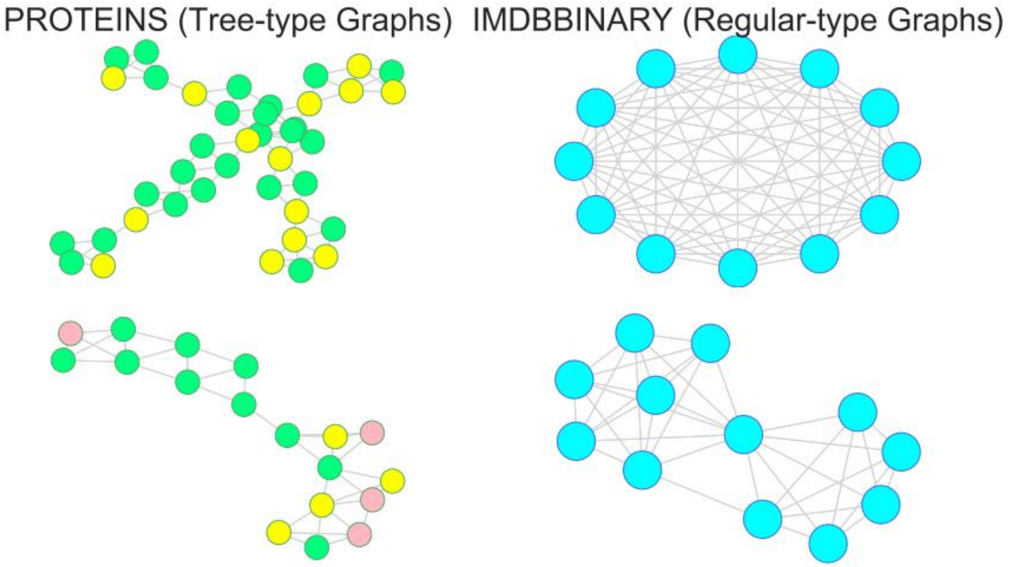
$$\lambda_1 \leq \mu_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \mu_{n-1} \leq \lambda_n, \qquad (7)$$

*where $\lambda_i = \mu_i$ or $\lambda_i = \mu_{i-1}$ only if there exists one of the right singular vectors $\alpha_i$ of $Q$ associated with $\lambda_i$ satisfying $\mathbf{1}^\top \alpha_i = 0$.*

# Heavy batch noise in graphs



# Expressiveness degradation of InstanceNorm




**Proposition 4.1.** *For a $r$-regular graph with one-hot encodings as its features described above, we have for GIN,* $\mathrm{Norm}\left(W^{(1)}H^{(0)}Q_{\mathrm{GIN}}\right) = S\left(W^{(1)}H^{(0)}Q_{\mathrm{GIN}}\right)N = 0$, *i.e., the output of normalization layer is a zero matrix without any information of the graph structure.*

**Proposition 4.2.** *For a complete graph ($r = n - 1$), we have for GIN, $Q_{\mathrm{GIN}}N = \xi^{(k)}N$, i.e., graph structural information in $Q$ will be removed after multiplying $N$.*

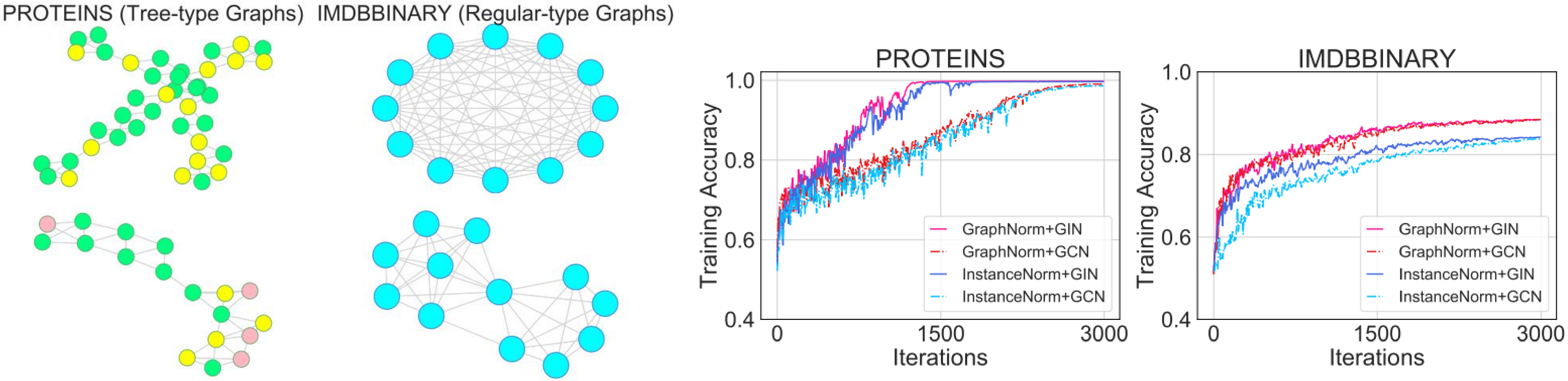
# Proposed method: **GraphNorm**

Key: learnable parameter to control how much the information we need to keep in the mean

$$\text{GraphNorm}\left(\hat{h}_{i,j}\right) = \gamma_j \cdot \frac{\hat{h}_{i,j} - \boxed{\alpha_j} \cdot \mu_j}{\hat{\sigma}_j} + \beta_j,$$

- Inheriting the merit of InstanceNorm
- Solving the expressiveness degradation problem

# GraphNorm addresses the issue of InstanceNorm



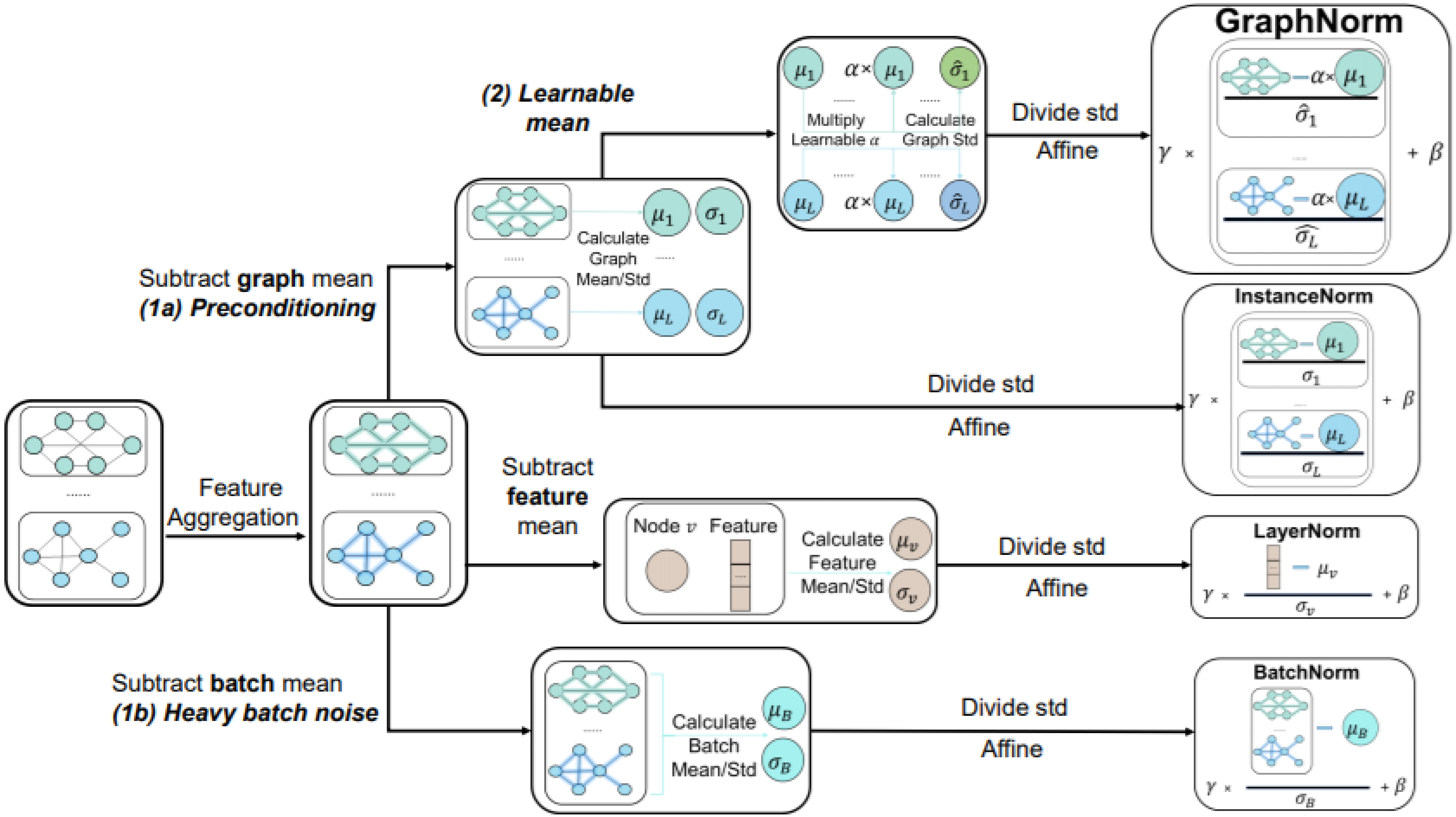
# GraphNorm achieves good performance

| Datasets | MUTAG | PTC | PROTEINS | NCI1 | IMDB-B | RDT-B | COLLAB |
|---|---|---|---|---|---|---|---|
| # graphs | 188 | 344 | 1113 | 4110 | 1000 | 2000 | 5000 |
| # classes | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Avg # nodes | 17.9 | 25.5 | 39.1 | 29.8 | 19.8 | 429.6 | 74.5 |
| WL SUBTREE (SHERVASHIDZE ET AL., 2011) | 90.4 ± 5.7 | 59.9 ± 4.3 | 75.0 ± 3.1 | **86.0 ± 1.8** | 73.8 ± 3.9 | 81.0 ± 3.1 | 78.9 ± 1.9 |
| DCNN (ATWOOD & TOWSLEY, 2016) | 67.0 | 56.6 | 61.3 | 62.6 | 49.1 | - | 52.1 |
| DGCNN (ZHANG ET AL., 2018) | 85.8 | 58.6 | 75.5 | 74.4 | 70.0 | - | 73.7 |
| AWL (IVANOV & BURNAEV, 2018) | 87.9 ± 9.8 | - | - | - | 74.5 ± 5.9 | 87.9 ± 2.5 | 73.9 ± 1.9 |
| GIN+LAYERNORM | 82.4 ± 6.4 | 62.8 ± 9.3 | 76.2 ± 3.0 | 78.3 ± 1,7 | 74.5 ± 4,4 | 82.8 ± 7.7 | 80.1 ± 0.8 |
| GIN+BATCHNORM ((XU ET AL., 2019)) | 89.4 ± 5.6 | 64.6 ± 7.0 | 76.2 ± 2.8 | 82.7 ± 1.7 | 75.1 ± 5.1 | 92.4 ± 2.5 | **80.2 ± 1.9** |
| GIN+INSTANCENORM | 90.5 ± 7.8 | 64.7 ± 5.9 | 76.5 ± 3.9 | 81.2 ± 1.8 | 74.8 ± 5.0 | 93.2 ± 1.7 | 80.0 ± 2.1 |
| **GIN+GraphNorm** | **91.6 ± 6.5** | **64.9 ± 7.5** | **77.4 ± 4.9** | 81.4 ± 2.4 | **76.0 ± 3.7** | **93.5 ± 2.1** | **80.2 ± 1.0** |

*Table 2.* **Test performance** on OGB.

| Datasets | OGBG-MOLHIV |
|---|---|
| # graphs | 41,127 |
| # classes | 2 |
| Avg # nodes | 25.5 |
| GCN (Hu et al., 2020) | 76.06 ± 0.97 |
| GIN (Hu et al., 2020) | 75.58 ± 1.40 |
| GCN+LayerNorm | 75.04 ± 0.48 |
| GCN+BatchNorm | 76.22 ± 0.95 |
| GCN+InstanceNorm | 78.18 ± 0.42 |
| **GCN+GraphNorm** | **78.30 ± 0.69** |
| GIN+LayerNorm | 74.79 ± 0.92 |
| GIN+BatchNorm | 76.61 ± 0.97 |
| GIN+InstanceNorm | 77.54 ± 1.27 |
| **GIN+GraphNorm** | **77.73 ± 1.29** |

# Thank you : )



GraphNorm
InstanceNorm
LayerNorm
BatchNorm
(2) Learnable mean
Multiply Learnable α
Calculate Graph Std
Divide std
Affine
Calculate Graph Mean/Std
Subtract graph mean
(1a) Preconditioning
Feature Aggregation
Subtract feature mean
Node v
Feature
Calculate Feature Mean/Std
Subtract batch mean
(1b) Heavy batch noise
Calculate Batch Mean/Std