Edge-Aware Graph Neural Network Baselines for Protein Function Prediction on OGBN-Proteins

Aleksandar Stanković
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
stankovic.sv25.2022@uns.ac.rs

Dejan Lisica
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
lisica.sv49.2022@uns.ac.rs

***Abstract*—We present an engineering-oriented study of edge-aware graph neural network (GNN) baselines for the OGBN-Proteins benchmark in PyTorch Geometric (PyG). The dataset models a protein–protein interaction network where edges carry 8-dimensional association evidence and each protein has 112 functional labels. We compare simple aggregation schemes that convert edge features into node inputs and study how edge information is used inside message passing. Our best-performing baseline is GraphSAGE with sum-based edge-to-node features. We then compare Batch Normalization, Layer Normalization, and a species-aware Conditional LayerNorm, and report predictive quality (ROC-AUC and micro-F1 at threshold 0.5, as well as micro-F1 after calibration and per-label thresholding) and resource usage (training time, GPU memory, and parameter count). Finally, we evaluate post-hoc calibration and light label-correlation smoothing, which improve decision quality with negligible impact on ROC-AUC. All experiments are released as reproducible PyG scripts, providing practical guidance for deploying GNNs on large edge-attributed graphs.**

***Index Terms*—graph neural networks, protein function prediction, OGBN-Proteins, bioengineering and biomedicine, PyTorch Geometric**

I. Introduction

Graph-structured data appear in many information and communication technology (ICT) applications, from telecommunication and social networks to biological interaction networks. Protein–protein interaction (PPI) graphs are a typical example in bioinformatics and bioengineering, where vertices represent proteins and edges encode different types of experimental evidence for interactions.

The OGBN-Proteins benchmark [1] models such a PPI network. Nodes are proteins, edges carry an 8-dimensional vector of association scores, and each protein has 112 binary functional labels. The official evaluation metric is the mean ROC-AUC across labels, using a species-wise split in which validation and test proteins come from species that are unseen during training.

Graph neural networks (GNNs) are a natural modelling tool for this kind of data. However, reproducing strong baselines in a specific software stack is time-consuming, and apparently minor design decisions, such as how to aggregate edge features or which normalization scheme to use, can noticeably change the accuracy-cost trade-off. For engineers who want to deploy GNN models in practical systems, it is therefore useful to have a clear, reproducible baseline implementation together with a systematic comparison of these design choices.

In this paper we provide such a study for the OGBN-Proteins benchmark in PyTorch Geometric (PyG). We focus on two practical questions that arise in many edge-attributed graphs:

- how to construct node input features from incident 8-dimensional edge features; and
- how to incorporate edge information inside the message-passing layers of a GNN.

On top of this, we compare several normalization schemes (BatchNorm, LayerNorm, and a simple species-aware Conditional LayerNorm), and we analyse post-hoc calibration and label-correlation smoothing for multi-label prediction. Throughout the paper we report not only ROC-AUC and micro-F1, but also resource usage (training time, GPU memory, and parameter count), with the goal of giving practitioners a realistic picture of the accuracy–cost trade-offs of different configurations.

Our strongest baseline is a GraphSAGE model with sum-based edge-to-node features, trained under a standardized, scriptable setup. We release all code and configuration files to support reproducible experimentation and further extensions.

II. Background

a) Problem: Given a directed graph $G = (V, E)$, an edge (i, j) has $\mathbf{e}_{ij} \in [0, 1]^8$. A node i has a multi-label target $\mathbf{y}_i \in \{0, 1\}^{112}$. Models output per-label logits $\hat{\mathbf{z}}_i \in \mathbb{R}^{112}$.

b) Message passing: Modern GNNs iteratively update node states via neighbor aggregation [5].

GraphSAGE [3] uses (weighted) means; GIN [4] uses sum-like updates with an MLP. In edge-network MPNNs [5], the edge weight α_{ij} is learned from $\mathbf{e}_{ij}$ [12], [13].

c) Normalization: **BN** [6] normalizes per batch; **LN** [7] per node. **Conditional LN (CLN)** conditions LN's affine parameters on auxiliary context (here: species), following conditional normalization ideas [8], [9], [14].

III. Related Work

a) GNNs with edge information: Message-passing neural networks (MPNNs) provide a general template for incorporating edge features via learned gates or edge-conditioned weights [5]. Variants such as Gated Graph Neural Networks and Residual Gated Graph ConvNets explicitly modulate messages with edge-dependent gating [12], [13]. Inductive baselines like GraphSAGE [3] aggregate neighbor states with (weighted) means, whereas GIN [4] emphasizes sum-like updates. Our study is complementary: rather than proposing a new architecture, we quantify how simple, *edge-aware* choices such as (i) aggregating 8-D edge evidence into node inputs and (ii) scalarizing edge channels to weight messages, shift the accuracy-cost frontier on `ogbn-proteins`.

Related work also covers link prediction and position/identity-aware message passing [23]–[25].

b) Normalization and conditioning: Normalization is a core component in deep GNNs. BatchNorm [6] and LayerNorm [7] are standard, while conditional normalization methods modulate affine parameters using side information [8], [9], [14]. We adopt a lightweight Conditional LayerNorm (CLN) that conditions on species descriptors and test whether it improves cross-species generalization under the official split (mouse→zebrafish).

c) Calibration and decision thresholds: Neural networks are often miscalibrated. Temperature scaling provides a simple, effective post-hoc fix [10]; broader post-hoc calibrators include Platt scaling [17], isotonic regression for probabilistic outputs [18], Bayesian binning into quantiles [19], and Dirichlet calibration [20]. While most prior reports on `ogbn-proteins` focus on ROC-AUC, we pair calibration with *per-label* thresholds and report ECE and Brier score [11], showing large gains in micro-F1 at essentially unchanged AUC.

d) Exploiting label dependencies: Multi-label methods often leverage inter-label structure, e.g., Classifier Chains [21] or GCNs on a learned label graph (ML-GCN) [22]. To keep the baseline simple and leakage-free, we compute a label co-occurrence matrix from training labels only and apply a small logit-space smoothing; we treat full label-graph learning as future work.

IV. Dataset and Setup

We follow the OGB `ogbn-proteins` protocol [1]. Initial node features are built by aggregating incident 8-D edge features with **mean**, **sum**, or **max**. We train on training species, validate on an unseen species, and test on a different unseen species. Hardware: NVIDIA A800-SXM4-40GB.

a) Species split: The validation split contains a single species (NCBI **10090**, mouse); the test split also contains a single species (**7955**, zebrafish). Our per-species plots (Sec. VI-D) therefore display one bar per split while comparing model variants.

b) Metrics: Primary: mean ROC-AUC across 112 labels. Secondary (fixed threshold): micro-F1@0.5, computed by applying a global 0.5 threshold to sigmoid probabilities for all labels. Decision quality (post-hoc): micro-$F1_{cal+thr}$, computed after per-label temperature scaling and per-label thresholds tuned on validation (Sec. VI-E).

c) Implementation: PyTorch Geometric [2]. Seeds $\{1, 2, 3\}$; early stopping on validation AUC. Each run exports `args.json`, `metrics.json`, and `logits_{val,test}.npz` with `node_id`, `species_id`, `logits[112]`, `labels[112]`.

Repository https://github.com/SV25-22/ECHO-Proteins

V. Methods

a) Edge→node feature construction: For each node i, we aggregate features of incident edges $\{\mathbf{e}_{ji}\}$ into a node input $\mathbf{x}_i$ using MEAN, SUM, or MAX. This isolates the contribution of edge evidence to node representations.

b) MLP (no-graph) baseline: To isolate the effect of message passing, we train a 3-layer, and 4-layer MLP directly on the edge→node features $\mathbf{x}_i$ to predict 112 logits (BCEWithLogits). Each hidden layer uses BN/LN/CLN + LeakyReLU + dropout. Training mirrors the GNN setup. See Sec. VI-B.

c) GraphSAGE / GIN with scalarized edges: We reduce $\mathbf{e}_{ij} \in \mathbb{R}^8$ to a scalar $\alpha_{ij} \in \mathbb{R}_+$ (default: channel SUM; we also test a learned 1-D scalarizer), and use α_{ij} to weight messages in SAGE/GIN. This retains a simple architecture while exploiting edge strength.

d) Add-on A - Normalization variants: We compare **LN**, **BN**, and **CLN** (LN whose affine parameters are predicted from a per-species descriptor; `cln_mode=desc`). Unless stated otherwise, SAGE uses hidden=512 and 3 layers.

e) Add-on B - Calibration & threshold: We apply post-hoc temperature scaling to validation logits to correct over- or under-confidence [17]–[20]. We consider two modes: a single global temperature parameter and per-label temperatures with L2 regularization toward the global value. Temperatures are optimized by minimizing the negative log-likelihood on the validation set. After calibration, we derive per-label thresholds by maximizing the F_β score on validation probabilities, falling back to ROC-based thresholds in degenerate cases. At test time, we apply the learned calibration and thresholds without re-fitting. To quantify calibration quality we compute the Expected Calibration Error (ECE) and the Brier score, in addition to ROC-AUC and F1.

f) Add-on C - Label correlation analysis: We construct a label–label co-occurrence matrix $P \in \mathbb{R}^{K \times K}$ ($K = 112$) from the training set labels. Each entry P_{jk} encodes how frequently label j and k appear together relative to the total occurrences of j, with rows normalized to sum to one. Predictions are then smoothed using:

$$z' = z + \lambda\, z P^\top \tag{1}$$

where z are the raw logits and λ is a small smoothing coefficient tuned on the validation set. This formulation allows information from correlated labels to adjust logits before thresholding. All correlation matrices are computed only from training labels to avoid data leakage [21], [22].

VI. Results

All means and standard deviations are over the three seeds $\{1, 2, 3\}$.

A. Main baselines

Table I summarizes the main GNN baselines under the standardized SAGE setup. With SAGE (sum, h=512, L=3; 3 seeds), **BN** attains the top ROC-AUC; **CLN** matches the AUC frontier while retaining much stronger fixed-threshold micro-F1@0.5. **LN** trails slightly in AUC but remains competitive with *sum* edge→node inputs. A learned 1-D edge scalarizer is close but does not surpass the simple *sum*. **GIN+BN** lags SAGE in AUC (0.761 vs. ∼0.79) but shows a higher fixed-threshold F1 than SAGE+BN (0.149 vs. 0.096), highlighting threshold/calibration sensitivity; overall, SAGE remains the better choice on this benchmark.

B. MLP baselines

To isolate the value of message passing, we train MLPs on edge-aggregated node features using mean, sum, and max aggregation. We evaluate 33 different configurations varying aggregation method, normalization (BatchNorm, LayerNorm, none), and architecture depth (uniform, tapering, deep). Table II reports representative MLP configurations across aggregation, normalization, and depth.

a) Key findings: **Aggregation:** SUM aggregation dominates top performers (all top 4 models), achieving 0.74+ test AUC vs. 0.69–0.72 for others. **Normalization:** BatchNorm shows best performance with SUM aggregation, while no normalization performs poorly, especially with SUM. **Architecture:** Uniform width (256, 256) and tapering (512→256→128) perform similarly well as the 4 layer deep network (512→256→256→128).

TABLE I: Main GNN baselines on OGBN-Proteins (3 layers, hidden size 512, sum edge-to-node features; results averaged over 3 seeds).

Model	Norm / scalar	Val AUC	Test AUC	Test F1	Params (M)
SAGE	BN / sum	**0.864**	**0.792**	0.096	0.65
SAGE	CLN / sum	0.855	**0.792**	0.145	1.71
SAGE	LN / learned1d	0.850	0.787	0.128	0.65
SAGE	LN / sum	0.855	0.789	0.144	0.65
GIN	BN / sum	0.845	0.761	**0.149**	0.85

TABLE II: MLP baselines on OGBN-Proteins. SUM aggregation consistently outperforms MEAN and MAX. BatchNorm provides optimal performance with SUM. VRAM is in MB and Params in millions.

Config	Val	Test	F1	VRAM	Par.
sum_deep_none	0.796	**0.743**	**0.145**	847	0.18
sum_taper_bn	0.799	**0.743**	0.128	1075	0.19
sum_deep_bn	**0.802**	0.742	0.125	1073	0.18
sum_uniform_256_bn	0.795	0.740	0.137	732	0.10
max_uniform_256_cln_desc	0.741	0.715	0.120	1021	0.10
mean_uniform_256_none	0.636	0.577	0.075	**603**	0.10
sum_taper_none	0.447	0.465	0.049	848	0.18

Calibration: SUM + BatchNorm models achieve the best calibration ($T_{\text{global}} \approx 0.95$–$1.00$), while LayerNorm models are overconfident ($T_{\text{global}} > 1.05$).

C. Edge-to-node aggregation ablation

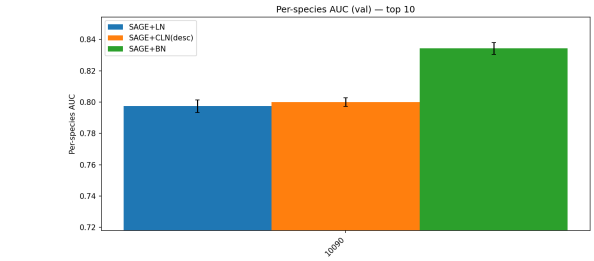
a) Takeaways: Table III shows that replacing MEAN with SUM for edge→node features improves both AUC and micro-F1. MAX is competitive but slightly below SUM. A simple learned 1-D scalarizer for edges is close to SUM but does not consistently beat it.

D. Per-species analysis

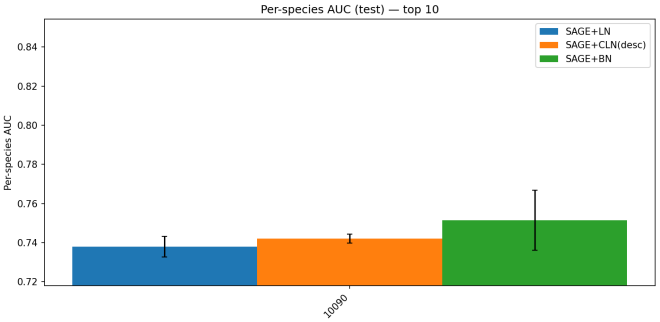
Table IV summarizes statistics of the training-derived co-occurrence matrix P. Because the benchmark has one validation species (mouse; 10090) and one test species (zebrafish; 7955), we compare normalization choices on those species using the new SAGE (hid=512, L=3) runs (seeds 1–3). As shown in Fig. 1, BN and CLN are statistically tied on mouse within error bars; on zebrafish, CLN and BN remain close, with LN slightly behind. This supports CLN as a robust alternative to BN for cross-species generalization.

TABLE III: Edge-to-node aggregation ablation for SAGE with LayerNorm (3 layers, hidden size 512; results averaged over 3 seeds).

Variant	Agg / scalar	Val AUC	Test AUC	Test F1	Params (M)
SAGE (LN)	max / sum	0.824	0.779	0.141	0.65
SAGE (LN)	mean / sum	0.839	0.775	0.138	0.65
SAGE (LN)	sum / learned1d	0.850	0.787	0.128	0.65
SAGE (LN)	sum / sum	**0.855**	**0.789**	**0.144**	0.65



(a) Validation species (10090)



(b) Test species (7955)

Fig. 1: Per-species AUC (mean ± s.d., 3 seeds) for SAGE (hid=512, L=3) with LN/BN/CLN.

TABLE IV: Summary of the training-derived label co-occurrence matrix P (row-normalized; computed from training labels only). We report off-diagonal statistics because each row sums to one by construction.

Statistic (training-only)	Value	Meaning
Number of labels (K)	112	Total functional labels
Mean off-diagonal P_{jk}	0.009	Avg. conditional co-occurrence mass
Max off-diagonal P_{jk}	0.061	Strongest conditional relation
Min off-diagonal P_{jk}	0.0003	Weakest conditional relation

a) Efficiency: As shown in Fig. 2, SAGE baselines form a favorable frontier of AUC vs. wall-clock and VRAM. CLN adds parameters versus LN/BN but remains efficient; GIN (not shown in the main table) remains a weaker baseline even when strengthened.

E. Calibration analysis

We apply per-label temperature scaling on validation logits and derive per-label thresholds (F_β=1). This markedly improves decision quality (micro-F1) and calibration (ECE) while leaving AUC essentially unchanged. Adding label-correlation smoothing (conditional-centered graph, λ=0.1, logit space) provides small, consistent gains.

a) Takeaways: Table V reports test metrics before/after label-correlation smoothing. Per-label temperature + thresholds dramatically improves micro-F1 (e.g., SAGE+LN from ∼0.14 (micro-F1@0.5) to ∼0.80 (micro-$F1_{cal+thr}$) with negligible AUC change. Smoothing yields small, consistent improvements and slightly better ECE for LN/CLN. BN remains the least calibrated even after post-hoc correction.

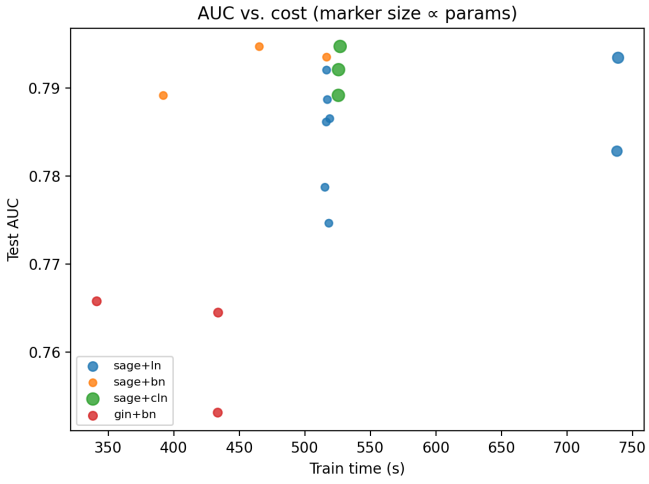


Fig. 2: AUC vs. training cost. Marker size ∝ parameter count; color encodes model+norm. The Pareto frontier is dominated by SAGE variants.

TABLE V: Test metrics before (left) and after (right) label-correlation smoothing. All F1 values are micro-$F1_{cal+thr}$ (per-label temperature scaling + per-label thresholds).

Model	AUC	micro-$F1_{cal+thr}$	ECE
SAGE+LN (sum)	0.792 → 0.794	0.795 → 0.796	0.188 → 0.178
SAGE+CLN (sum)	0.795 → 0.796	0.786 → 0.787	0.183 → 0.173
SAGE+BN (sum)	0.795 → 0.794	0.592 → 0.587	0.350 → 0.348

F. Label correlation

a) Key findings: Because P is row-normalized, each row sums to one by construction; therefore, "outgoing/incoming" sums near 1 are not informative and are not reported. The small mean off-diagonal value (0.009) indicates that, on average, co-occurrence mass is spread thinly across labels, while the maximum value (0.061) shows that a small number of label pairs have noticeably stronger conditional associations. This supports using only light smoothing (λ small) to avoid over-coupling largely independent labels.

VII. Conclusion

Seemingly small design choices matter on OGBN-Proteins and, more broadly, on large edge-attributed graphs used in bioengineering and biomedicine. Using sum aggregation for edge-to-node feature construction consistently improves GraphSAGE; BatchNorm delivers the best AUC, while Conditional LayerNorm matches the AUC frontier and avoids BatchNorm's poor fixed-threshold behavior. Post-hoc per-label temperature scaling and per-label thresholds are essential for reliable multi-label decisions, and light label-correlation smoothing adds small, consistent gains. A frozen-gate MPNN collapses as expected; we outline a practical unfrozen configuration. We standardize outputs (`args.json`, `metrics.json`, `logits_{train,val,test}.npz`) and ship scripts of the runs to support transparent reruns and analysis, giving practitioners a ready-to-use baseline implementation in PyTorch Geometric.

VIII. Future Work

a) Unfrozen edge-network MPNN: Train the edge gate end-to-end and compare to SAGE at matched params/VRAM (3 seeds). A practical starting point is: `--backend sparse --hid 256 --gate_hid 64 --dropout 0.1 --epochs 120 --patience 12 --lr 2e-3 --amp 0 --norm ln/cln --alpha_chunk 2000000 --freeze_edge 0` (with `--backend scatter` + chunking if memory is tight). Report AUC/F1/ECE.

b) Graph Transformers with edge channels: Evaluate a lightweight graph transformer that ingests the 8-D edge evidence as attention biases or per-head gates. Compare against SAGE at matched params/VRAM to see if attention helps on cross-species transfer [15], [16].

c) Stronger calibration: Beyond temperature scaling, try vector scaling / classwise Platt / isotonic per label; add *species-conditional* temperatures (fit on validation species, apply to test). Report micro/macro F1, ECE, and reliability plots.

d) Cost-sensitive thresholds: Optimize thresholds for target operating points (e.g., fixed precision 90% or fixed recall 80%), and show precision–recall trade-offs per label family. This reflects realistic lab-screening constraints.

e) Label graph learning: Replace heuristic P with a learned label–label graph (e.g., from a small GNN over labels) trained on validation only [22]. Add GO hierarchy priors and compare logit- vs. prob-space smoothing, including per-label λ.

ACKNOWLEDGMENT

We gratefully acknowledge computational support from Xinming Wang at the Institute of Automation, Chinese Academy of Sciences (CASIA).

REFERENCES

[1] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open Graph Benchmark: Datasets for machine learning on graphs," in *Advances in Neural Information Processing Systems*, 2020.
[2] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR (Workshop)*, 2019.
[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *ICLR*, 2019.
[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017.
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
[7] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv:1607.06450, 2016.
[8] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *NeurIPS*, 2017.
[9] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017.
[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML*, 2017.
[11] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.
[12] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *ICLR*, 2016.
[13] X. Bresson and T. Laurent, "Residual gated graph ConvNets," arXiv:1711.07553, 2017.
[14] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.
[15] C. Ying et al., "Do transformers really perform bad for graph representation? (Graphormer)," in *NeurIPS*, 2021.
[16] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," in *AAAI DLG Workshop*, 2021.
[17] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999.
[18] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *KDD*, 2002.
[19] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well-calibrated probabilities using Bayesian binning into quantiles," in *AAAI*, 2015.
[20] M. Kull, M. Perello-Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach, "Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration," in *NeurIPS*, 2019.
[21] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *ECML PKDD*, 2009.
[22] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *CVPR*, 2019.
[23] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *ICML*, 2019.
[24] J. You, J. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," arXiv:2101.10320, 2021.
[25] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Labeling trick: A theory of using graph neural networks for multi-node representation learning," arXiv:2010.16103, 2020.