Edge-aware baselines for `ogbn-proteins` in PyTorch Geometric

Species-wise normalization, post-hoc calibration, and cost–accuracy trade-offs

Aleksandar Stanković* Dejan Lisica†

November 18, 2025

Abstract

We present reproducible, edge-aware baselines for `ogbn-proteins` in PyTorch Geometric (PyG). We study two system choices that dominate practice: (i) how 8-D edge evidence is aggregated into node inputs, and (ii) how edges are used inside message passing. Our strongest baseline is GraphSAGE with *sum*-based edge→node features. We compare LayerNorm (LN), BatchNorm (BN), and a species-aware Conditional LayerNorm (CLN), and report compute cost (time, VRAM, parameters) together with accuracy (ROC-AUC) and decision quality. In our primary experimental setup (hidden=512, 3 layers, 3 seeds), *sum* consistently beats *mean*/*max*; BN attains the best AUC, while CLN matches the AUC frontier with better thresholded F1. Finally, post-hoc *per-label* temperature scaling plus per-label thresholds substantially improves micro-F1 and ECE with negligible AUC change, and light label-correlation smoothing yields small additional gains. We release standardized artifacts and scripts used for all of the runs presented in the paper.

1 Introduction

Many real-world problems are graphs: nodes are entities and edges describe relations. In `ogbn-proteins` [1], nodes are proteins and edges carry an 8-D vector of association evidence; each protein has 112 binary functional labels. The official metric is mean ROC-AUC across labels, with a *species-wise* split: validation and test contain species unseen during training.

This work focuses on two practical design questions: **(i)** how to construct node inputs from incident 8-D edge features; and **(ii)** how to incorporate edge information inside message passing. We then compare normalization schemes (BN/LN/CLN) and report compute cost. Our strongest baseline is GraphSAGE, trained and reported under a standardized artifact format for reproducibility.

2 Background

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 $\hat{\mathbf{z}}_i \in \mathbb{R}^{112}$.

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

$$\mathbf{h}_i^{(\ell+1)} = \sigma\Big(\mathbf{W}_{\text{self}}\mathbf{h}_i^{(\ell)} + \text{AGG}_{j\in\mathcal{N}(i)}\, \alpha_{ij}\, \mathbf{W}\mathbf{h}_j^{(\ell)} + \mathbf{b}\Big).$$

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].

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].

*Faculty of Technical Sciences, University of Novi Sad. `stankovic.sv25.2022@uns.ac.rs`
†Faculty of Technical Sciences, University of Novi Sad. `lisica.sv49.2022@uns.ac.rs`

3 Related Work

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—(i) aggregating 8-D edge evidence into node inputs; (ii) scalarizing edge channels to weight messages—shift the accuracy–cost frontier on `ogbn-proteins`.
Our baselines also sit alongside work that studies the expressive power of GNNs for multi-node prediction tasks such as link prediction [25] and identity-/position-aware message passing [23, 24].

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).

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.

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.

4 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.

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. 6.4) therefore display one bar per split while comparing model variants.

Metrics. Primary: mean ROC-AUC across 112 labels. Secondary: micro-F1 at a fixed 0.5 threshold.

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

5 Methods

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.

MLP (no-graph) baseline. To isolate the effect of message passing, we train a 3-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. 6.2.

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.

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.

Add-on B: Calibration & thresholds. We apply post-hoc temperature scaling to validation logits to correct over- or under-confidence [17, 18, 19, 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.

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:

$$z' = z + \lambda\, zP^\top$$

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].

6 Results

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

6.1 Main baselines

Model	Norm	Edge→node	Layers	Hid	Edge scalar	Val AUC	Test AUC	Test F1	Params (M)
sage	bn	sum	3	512	sum	0.864	0.792	0.096	0.650
sage	cln	sum	3	512	sum	0.855	0.792	0.145	1.710
sage	ln	sum	3	512	learned1d	0.850	0.787	0.128	0.650
sage	ln	sum	3	512	sum	0.855	0.789	0.144	0.650
gin	bn	sum	3	512	sum	0.845	0.761	0.149	0.852

Table 1: Main GNN baselines on `ogbn-proteins` (3 seeds).

Observations. 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. **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.

6.2 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).

MLP Configuration	**Val AUC**	**Test AUC**	**Test micro-F1@0.5**	**VRAM (MB)**	**Params (M)**
sum_deep_none (best)	0.796	**0.743**	0.145	847	0.181
sum_taper_bn	0.799	**0.743**	0.128	1075	0.185
sum_deep_bn	0.802	**0.742**	0.125	1073	0.183
sum_uniform_256_bn	0.795	**0.740**	0.137	732	0.098
max_uniform_256_cln_desc	0.741	**0.715**	0.120	1021	0.098
mean_uniform_256_none	0.636	**0.577**	0.075	603	0.097
sum_taper_none (worst)	0.447	**0.465**	0.049	848	0.183

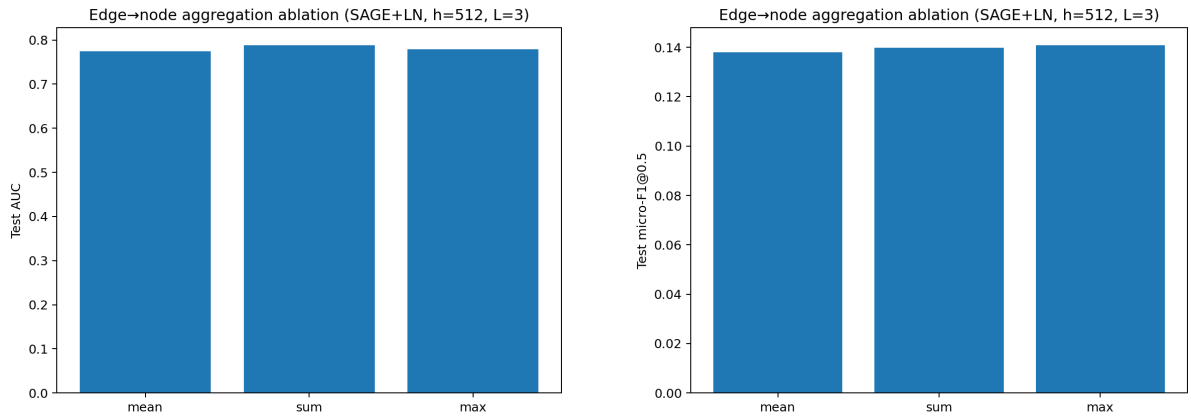
Table 2: MLP baselines on `ogbn-proteins`. SUM aggregation consistently outperforms MEAN and MAX. BatchNorm provides optimal performance with SUM aggregation, while no normalization performs poorly. The best MLP achieves 0.743 test AUC, demonstrating that edge aggregation alone captures significant signal.

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

6.3 Edge-to-node aggregation ablation

Model	Norm	Edge→node	Layers	Hid	Edge scalar	Val AUC	Test AUC	Test F1	Params
sage	ln	max	3	512	sum	0.824 ± 0.000	0.779	0.141	0.650
sage	ln	mean	3	512	sum	0.839	0.775	0.138	0.650
sage	ln	sum	3	512	learned1d	0.850	0.787	0.128	0.650
sage	ln	sum	3	512	sum	0.855	0.789	0.144	0.650

Table 3: Edge→node aggregation ablation for SAGE+LN (h=512, L=3).



(a) Test AUC

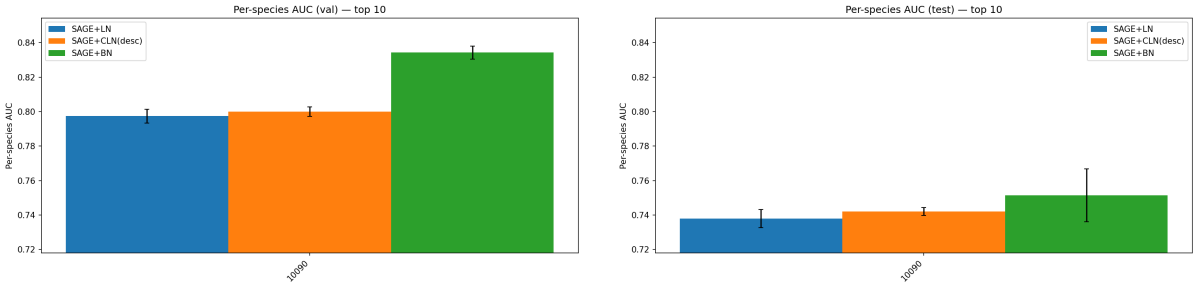
(b) Test micro-F1@0.5

Figure 1: **Aggregation ablation** with SAGE+LN (3 seeds).

Takeaways. 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.

6.4 Per-species analysis

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). 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.



(a) Validation species (10090)

(b) Test species (7955)

Figure 2: **Per-species AUC** (mean ± s.d., 3 seeds) for SAGE (hid=512, L=3) with LN/BN/CLN.

6.5 AUC vs. cost

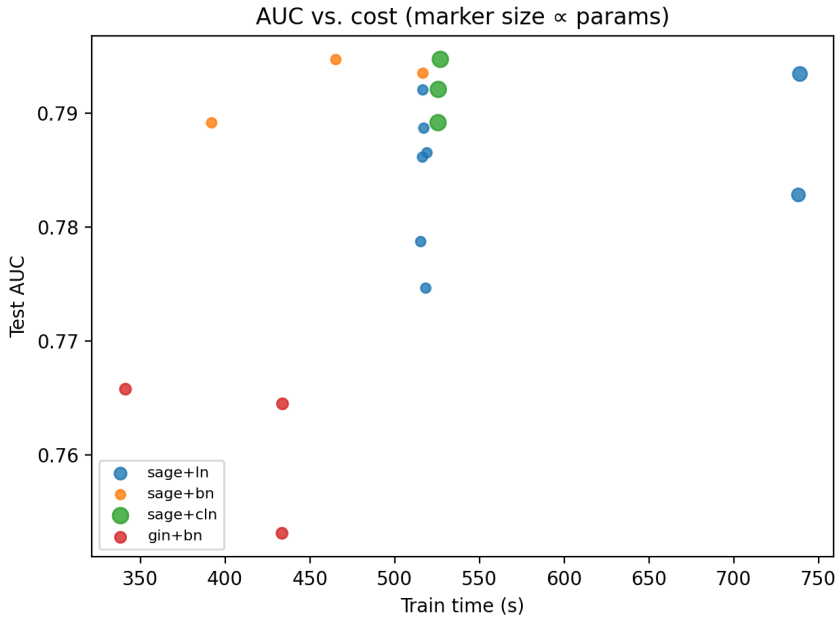


Figure 3: **AUC vs. training cost.** Marker size ∝ parameter count; color encodes model+norm. The Pareto frontier is dominated by SAGE variants.

Efficiency. On our machine, 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.

6.6 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.

	Calib (per-label T + thresholds)			+ Smoothing (λ=0.1)		
Model	**AUC**	**F1$_{\text{micro}}$**	**ECE**	**AUC**	**F1$_{\text{micro}}$**	**ECE**
SAGE+LN (sum)	0.792	0.795	0.188	0.794	0.796	0.178
SAGE+CLN(desc) (sum)	0.795	0.786	0.183	0.796	0.787	0.173
SAGE+BN (sum)	0.795	0.592	0.350	0.794	0.587	0.348

Table 4: Test metrics after per-label temperature scaling & per-label thresholds, with/without label-correlation smoothing (conditional-centered, logit space). Values are rounded from the new runs.

Takeaways. Per-label temperature + thresholds dramatically improves micro-F1 (e.g., SAGE+LN from 0.14@0.5 to ∼0.80) 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.

6.7 Label correlation

Correlation Metric	Value	Interpretation
Number of labels	112	Total protein functions
Sparsity	0.89%	Very sparse label co-occurrence
Mean correlation	0.009	Low pairwise correlations
Max correlation	0.061	Some strong functional relationships
Min correlation	0.0003	Many independent functions
Mean outgoing correlation	0.999996	Very high directional consistency
Mean incoming correlation	0.999996	Balanced directional relationships

Table 5: Label correlation statistics.

Key findings. **Sparsity:** The 0.89% sparsity indicates that most protein function pairs are independent, with only a small fraction showing meaningful co-occurrence. **Directional relationships:** The extremely high outgoing and incoming correlations (0.999996) suggest strong directional consistency in functional dependencies. **Biological interpretation:** The low mean correlation (0.009) with some high maximum correlations (0.061) indicates that while most protein functions are independent, there exist specific functional modules with strong interdependencies.

7 Conclusion

Seemingly small design choices matter on `ogbn-proteins`. Using *sum* for edge→node feature construction consistently improves GraphSAGE; BN delivers the best AUC, while CLN matches the AUC frontier and avoids BN'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.

Limitations. We did not include unfrozen edge-network MPNN results due to time; our configuration is provided for straightforward reproduction.

8 Future Work

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.

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].

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.

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.

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.

References

[1] Hu, W. et al. *Open Graph Benchmark: Datasets for Machine Learning on Graphs*. NeurIPS 2020.

[2] Fey, M., Lenssen, J.E. *Fast Graph Representation Learning with PyTorch Geometric*. ICLR (Workshop) 2019.

[3] Hamilton, W.L., Ying, R., Leskovec, J. *Inductive Representation Learning on Large Graphs*. NeurIPS 2017.

[4] Xu, K. et al. *How Powerful are Graph Neural Networks?* ICLR 2019.

[5] Gilmer, J. et al. *Neural Message Passing for Quantum Chemistry*. ICML 2017.

[6] Ioffe, S., Szegedy, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. ICML 2015.

[7] Ba, J.L., Kiros, J.R., Hinton, G.E. *Layer Normalization*. arXiv:1607.06450.

[8] de Vries, H. et al. *Modulating early visual processing by language*. NeurIPS 2017. (conditional normalization)

[9] Huang, X., Belongie, S. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. ICCV 2017.

[10] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q. *On Calibration of Modern Neural Networks*. ICML 2017.

[11] Brier, G.W. *Verification of forecasts expressed in terms of probability*. Monthly Weather Review, 1950.

[12] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R. *Gated Graph Sequence Neural Networks*. ICLR 2016. arXiv:1511.05493.

[13] Bresson, X., Laurent, T. *Residual Gated Graph ConvNets*. arXiv:1711.07553, 2017.

[14] Perez, E., Strub, F., de Vries, H., Dumoulin, V., Courville, A. *FiLM: Visual Reasoning with a General Conditioning Layer*. AAAI 2018.

[15] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.-Y. *Do Transformers Really Perform Bad for Graph Representation?* (Graphormer). NeurIPS 2021.

[16] Dwivedi, V. P., Bresson, X. *A Generalization of Transformer Networks to Graphs*. AAAI DLG Workshop 2021. arXiv:2012.09699.

[17] Platt, J. *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*. In: Smola et al. (eds) Advances in Large Margin Classifiers, 1999.

[18] Zadrozny, B., Elkan, C. *Transforming Classifier Scores into Accurate Multiclass Probability Estimates*. KDD 2002.

[19] Naeini, M. P., Cooper, G., Hauskrecht, M. *Obtaining Well-Calibrated Probabilities Using Bayesian Binning into Quantiles*. AAAI 2015.

[20] Kull, M., Perello-Nieto, M., Kängsepp, M., Silva Filho, T., Song, H., Flach, P. *Beyond Temperature Scaling: Obtaining Well-Calibrated Multiclass Probabilities with Dirichlet Calibration*. NeurIPS 2019.

[21] Read, J., Pfahringer, B., Holmes, G., Frank, E. *Classifier Chains for Multi-label Classification*. ECML PKDD 2009.

[22] Chen, Z.-M., Wei, X.-S., Wang, P., Guo, Y. *Multi-Label Image Recognition with Graph Convolutional Networks*. CVPR 2019.

[23] You, J., Ying, R., Leskovec, J. Position-aware Graph Neural Networks. In *ICML*, 2019.

[24] You, J., Gomes-Selman, J., Ying, R., Leskovec, J. Identity-aware Graph Neural Networks. arXiv preprint arXiv:2101.10320, 2021.

[25] Zhang, M., Li, P., Xia, Y., Wang, K., Jin, L. Revisiting Graph Neural Networks for Link Prediction. arXiv preprint arXiv:2010.16103, 2020.