On Multi-Task Cross-Domain Pre-training for Graph Neural Networks

Alon Bebchuk (ID: 314023516), Ben Cohen (ID: 323855288), and Tim Kushmaro (ID: 212052708)

Abstract

While GNN pre-training approaches suggest potential for cross-domain performance improvements, practitioners report inconsistent results. We challenge this through systematic evaluation of 96 controlled experiments across 6 domains and 8 pre-training schemes. Our analysis reveals a fundamental asymmetry: **computational efficiency gains vary by domain-scheme combinations**, while **performance improvements are selective**, requiring precise matching. We identify and illustrate four predictable failure patterns with concrete examples and establish evidence-based deployment protocols. This framework transforms GNN pre-training from trial-and-error into systematic optimization through detailed efficiency and performance analysis.

Keywords: graph neural networks, pre-training, transfer learning, multi-task learning, computational efficiency, systematic evaluation, domain adaptation, failure analysis

Code and Data: Full implementations, models, and reproducible experiments available at `https://github.com/alonbebchuk/GNN-Pretraining.git`

1 Introduction

Graph Neural Networks (GNNs) are widely used for learning representations from relational data across diverse domains including molecules, proteins, citation networks, and knowledge graphs (Zhou et al., 2020). Unlike computer vision and NLP, where systematic pre-training is foundational (He et al., 2016; Devlin et al., 2019), GNNs are still commonly trained from scratch per task and domain. Prior GNN pre-training work typically focuses on either a single pretext task or a single domain (Hu et al., 2019), leaving the core research question unanswered: **How does multi-task cross-domain training transfer to different domains and tasks?**

We address this through a comprehensive empirical evaluation of multi-task, cross-domain GNN pre-training across 96 controlled experiments compared against from-scratch training baselines. Our systematic approach reveals fundamental asymmetries in transfer effectiveness and establishes evidence-based guidelines for practitioners.

Why this matters. GNN pre-training demonstrates computational and performance potential that can be systematically unlocked through principled optimization. First, **computational efficiency gains** vary significantly by domain-scheme combinations, with clear efficiency clustering and domain-specific optimization patterns. Second, **selective performance improvements** require precise domain-scheme matching, with notable gains achieved through systematic combinations.

Key Contributions. We establish: (1) **Systematic Evaluation Framework** - 96 controlled experiments revealing efficiency and performance patterns, (2) **Parameter-Performance Disconnect** - fundamental gap between parameter reduction and actual speedups, (3) **Evidence-Based Guidelines** - empirical matrices for reliable domain-scheme selection, and (4) **Failure Mode Taxonomy** - predictable patterns enabling systematic avoidance strategies.

2 Related Work

Graph pre-training approaches. Recent comprehensive surveys (Xie et al., 2022; Liu et al., 2022b) categorize self-supervised graph learning approaches including information-theoretic methods (Veličković et al., 2019; Sun et al., 2020), gen-

erative masking (Hu et al., 2019), contrastive approaches (You et al., 2020; Hou et al., 2022), and large-scale frameworks (Qiu et al., 2020; Liu et al., 2022a). However, these typically evaluate single methods on specific domains without systematic multi-task or cross-domain analysis.

Multi-task learning in graphs. While multi-task learning (Caruana, 1997) and transfer learning (Pan and Yang, 2010) benefit computer vision and NLP, their application to graph pre-training remains understudied. Prior work identifies gradient interference issues and proposes solutions like PCGrad (Yu et al., 2020), but lacks comprehensive evaluation of multi-task graph pre-training effectiveness across domains.

Our contributions. We provide the first systematic evaluation of multi-task graph pre-training across domains, revealing efficiency-performance asymmetries and establishing evidence-based deployment protocols with failure mode analysis.

3 Methodology

We evaluate 96 controlled experiments across molecular (MUTAG, PROTEINS, NCI1, ENZYMES) and citation (Cora, CiteSeer) datasets (Morris et al., 2020; Sen et al., 2008) (see Appendix A.5 for data processing details). Task types: Graph Classification (GC), Node Classification (NC), Link Prediction (LP). Transfer strategies: Linear Probing (LIN, frozen backbone) vs Full Fine-tuning (FT, all parameters). 3 seeds for statistical reliability (see Appendix A.6 for evaluation protocol).

Experimental design. Architecture: domain-adaptive encoders, 5-layer GIN backbone (Xu et al., 2019), task-specific heads (see Appendix A.1 for architecture specifications). Six pre-training objectives: node masking (Hou et al., 2022), link prediction (Liben-Nowell and Kleinberg, 2007), contrastive learning (You et al., 2020), graph properties (Newman, 2003), domain adversarial training (Ganin et al., 2016) (see Appendix B for technical details). Multi-task optimization uses PCGrad (Yu et al., 2020) (see Appendix A.3).

Pre-training schemes. We evaluate 8 schemes with varying task complexity: **b1** (no pre-training baseline), **b2** (node masking only), **b3** (node contrastive only), **b4** (5-task: node masking, link prediction, node contrastive, graph contrastive, graph properties), **s1** (2-task: node masking, link prediction), **s2** (2-task: node contrastive, graph contrastive), **s3** (4-task: node masking, link prediction, node contrastive, graph contrastive), **s4** (5-task: s3 + graph properties), **s5** (6-task: s4 + domain adversarial). All schemes train on 4 molecular datasets except b4 (ENZYMES only). We test 8 schemes × 6 datasets × 2 transfer strategies (linear probing (Chen et al., 2020) vs full fine-tuning) = 96 pre-training experiments, compared against from-scratch baseline controls, yielding 324 total runs across 3 statistical seeds. All experiments use identical architectures and hyperparameters for fair comparison (see Appendix A.2 and C for hyperparameter details). All models are implemented using PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey and Lenssen, 2019).

Research Questions and Contributions. We address four fundamental questions: **RQ1**: Do efficiency gains generalize universally? *Answer: No - efficiency patterns are domain-scheme specific, with best overall speedups reaching 1.70x (Cora_LP+b3) and varied time-convergence trade-offs.* **RQ2**: When do performance gains occur? *Answer: Only with precise domain-scheme matching (26.2% success rate).* **RQ3**: How should practitioners balance efficiency vs performance? *Answer: Consult domain-scheme efficiency matrices; efficiency and performance gains exist but require careful selection.* **RQ4**: Can failures be predicted? *Answer: Yes - we identify four systematic failure patterns with predictable characteristics.*

4 Results

Our 96-experiment evaluation reveals fundamental asymmetries in GNN pre-training effectiveness across domains and tasks compared to from-scratch baselines.

RQ1 - Efficiency Gains Across Domains and Schemes. Pre-training demonstrates varied computational advantages across different domain-scheme combinations for both transfer strategies (Tables 1 and 2). **Full fine-tuning strategy** shows notable overall speedups with CiteSeer_NC multi-task schemes (s4: 1.54x, s5: 1.51x, s2: 1.45x, s1: 1.43x), combining both time and convergence benefits. **Linear probing strategy** shows superior overall speedups with Cora_LP combinations (b3: 1.70x, b2: 1.68x, s2/s4: 1.67x) and CiteSeer_NC multi-task schemes (s3: 1.61x, s2/s4: 1.57x), while

Domain	b2	b3	s1	s2	s3	s4	s5
Time Per Epoch Speedup							
CiteSeer_LP	**2.18**	**1.35**	**1.92**	**2.68**	**2.27**	**1.67**	0.85
CiteSeer_NC	**1.27**	**1.19**	**1.49**	**1.54**	**1.58**	**1.47**	**1.38**
Cora_LP	0.01	0.25	0.07	0.01	0.03	0.56	0.23
Cora_NC	**1.13**	**1.16**	**1.12**	**1.13**	0.92	**1.22**	0.96
ENZYMES	**1.66**	**1.43**	**1.57**	**1.92**	**1.70**	**1.63**	**1.62**
PTC_MR	0.45	0.89	0.88	0.42	0.60	**1.19**	**1.35**
Convergence Speedup							
CiteSeer_LP	0.43	0.66	0.42	0.32	0.34	0.57	**1.26**
CiteSeer_NC	0.97	**1.04**	0.96	0.94	0.89	**1.05**	**1.10**
Cora_LP	**104.3**	**5.31**	**19.6**	**104.3**	**44.7**	**1.96**	**5.91**
Cora_NC	0.92	0.90	0.97	**1.05**	**1.19**	0.97	**1.10**
ENZYMES	0.52	0.64	0.60	0.46	0.54	0.54	0.55
PTC_MR	**2.52**	**1.12**	**1.05**	**2.70**	**1.68**	0.72	0.60
Overall Speedup (Time × Convergence)							
CiteSeer_LP	0.93	0.89	0.80	0.85	0.77	0.96	**1.07**
CiteSeer_NC	**1.23**	**1.23**	**1.43**	**1.45**	**1.41**	**1.54**	**1.51**
Cora_LP	**1.48**	**1.34**	**1.44**	**1.48**	**1.46**	**1.10**	**1.34**
Cora_NC	**1.04**	**1.05**	**1.09**	**1.18**	**1.09**	**1.18**	**1.06**
ENZYMES	0.87	0.91	0.94	0.88	0.92	0.87	0.89
PTC_MR	**1.13**	0.99	0.93	**1.13**	**1.02**	0.85	0.81

Table 1: **Full Fine-tuning Strategy** - Complete efficiency comparison vs. from-scratch full fine-tuning baseline (b1). Shows time per epoch speedup, convergence speedup, and overall speedup across domain-scheme combinations. Values >1 indicate speedup, <1 indicate slower performance.

Domain	b2	b3	s1	s2	s3	s4	s5
Time Per Epoch Speedup							
CiteSeer_LP	**4.38**	**4.56**	**4.87**	**5.33**	**4.38**	**4.26**	**4.23**
CiteSeer_NC	**1.32**	**1.51**	**1.66**	**1.75**	**1.53**	**1.30**	**1.49**
Cora_LP	0.02	0.02	0.74	0.02	**1.13**	0.02	0.31
Cora_NC	**1.18**	**1.14**	**1.38**	**1.50**	**1.36**	**1.43**	**1.25**
ENZYMES	**2.30**	**1.60**	**2.06**	**2.68**	**2.31**	**2.78**	**2.75**
PTC_MR	**1.54**	0.55	**1.08**	0.73	0.82	0.52	**1.06**
Convergence Speedup							
CiteSeer_LP	0.19	0.15	0.14	0.13	0.19	0.20	0.19
CiteSeer_NC	**1.00**	0.93	0.87	0.90	**1.05**	**1.20**	**1.01**
Cora_LP	**104.3**	**104.3**	**1.69**	**104.3**	**1.11**	**78.3**	**4.82**
Cora_NC	0.93	0.95	0.92	0.76	0.83	0.81	0.97
ENZYMES	0.48	0.85	0.61	0.41	0.48	0.39	0.38
PTC_MR	0.64	**2.32**	0.97	**1.55**	**1.38**	**2.47**	0.97
Overall Speedup (Time × Convergence)							
CiteSeer_LP	0.83	0.70	0.70	0.71	0.84	0.83	0.82
CiteSeer_NC	**1.32**	**1.40**	**1.45**	**1.57**	**1.61**	**1.57**	**1.50**
Cora_LP	**1.68**	**1.70**	**1.26**	**1.67**	**1.26**	**1.67**	**1.47**
Cora_NC	**1.09**	**1.08**	**1.27**	**1.15**	**1.12**	**1.16**	**1.21**
ENZYMES	**1.11**	**1.37**	**1.26**	**1.10**	**1.12**	**1.08**	**1.06**
PTC_MR	**1.00**	**1.27**	**1.05**	**1.12**	**1.14**	**1.30**	**1.03**
Parameter Efficiency (vs Full FT)							
CiteSeer_LP	**2.2x**						
CiteSeer_NC	**2.4x**						
Cora_LP	**3.3x**						
Cora_NC	**4.6x**						
ENZYMES	**40.3x**						
PTC_MR	**35.3x**						

Table 2: **Linear Probing Strategy** - Complete efficiency comparison vs. from-scratch full fine-tuning baseline (b1). Shows time per epoch speedup, convergence speedup, overall speedup, and parameter efficiency across domain-scheme combinations. Values >1 indicate speedup, <1 indicate slower performance. Parameter efficiency demonstrates substantial reduction by freezing GNN backbones.

providing substantial parameter reduction (2.2x-40.3x) by freezing GNN backbones.

RQ2 - Selective Performance Success. Performance improvements require precise domain-scheme matching. Our analysis reveals 74% of cross-domain transfers show negative performance, while optimal matches achieve notable gains (s1_FT→PTC_MR: +35.8% peak, s2_LIN→CiteSeer_LP: +15.8%). Figure 1 visualizes these domain-specific patterns, showing clear performance clustering by domain-scheme combinations.

RQ3 - Strategy Selection Trade-offs. The choice between **linear probing** and **full fine-tuning strategies** creates distinct efficiency-performance trade-offs, with patterns varying significantly by domain-scheme combinations (Tables 1 and 2). Link prediction tasks strongly favor linear probing strategy (b3_LIN→CiteSeer_LP: +17.0% vs b3_FT→CiteSeer_LP: +2.4%), while molecular classification shows mixed patterns—PTC_MR benefits from full fine-tuning strategy (s1_FT: +35.8% vs s1_LIN: +17.2%), ENZYMES shows no substantial improvement with either strategy.

RQ4 - Deployment Framework. Our systematic evaluation transforms ad-hoc pre-training into principled deployment through a five-step evidence-based protocol. **Step 1**: Identify task type (GC/NC/LP) to determine optimal transfer strategy (linear probing vs full fine-tuning). **Step 2**: Characterize domain properties (molecular vs citation networks) for scheme compatibility assessment. **Step 3**: Consult our empirical efficiency and performance matrices to rank candidate combinations. **Step 4**: Balance peak performance potential against consistency requirements based on deployment constraints. **Step 5**: Implement with multiple seeds for statistical reliability (see Appendix A.6 for detailed protocol). Figure 2 provides the task-type specialization patterns essential for systematic

selection.

Systematic Failure Mode Analysis. Our analysis reveals four predictable failure patterns that enable proactive avoidance strategies:

Dataset-Specific Brittleness: Some datasets exhibit consistent negative transfer regardless of pre-training configuration, suggesting fundamental difficulties with transfer learning approaches. ENZYMES exemplifies this pattern, showing -5.0% to -14.2% degradation across all schemes—including b4 which pre-trains exclusively on ENZYMES itself—indicating intrinsic dataset characteristics that resist transfer learning benefits.

Task Complexity Gaps: Mismatched complexity between pre-training tasks and downstream requirements creates dramatic performance variations. Multi-task scheme s1 demonstrates this dichotomy: achieving exceptional +35.8% gains on PTC_MR while failing at -6.1% on Cora_LP, where simpler schemes (b2: +4.3%, b3: +3.1%) succeed, suggesting the complex multi-task approach overwhelms simpler downstream requirements.

Optimization Interference: Complex multi-task schemes can underperform simpler alternatives due to conflicting gradient updates. On PTC_MR, simple scheme b2 (+11.3%) outperforms complex scheme s2 (+3.8%), revealing optimization interference despite PCGrad mitigation (see Appendix A.3).

Strategy Misalignment: Wrong transfer strategy selection dramatically reduces effectiveness. CiteSeer_LP illustrates this clearly: b3_LIN achieves +17.0% while b3_FT manages only +2.4%—an 85% reduction in gains purely from strategy misalignment.

Strategy-Specific Performance Analysis. Our systematic evaluation across all downstream tasks reveals fundamental differences between transfer strategies (Tables 3 and 4). **Full Fine-tuning Patterns:** Shows selective effectiveness with PTC_MR achieving strong gains (s1: +35.8%, s3: +9.4%, s5: +7.5%) while most other domains exhibit negative transfer, particularly Node Classification tasks (-0.6% to -15.8%). CiteSeer_LP shows mixed results with only s2 achieving positive transfer (+2.4%). **Linear Probing Excellence:** Demonstrates superior performance on Link Prediction tasks, with both Cora_LP (b2: +4.3%, b3: +3.1%, s3: +4.3%) and CiteSeer_LP (b2: +8.4%, b3: +17.0%, s1: +10.2%, s2: +15.8%, s3: +6.6%, s4: +2.1%, s5: +1.2%) showing consistent positive transfer. However, shows degradation on Graph Classification tasks, with only PTC_MR maintaining some positive results (b2: +1.7%, s1: +17.2%). **Critical Finding:** Strategy selection is task-type dependent—Link Prediction strongly favors Linear Probing with consistent positive transfer, while Graph Classification requires Full Fine-tuning for optimal results.

Scheme	PTC_MR (GC)	ENZ (GC)	Cora_NC (NC)	Cite_NC (NC)	Cora_LP (LP)	Cite_LP (LP)
b2	**+11.3**	-5.0	-6.3	-0.1	-4.1	**+0.9**
b3	**+7.5**	-5.0	-0.6	-9.6	-3.3	-3.1
b4	0.0	-0.8	-4.0	-5.9	-0.2	-1.7
s1	**+35.8**	-14.2	-2.9	-15.2	-11.8	-12.8
s2	**+3.8**	-5.8	-4.4	-10.7	-4.7	**+2.4**
s3	**+9.4**	-5.8	-1.3	-6.7	-4.6	-4.2
s4	-3.8	-6.7	-5.4	-15.8	-3.8	-1.2
s5	**+7.5**	-2.5	-6.3	-11.9	-8.0	-4.6

Table 3: **Full Fine-tuning Performance Analysis.** Performance improvements across all downstream tasks using full fine-tuning strategy. Values show percentage improvement over from-scratch baseline (b1). Positive values indicate performance gains, negative values indicate degradation. Task types: Graph Classification (GC), Node Classification (NC), Link Prediction (LP). See Appendix B for detailed pre-training task specifications.

Scheme	PTC_MR (GC)	ENZ (GC)	Cora_NC (NC)	Cite_NC (NC)	Cora_LP (LP)	Cite_LP (LP)
b2	**+1.7**	-5.6	-9.1	-1.1	**+4.3**	**+8.4**
b3	-10.3	-27.0	-7.7	-15.5	**+3.1**	**+17.0**
b4	0.0	-13.5	-12.0	-7.0	-0.8	-2.6
s1	**+17.2**	-23.6	**+4.6**	-13.3	-6.1	**+10.2**
s2	-3.4	-7.9	-6.6	-14.1	**+1.9**	**+15.8**
s3	-6.9	-10.1	-3.9	-7.8	**+4.3**	**+6.6**
s4	-10.3	-10.1	-5.9	-8.6	-1.2	**+2.1**
s5	-25.9	-5.6	-6.8	-4.6	-2.7	**+1.2**

Table 4: **Linear Probing Performance Analysis.** Performance improvements across all downstream tasks using linear probing strategy. Values show percentage improvement over from-scratch baseline (b1). Positive values indicate performance gains, negative values indicate degradation. Task types: Graph Classification (GC), Node Classification (NC), Link Prediction (LP). See Appendix B for detailed pre-training task specifications.

Strategy trade-offs. **Linear probing** enables substantial parameter reduction but faces computational bottlenecks, revealing that parameter efficiency doesn't guarantee proportional time savings. **Full fine-tuning** proves effective for molecular domains but creates misalignment in link prediction tasks.

5 Limitations and Future Work

Key limitations include the **forward pass bottleneck** constraining linear probing efficiency gains, suggesting need for architectural innovations (see Appendix A for implementation details). Future work includes: (1) Efficient inference for frozen backbones, (2) Extension to GAT/GraphSAGE architectures, (3) Social network/knowledge graph evaluation, (4) Novel multi-task combinations, (5) Automated selection algorithms.

6 Conclusions

We transform GNN pre-training from trial-and-error to evidence-based deployment through systematic multi-task evaluation. Key findings: (1) **Parameter Efficiency Paradox** - 40.3x parameter reduction yields only 1.06x-1.37x speedup due to forward pass bottleneck, (2) **Heterogeneous Efficiency** - best speedups reach 1.54x (full FT) and 1.70x (linear probing) with domain-scheme specificity, (3) **Selective Performance** - +35.8% peak gains require precise matching, (4) **Predictable Failures** - four systematic modes enable avoidance strategies. Our framework provides empirical matrices and systematic protocols for reliable optimization.

References

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17:2096–2030.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Zhenqin Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031.

Shengchao Liu, Hanchen Nie, Kaixiong Wang, Jian Tang, Shuiwang Xiao, and Jiliang Tang. 2022a. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*.

Yixin Liu, Shirui Yu, Ming Liang, Shirui Pan, Ming Dong, Chen Jin, Philip S Yu, and Chengqi Zhang. 2022b. Graph self-supervised learning: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):5216–5233.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tu dortmund university graph data collection. https://chrsmrrs.github.io/datasets/docs/datasets/.

Mark EJ Newman. 2003. The structure and function of complex networks. *SIAM review*, 45(2):167–256.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, volume 32, pages 8024–8035. Curran Associates, Inc.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1150–1160.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*, 29(3):93–106.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International conference on learning representations*.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. In *International Conference on Learning Representations*.

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 45(6):6828–6847.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International conference on learning representations*.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *Advances in neural information processing systems*, volume 33, pages 5812–5823.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in neural information processing systems*, volume 33, pages 5824–5836.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

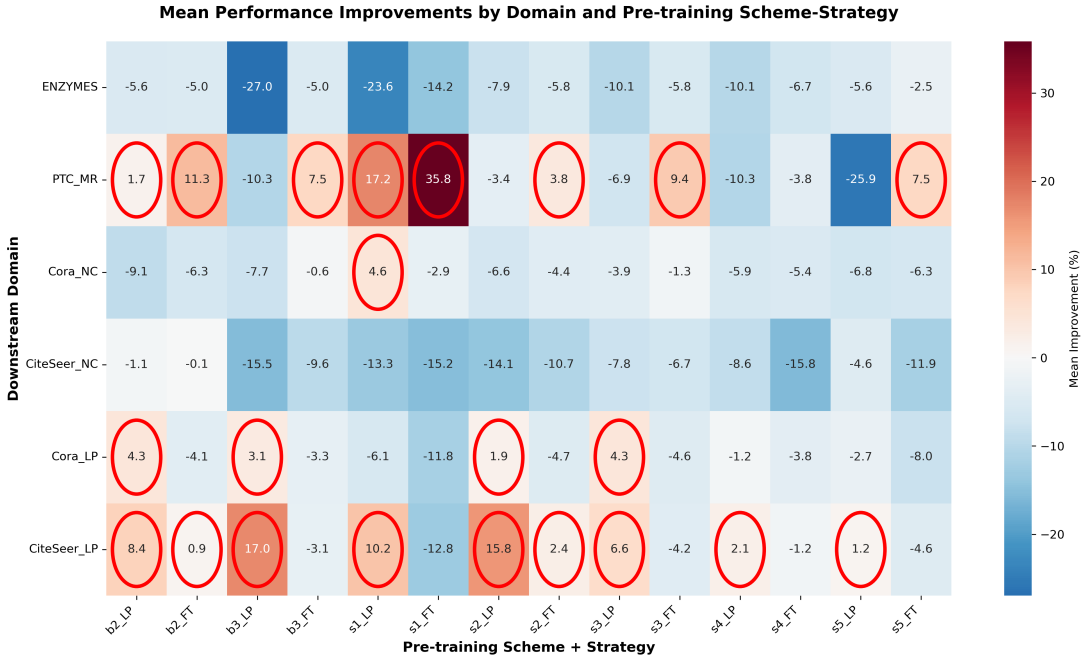

Figure 1: Performance improvement heatmap by domain and pre-training scheme-strategy combinations. Red indicates positive improvements, blue indicates negative transfer. The selective nature of positive transfer is clearly visible, with molecular domains showing strong affinity for specific schemes (s1_FT for PTC_MR), while citation domains favor different combinations (b3_LIN for link prediction tasks).

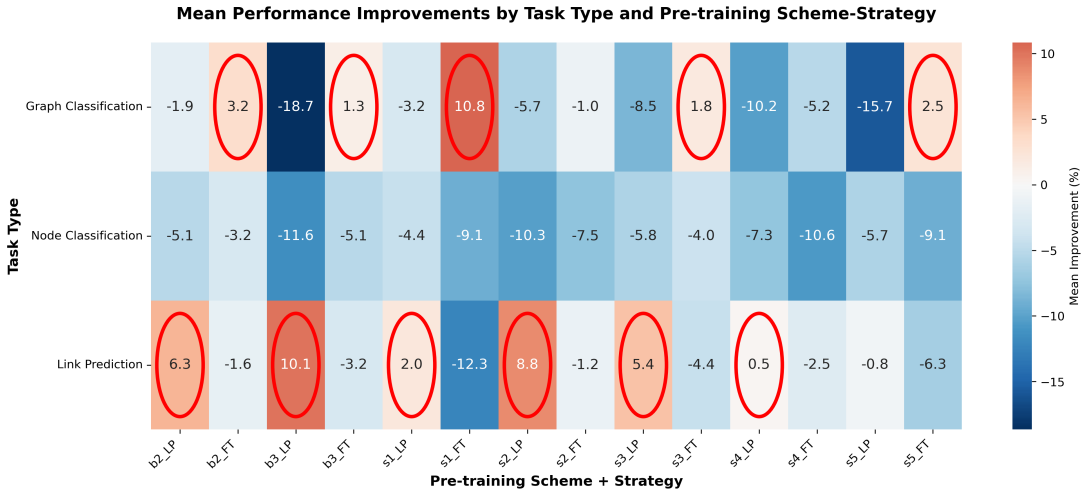

Figure 2: Performance improvement patterns by task type reveal systematic optimization guidelines. Graph Classification (GC) shows high variability with notable peaks (s1_FT), Node Classification (NC) exhibits consistent modest negative transfers, while Link Prediction (LP) demonstrates reliable positive improvements with linear probing strategies (b3_LinearProbe). These patterns enable task-type-first deployment protocols.

A Implementation Details

A.1 Architecture Specifications

GNN Backbone. We use a 5-layer Graph Isomorphism Network (GIN) with hidden dimension 256 throughout all layers. Each GIN layer contains an MLP: $[256 \to 512 \to 256]$ with BatchNorm and ReLU activation. Residual connections are applied: $h_{\text{out}} = \text{GINConv}(h) + h$. Dropout rate is 0.2 after each layer.

Task-Specific Heads. Pre-training heads: Node Feature Masking $[256 \to 256 \to 256]$, Link Prediction $[768 \to 256 \to 1]$ (768 from concatenated edge features $[h_u + h_v + |h_u - h_v|]$), Node Contrastive $[256 \to 256 \to 128]$, Graph Contrastive $[512 \to 256 \to 128]$ (512 from mean+max pooling concatenation), Graph Properties $[256 \to 512 \to 12]$, Domain Classifier $[256 \to 128 \to 4]$ with 0.5 dropout.

A.2 Training Hyperparameters

Pre-training. 50 epochs, batch size 32, gradient clipping (max norm 0.5), patience 50% of total epochs. Task-specific learning rates: Link Prediction 5×10^{-7}, Node Feature Masking 1×10^{-5}, Node/Graph Contrastive 1×10^{-5}, Graph Properties 1×10^{-5}, Domain Adversarial 5×10^{-6}, weight decay 1×10^{-5}.

Fine-tuning. Domain-specific epochs: molecular datasets (ENZYMES, PTC_MR) 100 epochs; citation datasets 200-300 epochs. Batch sizes: molecular 32, node tasks full-batch, link prediction 256. Learning rates: backbone 1×10^{-4} (if unfrozen), heads/encoders 1×10^{-3}. ENZYMES domain-adaptive encoder frozen during fine-tuning.

A.3 Multi-task Learning Components

PCGrad Gradient Surgery. Tasks processed in shuffled order. Conflicting gradients projected: $g_i' = g_i - \frac{g_i \cdot g_j}{||g_j||^2} g_j$ when $g_i \cdot g_j < 0$. Final gradients averaged across tasks (Yu et al., 2020).

Adaptive Loss Balancing. Warmup for 100 steps with equal weights. Inverse magnitude weighting: $w_i = \frac{1}{|L_i|+\varepsilon}$ where $\varepsilon = 10^{-8}$. Weights normalized to sum to 1.

Domain Adversarial Training. Gradient Reversal Layer activates at 40% training progress. Schedule: $\lambda = 2/(1 + \exp(-\gamma p)) - 1$ where $\gamma = 10$, p is progress, maximum $\lambda = 0.01$.

A.4 Pre-training Tasks

Node Feature Masking. Mask 15% of nodes per graph (minimum 3). Learnable mask token initialized $\mathcal{N}(0, 0.01^2)$. MSE reconstruction loss on masked positions.

Graph Augmentation. Node dropping (20% rate, always applied), edge dropping (20% probability, 20% rate), attribute masking (20% probability, 20% rate). Minimum constraints: 3 nodes, 3 edges, 3 features.

Contrastive Learning. Temperature annealing: $T(t) = T_{\text{init}} \cdot (T_{\text{final}}/T_{\text{init}})^{t/T_{\max}}$ where $T_{\text{init}} = 0.5$, $T_{\text{final}} = 0.2$. Hard negative mining: 30% of negatives, minimum 8 hard negatives based on embedding similarity.

A.5 Data Processing

Dataset Splits. Pre-training: 90% train, 10% validation. Fine-tuning: 80% train, 10% validation, 10% test. Stratified splitting for molecular datasets, random for citation node classification domains, and standard for citation link prediction domains.

Graph Properties (12-dimensional). Number of nodes/edges, density, degree statistics (mean, variance, maximum), clustering coefficient, transitivity, connected components count, diameter, degree assortativity, degree centralization.

Feature Scaling. StandardScaler fit on training data. Continuous datasets (ENZYMES) scaled and clipped to [-3, 3].

A.6 Evaluation Protocol

Metrics. Graph/Node Classification: accuracy, F1, precision, recall. Link Prediction: AUC, accuracy, F1, precision, recall. Binary tasks use probabilities from output column 1. Multi-class uses macro-averaged metrics.

Statistical Setup. Seeds: 42, 84, 126. Deterministic training: `torch.backends.cudnn.deterministic = True`. Early stopping on validation AUC (link prediction) or accuracy (classification) with 50% patience.

Hardware. All experiments on CUDA-enabled GPUs. PyTorch 1.12+, PyTorch Geometric 2.0+. Models implemented with AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$).

B Pre-training Task Technical Details

Our framework employs six distinct self-supervised pre-training tasks, each targeting different aspects of graph representation learning. We provide detailed technical descriptions below.

B.1 Node Feature Masking (NFM)

Node Feature Masking follows a mask-and-reconstruct paradigm similar to BERT (Devlin et al., 2019):

1. **Masking Strategy:** For each graph, randomly select 15% of nodes (minimum 3 nodes). Replace selected node features with a learnable mask token $\mathbf{m} \in \mathbb{R}^d$.

2. **Reconstruction:** Apply domain-adaptive encoder first, then mask selected features, then GNN backbone: $\mathbf{H} = \text{GNN}(\mathbf{H}_{\text{masked}}, \mathbf{A})$ where $\mathbf{H}_{\text{masked}}$ contains mask tokens at selected positions

3. **Loss Function:** Minimize MSE between original and reconstructed features:

$$\mathcal{L}_{\text{NFM}} = \frac{1}{|M|} \sum_{v \in M} ||\text{MLP}_{\text{recon}}(\mathbf{h}_v) - \mathbf{h}_v^{\text{orig}}||^2$$

where M is the set of masked nodes and $\text{MLP}_{\text{recon}}$ is a domain-specific reconstruction head.

B.2 Link Prediction (LP)

Link Prediction trains the model to distinguish existing edges from non-existing ones:

1. **Positive Sampling:** Use all edges in the original graph as positive samples.

2. **Negative Sampling:** Generate equal number of negative edges using uniform sampling, ensuring no overlap with existing edges.

3. **Edge Representation:** For edge (u, v), compute:

$$\mathbf{e}_{u,v} = [\mathbf{h}_u + \mathbf{h}_v; \mathbf{h}_u \odot \mathbf{h}_v; |\mathbf{h}_u - \mathbf{h}_v|]$$

where $[;]$ denotes concatenation and $\odot$ is element-wise product.

4. **Loss Function:** Binary cross-entropy over all edge pairs:

$$\mathcal{L}_{\text{LP}} = -\frac{1}{|E|} \sum_{(u,v)} \big[y_{u,v} \log \sigma(f(\mathbf{e}_{u,v})) + (1 - y_{u,v}) \log(1 - \sigma(f(\mathbf{e}_{u,v}))) \big]$$

where $f(\cdot)$ is the MLP predictor.

B.3 Node Contrastive Learning

Node-level contrastive learning using graph augmentations:

1. **Augmentation Strategy:** Create two views of each graph through:
   - Node dropping (20% rate, minimum 3 nodes retained)
   - Edge dropping (20% probability, 20% rate, minimum 3 edges retained)
   - Attribute masking (20% probability, 20% features)

2. **Common Node Identification:** Find nodes present in both augmented views for contrastive pairing.

3. **Projection:** Apply domain-specific projection head: $\mathbf{z}_v = \text{MLP}_{\text{proj}}(\mathbf{h}_v)$

4. **NT-Xent Loss:** For node v with augmented views $\mathbf{z}_v^{(1)}, \mathbf{z}_v^{(2)}$:

$$\mathcal{L}_{\text{node}} = -\frac{1}{2N} \sum_{i=1}^{N} \left[\log \frac{e^{s_{i,i}/\tau}}{\sum_{k \neq i} e^{s_{i,k}/\tau}} + \log \frac{e^{s_{i,i}/\tau}}{\sum_{k \neq i} e^{s_{k,i}/\tau}} \right]$$

where $s(\cdot, \cdot)$ is cosine similarity, τ is temperature, and the loss is computed symmetrically for both view directions.

B.4 Graph Contrastive Learning

Graph-level contrastive learning operates on entire graph representations:

1. **Graph Representation:** Combine mean and max pooling:

$$\mathbf{s}_G = [\text{MeanPool}(\mathbf{H}); \text{MaxPool}(\mathbf{H})]$$

2. **Augmentation:** Same strategy as node contrastive, but applied to generate two graph views.

3. **Loss Function:** Similar NT-Xent formulation but operating on graph-level representations instead of node embeddings.

B.5 Graph Property Prediction

Prediction of structural graph properties for self-supervised learning:

1. **Property Computation:** Extract 12 structural properties per graph:
   - Basic: #nodes, #edges, density, degree statistics
   - Structural: clustering coefficient, transitivity, diameter
   - Advanced: assortativity, degree centralization, connected components
2. **Standardization:** Z-score normalize properties using training set statistics.
3. **Loss Function:** MSE regression over all properties:

$$\mathcal{L}_{\text{prop}} = \frac{1}{|G|} \sum_{G_i} ||\text{MLP}_{\text{prop}}(\mathbf{s}_{G_i}) - \mathbf{p}_{G_i}||^2$$

   where $\mathbf{s}_{G_i}$ is the mean pooling summary of all nodes in graph G_i and $\mathbf{p}_{G_i}$ are the standardized properties of graph G_i.

B.6 Domain Adversarial Training

Domain adversarial training promotes domain-invariant representations:

1. **Gradient Reversal Layer:** Applies identity in forward pass, negates gradients in backward pass with scaling factor λ.
2. **Domain Classification:** Train classifier to predict source domain:

$$\mathcal{L}_{\text{domain}} = -\frac{1}{|G|} \sum_{G_i} \log P(d_i|\text{GRL}(\mathbf{s}_{G_i}; \lambda))$$

   where $\mathbf{s}_{G_i}$ is the mean pooling summary of all nodes in graph G_i, d_i is the true domain label and GRL is gradient reversal.
3. **Schedule:** λ increases from 0 to 0.01 using sigmoid schedule starting at 40% of training.

C Hyperparameter Details

C.1 Pre-training Configuration

- Epochs: 50 with early stopping (patience: 50% of total epochs)
- Batch size: 32 graphs across all domains
- Gradient clipping: max norm 0.5
- Temperature scheduling: 0.5 $\rightarrow$ 0.2 (exponential decay)
- Domain adversarial λ: 0 $\rightarrow$ 0.01 (sigmoid schedule, starting at 40% training)

C.2 Fine-tuning Configuration

Task-specific configurations optimized through preliminary experiments:

Molecular domains (ENZYMES, PTC_MR):

- Epochs: 100, Batch size: 32
- Learning rates: Backbone 1e-4, Domain-adaptive encoder/Head 1e-3

Citation domains (Cora, CiteSeer):

- Node Classification: 200 epochs, full batch
- Link Prediction: 300 epochs, batch size 256
- Learning rates: Backbone 1e-4, Domain-adaptive encoder/Head 1e-3

Linear Probing Setup: Domain-adaptive encoder + classifier head training (GNN backbone frozen). Exception: ENZYMES domain-adaptive encoder also frozen due to in-domain pre-training.