Enhancing Molecular Property Prediction with Auxiliary Learning and Task-Specific Adaptation

Vishal Dey
The Ohio State University, Columbus
dey.78@osu.edu

Xia Ning
The Ohio State University, Columbus
ning.104@osu.edu

Abstract

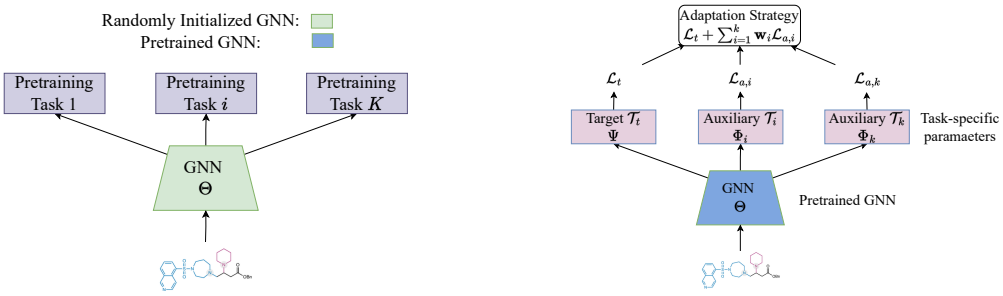
Pretrained Graph Neural Networks (GNNs) have been widely adopted for various molecular property prediction tasks. Despite their ability to capture rich chemical knowledge, traditional finetuning of such pretrained GNNs on the target task can lead to poor generalization. To address this, we explore the adaptation of pretrained GNNs to the target task by jointly training them with multiple auxiliary tasks. This could enable the GNNs to learn both general and task-specific features, which may benefit the target task. However, an effective adaptation strategy needs to determine the relevance of auxiliary tasks with the target task, which poses a major challenge. In this regard, we investigate multiple strategies to adaptively combine task gradients or learn task weights via bi-level optimization. Our experiments with state-of-the-art pretrained GNNs demonstrate the efficacy of our proposed methods, with improvements of up to 8.45% over finetuning. Overall, this suggests that incorporating auxiliary tasks along with target task finetuning can be an effective way to improve the generalizability of pretrained GNNs for molecular property prediction tasks, and thus inspires future research.

1 Introduction

Accurate prediction of molecular properties is pivotal in drug discovery[1], as it accelerates the identification of potential molecules with desired characteristics. Developing computational models for property prediction relies on learning effective representations of molecules[2]. In this regard, Graph Neural Networks (GNNs) have shown impressive results in learning effective representations for molecular property prediction tasks[3–7]. Inspired by the paradigm of pretraining followed by finetuning in large language models (LLMs)[8, 9], molecular GNNs are often pretrained[10] on a large corpus of molecules (for literature review, refer to Appendix A.1), which might encompass irrelevant data for the target property prediction task (e.g., toxicity). This can lead the GNNs to learn features that do not benefit the target task. Consequently, pretrained GNNs are finetuned with the target task to encode task-specific features. However, vanilla finetuning can potentially lead to poor generalization, particularly when dealing with diverse downstream tasks, limited data, and the need to generalize across varying scaffold distributions[11].

To improve generalization, auxiliary learning has recently garnered attention[12, 13], notably in the domains of natural language processing (NLP)[14] and computer vision[15, 16]. Auxiliary learning leverages informative signals from self-supervised tasks on unlabeled data, to improve the performance on the target tasks (for a brief literature review, refer to Appendix A.2). Following this line of work, we explore how to adapt pretrained molecular GNNs by combining widely-used self-supervised tasks with the target task using respective task-specific data (with self-supervised and target task labels). Our contribution lies in a preliminary investigation of multiple adaptation strategies for pretrained molecular GNNs in molecular property prediction using well-established concepts of auxiliary learning. The significance of our contribution lies in addressing the limited benefit of pretrained GNNs[17], and in improving generalizability across a diverse set of downstream tasks with limited data. Overall, our proposed adaptation strategies improved the target task performance by as much as 8.45% over vanilla finetuning. Moreover, our findings indicate that the proposed adaptation

Dey et al., Enhancing Molecular Property Prediction with Auxiliary Learning and Task-Specific Adaptation (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.



(a) Pretraining Stage **(b)** Adaptation Stage

Figure 1: Off-the-shelf available pretrained GNNs are transferred for target task-specific adaptation.

strategies are particularly effective in tasks with limited labeled data, which is a common challenge in molecular property prediction tasks.

2 Methods

Motivated by the success of continued pretraining and adaptation in pretrained Large Language Models (LLMs) [18–20], we investigate adaptation of off-the-shelf pretrained molecular GNNs (e.g., Supervised+ContextPred[10] denoted as `Sup-C`) to target molecular property prediction tasks. Via such an adaptation, we aim to leverage existing self-supervised (SSL) tasks designed for molecular GNNs and transfer learned knowledge from such tasks to the target task. We employ the existing SSL tasks typically used in molecular pretraining such as masked atom prediction (AM), edge prediction (EP), context prediction (CP), graph infomax (IG), and motif prediction (MP) (detailed in Appendix B.1). We refer to these tasks as auxiliary tasks. Intuitively, these auxiliary tasks can potentially capture diverse chemical semantics and rich structural patterns at varying granularities. By utilizing SSL objectives on target task-specific data, auxiliary tasks augment the pretrained GNNs with richer representations. Such representations, in turn, can improve the generalizability of the target property prediction task. Henceforth, the term "GNN" refers to off-the-shelf pretrained molecular GNN.

Figure 1 presents an overview of the adaptation setup. Formally, we adapt a GNN with parameters Θ to optimize the performance on the target task $\mathcal{T}_t$. We achieve this by jointly training $\mathcal{T}_t$ with auxiliary tasks $\{\mathcal{T}_a\}_{i=1}^{k}$:

$$\min_{\Theta,\Psi,\Phi_{i\in\{1..k\}}} \mathcal{L}_t + \sum_{i=1}^{k} \mathbf{w}_i \mathcal{L}_{a,i},$$

where $\mathcal{L}_t$ and $\mathcal{L}_{a,i}$ denote the target task loss and i-th auxiliary task loss, respectively, and Ψ and $\Phi_{i\in\{1,\ldots,k\}}$ denotes task-specific learnable parameters for the target and k-auxiliary tasks, respectively, and $\mathbf{w}$ indicates the influence of the auxiliary tasks on the target task. Through the above optimization, all the parameters are simultaneously updated in an end-to-end manner. Note that the above optimization does not optimize $\mathbf{w}$– we will introduce an approach that can additionally learn $\mathbf{w}$. In fact, the key to effective adaptation lies in accurately determining $\mathbf{w}$, such that the combined task gradients can backpropagate relevant training signals to the shared GNN as follows:

$$\Theta^{(t+1)} := \Theta^{(t)} - \alpha \left(\mathbf{g}_t + \sum\nolimits_{i=1}^{k} \mathbf{w}_i \mathbf{g}_{a,i} \right),$$

where $\mathbf{g}_t = \nabla_\Theta \mathcal{L}_t$, and $\mathbf{g}_{a,i} = \nabla_\Theta \mathcal{L}_{a,i}$ denote the gradients updating Θ from the target and i-th auxiliary task, respectively, and α denotes the learning rate. We experiment with multiple strategies to adaptively combine task gradients, or learn adaptive $\mathbf{w}$, as opposed to using fixed weights or conducting expensive grid-search to explore all possible $\mathbf{w}$.

2.1 Gradient Cosine Similarity (`GCS`)

One such strategy to meaningfully combine task gradients is based on gradient cosine similarity (GCS). Intuitively, GCS measures the alignment between task gradients during training, providing insights into the relatedness of auxiliary tasks with the target task. High GCS indicates that the auxiliary tasks provide complementary information and thus, can benefit the target task. Conversely, low GCS indicates potential orthogonality or even conflict between tasks. Thus, GCS can naturally be used to quantify the relatedness of auxiliary tasks with the target task over the course of training. We compute GCS and update Θ as:

$$\Theta^{(t+1)} := \Theta^{(t)} - \alpha \left(\mathbf{g}_t + \sum\nolimits_{i=1}^{k} \max\left(0, \cos\left(\mathbf{g}_t, \mathbf{g}_{a,i}\right)\right) \mathbf{g}_{a,i} \right),$$

where, in essence, we drop the tasks with conflicting gradients (i.e., with negative GCS). We adopt this strategy from the weighted version presented in Du et al.[21].

2.2 Gradient Scaling (GNS)

Alternatively, we adopt a simpler strategy of gradient scaling to adjust the influence of auxiliary tasks with respect to the target task. Our preliminary experiments as presented in Figure 3 revealed significant differences in the scales of the task gradient norms, and thus requiring careful adjustments. This is because if the gradient of an auxiliary task is much larger than that of the target task, Θ updates will be most dominated by such auxiliary tasks, thereby potentially resulting in worse target performance. On the other hand, if the gradient of an auxiliary task is relatively smaller, the training signals from such auxiliary tasks will be too weak to encode any relevant features in Θ. Thus, following [22, 23], we use a simple gradient scaling to dynamically adjust the influence of auxiliary tasks during updates of Θ as:

$$\Theta^{(t+1)} := \Theta^{(t)} - \alpha \left(\mathbf{g}_t + \sum_{i=1}^{k} \frac{||\mathbf{g}_t||}{||\mathbf{g}_{a,i}||} \mathbf{g}_{a,i} \right),$$

where $||\cdot||$ denotes the L^2 norm.

2.3 Bi-Level Optimization (BLO)

Unlike the previous approaches that combine task gradients, BLO learns $\mathbf{w}$ in an end-to-end manner such that the GNN generalizes well to the target task. Formally, we learn $\mathbf{w}$ that minimizes the target validation loss while ensuring that the GNN is optimized with a balanced combination of losses:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})}(\Theta^*(\mathbf{w})), \qquad \text{s.t.} \qquad \Theta^*(\mathbf{w}) = \arg\min_{\Theta} \mathcal{L}_f(\Theta, \mathbf{w})$$

where, $\mathcal{L}_f = \mathcal{L}_t + \sum_{i=1}^{k} \mathbf{w}_i \mathcal{L}_{a,i}$ is the combined loss on the training set, and $\mathcal{L}_t^{(\mathcal{A})}$ is the loss on the target task computed with a held-out auxiliary dataset $\mathcal{A}$, and $\Theta^*(\mathbf{w})$ is the best-response of Θ with current $\mathbf{w}$. This formulation is a bi-level optimization problem: updating $\mathbf{w}$ in the upper-level optimization requires computing $\nabla_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})} = \nabla_{\Theta} \mathcal{L}_t^{(\mathcal{A})} \cdot \nabla_{\mathbf{w}} \Theta^*$, where the latter gradient requires back-propagation through the inner-level optimization of Θ. Following [24], we leverage the Implicit Function Theorem (IFT) to compute $\nabla_{\mathbf{w}} \Theta^* = -(\nabla^2_{\Theta} \mathcal{L}_f)^{-1} \cdot \nabla_{\mathbf{w}} \nabla_{\Theta} \mathcal{L}_f$. Intuitively, IFT allows us to evaluate the $\nabla_{\mathbf{w}} \Theta^*$ locally around the approximate best-response Θ^*. Using the above, we can compute the gradients $\nabla_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})}$ as:

$$\nabla_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})}(\Theta^*(\mathbf{w})) = \nabla_{\Theta} \mathcal{L}_t^{(\mathcal{A})} \cdot \nabla_{\mathbf{w}} \Theta^*(\mathbf{w}) = -\nabla_{\Theta} \mathcal{L}_t^{(\mathcal{A})} \cdot (\nabla^2_{\Theta} \mathcal{L}_f)^{-1} \cdot \nabla_{\mathbf{w}} \nabla_{\Theta} \mathcal{L}_f.$$

To compute the Hessian inverse and vector products efficiently, we use the iterative algorithm by Lorraine et al.[24], which is summarized in Algorithm 2. Intuitively, it uses a Neumann series expansion to approximate the Hessian inverse with unrolling differentiation for M steps around locally approximate best-response Θ^*. Following [25], in practice, we don't train Θ till convergence (i.e., Θ^* such that $\nabla_{\Theta} \mathcal{L}_f = 0$). Instead, we approximate Θ^* by simultaneously training both Θ and $\mathbf{w}$, and alternately optimizing $\mathbf{w}$ for every r updates of Θ. We described the above process in Algorithm 1. We refer the readers to [24] for theoretical considerations on approximations and convergence. Note that we use 20% of the training set as $\mathcal{A}$ instead of using the validation set to avoid data leakage and unfair comparison with baselines. Optimizing $\mathbf{w}$ on a held-out $\mathcal{A}$ rather than on the training set aligns with the goal of improving target task generalizability.

Algorithm 1 Learning Task Weights with BLO

Input: N, r, α
Initialize $\mathbf{w}$ with $1/k$, Θ from pretrained GNN, Ψ and Φ with default Xavier initializer
for *epoch* from 1 to N **do**
  Compute $\mathcal{L}_f = \mathcal{L}_t + \sum_{i=1}^{k} \mathbf{w}_i \mathcal{L}_{a,i}$
  $\Theta \leftarrow \Theta - \alpha \nabla_{\Theta} \mathcal{L}_f$, $\Phi \leftarrow \Phi - \alpha \nabla_{\Phi} \mathcal{L}_a$, $\Psi \leftarrow \Psi - \alpha \nabla_{\Psi} \mathcal{L}_t$
  if $epoch \% r == 0$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} - \nabla_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})}(\Theta(\mathbf{w}))$ ▷ Algorithm 2
  end if
end for
Return Θ, $\mathbf{w}$

Algorithm 2 Computing $\nabla_{\mathbf{w}} \mathcal{L}_t^{(\mathcal{A})}(\Theta(\mathbf{w}))$

Input: $\mathcal{L}_f$, $\mathcal{L}_t^{(\mathcal{A})}$, current $\mathbf{w}$, Θ from Algorithm 1, M, β
Initialize $p = q = \nabla_{\Theta} \mathcal{L}_t^{(\mathcal{A})}|_{(\mathbf{w},\Theta)}$ ▷ Hessian inverse approximation
for j from 1 to M **do**
  $p = p - \beta p \nabla^2_{\Theta} \mathcal{L}_f$
  $q = q + p$
end for
Return $-q \nabla_{\mathbf{w}} \nabla_{\Theta} \mathcal{L}_f|_{(\mathbf{w},\Theta)}$

We refer the readers to Appendix B.3 for implementation details.

Table 1: Test ROC-AUC using $\mathcal{T}_a$={AM,CP,EP,IG,MP} and Sup-C

Method	SIDER	ClinTox	BACE	BBBP	Tox21	ToxCast	HIV	MUV
FT	**62.05** (0.40)	**71.36** (0.70)	82.68 (1.14)	67.42 (0.66)	**77.55** (0.07)	**66.18** (0.14)	78.60 (0.80)	81.25 (2.17)
GTOT	61.98 (0.12)	**71.48** (0.77)	82.15 (2.20)	**71.34** (0.76)	**77.85** (0.52)	64.90 (0.72)	**80.03** (0.19)	82.38 (1.52)
MTL	56.39 (0.29)	55.69 (2.65)	77.11 (5.11)	64.89 (0.30)	74.31 (0.24)	64.32 (0.25)	76.79 (0.59)	80.81 (0.74)
GCS	59.52 (1.08)	63.10 (2.46)	**85.49** (0.40)	71.23 (0.26)	74.84 (0.36)	65.96 (0.14)	76.69 (0.20)	75.20 (2.42)
GNS	**62.14** (0.37)	68.07 (1.58)	**84.76** (0.30)	**71.60** (0.88)	76.44 (0.16)	**66.24** (0.11)	77.87 (0.13)	**83.77** (1.25)
BLO	58.09 (0.50)	65.33 (2.23)	84.28 (3.51)	69.18 (0.46)	76.04 (0.46)	65.93 (0.30)	**78.62** (0.48)	**82.74** (0.32)

We report the mean (and standard deviation) over 3 different seeds with scaffold splitting. Best- and second best-performing models are in **bold** and **bold**. Tasks are presented in increasing order of size. Error plots are in Figure 5.

3 Experiments

We compare our adaptation strategies with traditional finetuning (FT), one of the state-of-the-art regularization-based finetuning with optimal transport (GTOT)[26], and vanilla multi-task learning (MTL) that assigns equal weights to all auxiliary tasks. We report the overall performance in Table 1. We observed that vanilla MTL leads to drastic negative transfer, especially in smaller datasets. On the contrary, all adaptation strategies perform better than MTL and can help mitigate negative transfer on such datasets. Among such strategies, both GCS and GNS consistently outperform BLO. This can be due to the noisy task gradients which can lead to poor approximation of hyper-gradients. In contrast, GCS can offer robustness to noisy gradients since it relies on the alignment of their directions instead of magnitudes, and GNS is more robust to noisy gradients since it adjusts the scale of gradient norms with respect to the target task. Overall, GNS outperforms FT and GTOT on smaller datasets (except ClinTox), while achieving competitive performance on larger ones. Specifically, GNS improved ROC-AUC by 2.52% and 6.20% over FT in BACE and BBBP, and by 3.18% over GTOT in BACE. Furthermore, while analyzing gradient similarities of auxiliary tasks with the target task (Figure 4), we hypothesize that AM, IG, and MP may benefit the target task better than the other auxiliary tasks.

Table 2: Test ROC-AUC using $\mathcal{T}_a$={AM,IG,MP} and Sup-C

Method	SIDER	ClinTox	BACE	BBBP	Tox21	ToxCast	HIV	MUV
FT	62.05 (0.40)	**71.36** (0.70)	82.68 (1.14)	67.42 (0.66)	77.55 (0.07)	66.18 (0.14)	78.60 (0.80)	81.25 (2.17)
GTOT	61.98 (0.12)	**71.48** (0.77)	82.15 (2.20)	71.34 (0.76)	**77.85** (0.52)	64.90 (0.72)	**80.03** (0.19)	82.19 (1.52)
MTL	58.80 (0.38)	60.82 (1.58)	83.67 (0.42)	**72.50** (0.25)	75.22 (0.28)	65.05 (0.27)	78.18 (1.31)	81.26 (2.33)
GCS	**63.10** (0.17)	64.84 (2.21)	84.94 (0.42)	68.53 (5.93)	77.41 (0.13)	**66.63** (0.14)	79.23 (0.15)	**82.56** (1.42)
GNS	62.46 (0.19)	65.44 (1.01)	**85.02** (0.23)	72.38 (0.26)	76.62 (0.30)	65.96 (0.14)	79.18 (0.28)	**82.22** (0.11)
BLO	**63.46** (0.33)	63.06 (0.36)	**85.22** (0.36)	**73.12** (0.50)	**77.94** (0.35)	**66.64** (0.16)	**80.08** (0.44)	81.83 (0.91)

Best- and second best-performing models are in **bold** and **bold**. Error plots are in Figure 6.

Therefore, we compare our adaptation strategies with baselines using only AM, EP, and CP as auxiliary tasks, and report the results in Table 2. With fewer tasks in this setup, MTL exhibits diminished negative transfer compared to the previous setup. Compared to the previous setup, all adaptation strategies demonstrate better performance across almost all datasets. Notably, BLO outperforms all other methods across all datasets except ClinTox and MUV. Specifically, BLO improved ROC-AUC by 2.27%, 2.83% and 8.45% compared to FT, and by 2.39%, 3.74% and 2.50% compared to GTOT, in SIDER, BACE, and BBBP, respectively. To summarize, for small-scale datasets except ClinTox, GNS outperforms FT and GTOT when all auxiliary tasks are used, and BLO largely outperforms all other methods when a subset of informative auxiliary tasks is used. On larger datasets, all methods achieve similar performance. This suggests that with limited labeled data, FT struggles to capture relevant task-specific features. In contrast, our adaptation strategies enable learning more generalizable features from auxiliary tasks which benefits the target task, even with limited labeled data – which addresses a crucial challenge in molecular property prediction tasks. Additional results with another pretrained GNN, GraphMVP-C [27], are presented in Tables 4 and 5.

4 Conclusion and Future Work

In this study, we explored three adaptation strategies (GCS, GNS and BLO) to improve the performance of pretrained GNNs on molecular property prediction tasks. Our experiments demonstrate that GNS and BLO outperform all other approaches on smaller datasets (except ClinTox), depending on the choice of auxiliary tasks. Overall, the results of this study suggest that the adaptation of pretrained GNNs can be a promising direction to boost target task performance, especially with limited labeled data. In future work, we will explore other adaptation strategies to alleviate noisy gradients and to improve task selection with sparser task weights. We will further investigate the benefit of adapting GNNs to diverse downstream molecular regression tasks.

Acknowledgements

This research has been supported in part by the National Science Foundation grant support IIS-2133650 (X.N.). Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

References

[1] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020. 1

[2] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1): 1–22, 2020. 1

[3] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59 (8):3370–3388, 2019. 1

[4] Johannes Gasteiger, Chandan Yeshwanth, and Stephan Günnemann. Directional message passing on molecular graphs via synthetic coordinates. *Advances in Neural Information Processing Systems*, 34:15421–15433, 2021.

[5] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3): 279–287, 2022. 7

[6] Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, 4(2):127–134, 2022.

[7] Zhichun Guo, Kehan Guo, Bozhao Nan, Yijun Tian, Roshni G. Iyer, Yihong Ma, Olaf Wiest, Xiangliang Zhang, Wei Wang, Chuxu Zhang, and Nitesh V. Chawla. Graph-based molecular representation learning. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6638–6646. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/744. URL https://doi.org/10.24963/ijcai.2023/744. Survey Track. 1

[8] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 1

[9] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 1

[10] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2019. 1, 2, 7, 8, 9

[11] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 1, 9

[12] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018. 1

[13] Shikun Liu, Andrew Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[14] Lucio M Dery, Paul Michel, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Aang: Automating auxiliary learning. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 8

[15] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018. 1

[16] Hong Chen, Xin Wang, Chaoyu Guan, Yue Liu, and Wenwu Zhu. Auxiliary learning with joint task and data scheduling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 3634–3647. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/chen22y.html. 1, 8

[17] Ruoxi Sun, Hanjun Dai, and Adams Wei Yu. Does gnn pretraining help molecular representation? *Advances in Neural Information Processing Systems*, 35:12096–12109, 2022. 1, 8

[18] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, 2020. 2

[19] Lucio M Dery, Paul Michel, Ameet Talwalkar, and Graham Neubig. Should we be pre-training? an argument for end-task aware training as an alternative. In *International Conference on Learning Representations*, 2021.

[20] Eugene Yang, Suraj Nair, Ramraj Chandradevan, Rebecca Iglesias-Flores, and Douglas W Oard. C3: Continued pretraining with contrastive weak supervision for cross language ad-hoc retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2512, 2022. 2

[21] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018. 3, 8

[22] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 3

[23] Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. Metabalance: improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. In *Proceedings of the ACM Web Conference 2022*, pages 2205–2215, 2022. 3

[24] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pages 1540–1552. PMLR, 2020. 3

[25] Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. In *International Conference on Learning Representations*, 2020. 3, 8

[26] Jiying Zhang, Xi Xiao, Long-Kai Huang, Yu Rong, and Yatao Bian. Fine-tuning graph neural networks via graph topology induced optimal transport. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3730–3736. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/518. URL https://doi.org/10.24963/ijcai.2022/518. Main Track. 4, 8

[27] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. In *International Conference on Learning Representations*, 2021. 4, 7, 9

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 7

[29] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020. 7

[30] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020. 7

[31] Sheheryar Zaidi, Michael Schaarschmidt, James Martens, Hyunjik Kim, Yee Whye Teh, Alvaro Sanchez-Gonzalez, Peter Battaglia, Razvan Pascanu, and Jonathan Godwin. Pre-training via denoising for molecular property prediction. *arXiv preprint arXiv:2206.00133*, 2022. 7

[32] Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. Pre-training graph neural networks for molecular representations: retrospect and prospect. In *ICML 2022 2nd AI for Science Workshop*, 2022. 7

[33] Jun Xia, Jiangbin Zheng, Cheng Tan, Ge Wang, and Stan Z Li. Towards effective and generalizable fine-tuning for pre-trained molecular graph models. *bioRxiv*, pages 2022–02, 2022. 7, 8

[34] Hanchen Wang, Shengchao Liu, Jean Kaddour, Qi Liu, Jian Tang, Matt Kusner, and Joan Lasenby. Evaluating self-supervised learned molecular graphs. In *ICML 2022 2nd AI for Science Workshop*, 2022. 8

[35] Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pages 4965–4974. PMLR, 2018. 8

[36] Anish Nediyanchath, Periyasamy Paramasivam, and Promod Yenigalla. Multi-head attention for speech emotion recognition with auxiliary learning of gender recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7179–7183. IEEE, 2020.

[37] Yohan Lee. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303, 2021. 8

[38] Baifeng Shi, Judy Hoffman, Kate Saenko, Trevor Darrell, and Huijuan Xu. Auxiliary task reweighting for minimum-data learning. *Advances in Neural Information Processing Systems*, 33:7148–7160, 2020. 8

[39] Lucio M Dery, Yann Dauphin, and David Grangier. Auxiliary task update decomposition: The good, the bad and the neutral. *arXiv preprint arXiv:2108.11346*, 2021. 8

[40] Shuxiao Chen, Koby Crammer, Hangfeng He, Dan Roth, and Weijie J Su. Weighted training for cross-task learning. In *International Conference on Learning Representations*, 2021. 8

[41] Aviv Shamsian, Aviv Navon, Neta Glazer, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Auxiliary learning as an asymmetric bargaining game. *arXiv preprint arXiv:2301.13501*, 2023. 8

[42] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019. 8

[43] RDKit, online. RDKit: Open-source cheminformatics. `http://www.rdkit.org`. 8

[44] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation, 2023. URL `https://arxiv.org/abs/2306.03536`. 10

A Related Work

A.1 Pretraining and Finetuning GNNs

Pretraining followed by finetuning is widely used to leverage knowledge gained from related tasks and to improve model generalization. Typically, it involves training a model on large-scale data with self-supervised or supervised tasks, and then finetuning it on a small-scale labeled data. Inspired by the success of pretraining and finetuning paradigm in LLMs[28, 29], researchers have extended it to molecular GNNs[5, 10, 27, 30, 31]. In this regard, researchers have designed a number of self-supervised tasks as pretraining tasks that focus on capturing diverse chemical rules, connectivities, and patterns at varying granularities: node-, subgraph- and graph-level. We refer the readers to Xia et al[32] for an in-depth review of pretrained GNNs. Despite the promise of various pretrained GNNs that can capture diverse chemical knowledge, effectively extracting relevant knowledge through vanilla finetuning is non-trivial. Specifically, such finetuning often leads to overfitting as demonstrated in [33]. Unlike pretrained models in natural language processing (NLP) and computer vision, pretrained GNNs do not consistently demonstrate substantial improvements.

Furthermore, there exists a notable research gap in determining what self-supervised tasks can better benefit the downstream target tasks. In fact, prior studies in pretraining molecular GNNs

mostly leverage one or two self-supervised task(s), thereby resulting in a plethora of multiple pretrained GNNs. Interestingly, such pretrained GNNs capture different knowledge[34] and excel in different downstream molecular property prediction tasks[17]. Additionally, Sun et al.[17] recently demonstrated that self-supervised graph pretraining does not consistently/significantly outperform non-pretraining methods across various settings. Overall, although pretrained GNNs hold promise for molecular property prediction, their benefit over non-pretrained models seems limited possibly due to the discrepancy in learning objectives between the pretraining SSL tasks and downstream target task. To address this, some recent attempts[26, 33] to fine-tune pretrained GNNs have largely relied on existing ideas like regularization or weight constraints during fine-tuning. In contrast, our proposed approaches leverage auxiliary tasks to learn generalizable knowledge and prevent overfitting to the training set.

A.2 Knowledge Transfer with Auxiliary Learning

Knowledge transfer through auxiliary learning have demonstrated its effectiveness across a spectrum of domains [35–37]. This paradigm, distinct from multi-task learning, aims to optimize the target task's performance while leveraging auxiliary tasks to bolster generalization[38]. Prior research in other domains has developed multiple methods to automatically learn task weights, such as using gradient similarity[21, 39], using parameterized auxiliary network[14, 25], using bi-level optimization and implicit differentiation[16, 25], minimizing distances between task embeddings[40], or from the perspective of Nash equilibrium[41]. However, it is crucial to note that the application of auxiliary learning for adapting molecular GNNs to target tasks, particularly in the context of molecular property prediction, remains an underexplored area. In this study, we adopt and explore gradient similarity, gradient scaling, and bi-level optimization strategies.

B Experimental Details

B.1 On Auxiliary Tasks

We describe the auxiliary tasks and share key insights behind using them:

- Masked Atom Prediction (AM): AM involves predicting the identity of masked atoms within a molecular graph. It helps the GNN to learn the local chemical context and relationships between atoms and bonds, which are crucial for understanding molecular structure and function. The embedding out of GNN is fed to a linear classifier to predict the atom type of masked atoms.
- Edge Prediction (EP): EP focuses on predicting the presence or absence of bonds (edges) between pairs of atoms in a molecular graph. It helps the GNN to capture essential local structural information, including connectivity and spatial arrangement of atoms within molecules. Following existing design[17], the dot product of node embeddings is used to predict the existence of a bond.
- Context Prediction (CP): CP requires the model to predict neighboring graph structures (context) based on an anchor structure. This aids the GNN in distinguishing molecular contexts, enabling the model to capture subgraph-level information. The setup of Hu et al.[10] is followed to extract and distinguish positive and negative subgraph contexts.
- Graph Infomax (IG): IG maximizes the mutual information between local (node) and global (subgraph) representations. This helps the GNN to capture structural patterns, allowing it to understand how atoms form functional groups and larger molecular substructures. The existing setup[42] is followed to train a discriminator model that distinguishes between node embeddings from the same molecular graph and those from a different graph.
- Motif Prediction (MP): MP focuses on predicting the presence of specific recurring substructures (motifs) within a molecule. It helps the GNN to identify structural motifs indicative of chemical properties or functions. This task is formulated as a multi-label binary classification problem with each of 85 motifs[1] extracted from RDKIT[43] as labels.

Each of these tasks focuses on different aspects of molecular graphs, such as local connectivity, spatial arrangement, contextual information, hierarchical organization, and recurring structural patterns. In essence, these tasks are designed to equip the model with a richer understanding of molecular

[1] http://rdkit.org/docs/source/rdkit.Chem.Fragments.html

structures, ultimately improving its ability to generalize and make accurate predictions. Note that designing auxiliary tasks is beyond the scope of this study.

B.2 Dataset Overview

We perform our adaptation experiments on 8 benchmark classification datasets from MoleculeNet[11]. In this section, we give a brief overview and provide preliminary statistics of these datasets.

- BBBP: measures whether a molecule permeates the blood-brain barrier.
- BACE: measures whether a molecule inhibit the β-secretase 1 (BACE-1) enzyme.
- ClinTox: contains toxicity labels for clinical drugs, facilitating the assessment of drug safety profiles across various targets. It is important to note that these labels reflect both FDA approval outcomes and clinical trial failures due to toxicity. Such outcomes are determined by not just the molecular structures of the drugs. but also by external factors such as genetic predispositions, evaluation methodologies, and environmental conditions. This complexity can make methodological comparisons challenging.
- HIV: measures whether a molecule can prevent antiviral activity against the HIV virus.
- MUV: compiled and refined from PubChem bioassays, evaluating compound activity across multiple targets.
- Tox21: measures toxicity across a range of biological pathways used in the 2014 Tox21 challenge.
- ToxCast: measures compound toxicity across a range of biological systems.

Table 3: Overview of benchmark molecular property prediction datasets

Dataset	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE
No. mols	2,039	7.831	8,575	1,427	1,478	93,087	41,127	1,513
No. tasks	1	12	617	27	2	17	1	1
Avg. atoms	24.06	18.57	18.78	33.64	26.16	24.23	25.51	34.09
Avg. diameter	11.32	9.62	9.49	14.14	12.39	12.79	11.98	15.22

B.3 Reproducibility and Implementation Details

For all our experiments, we use the official publicly available checkpoints of two pretraining GIN models: supervised_contextpred[10] (denoted as `Sup-C`) and `GraphMVP-C` [27]. Following the prior line of research[10, 27], we use scaffold-split for the downstream target tasks, and use the same atom and bond features as in `GraphMVP-C` and other related works. All experimental details for the `FT` baseline follow the GraphMVP setup, which is identical to the `GTOT` finetuning setup. Specifically, for `FT` and adaptation approaches, we initialized a linear projection layer on top of the pretrained GNN as the target task classifier. In all approaches, both the pretrained GNN and task-specific layers are trainable. For `FT` and adaptation methods, we train the models for 100 epochs with Adam optimizer with an initial learning rate α of 0.001, we use a batch size of 256, an embedding dimension of 300, and a dropout probability of 0.5 for the GNN module. For `GTOT` experiments, we use the optimal hyper-parameters provided for each dataset, when finetuned on `Sup-C`. For `MTL` experiments, we assign equal weights to all auxiliary tasks. For `BLO` experiments, we use $M = 3$ in algorithm 2, update $\mathbf{w}$ every $r = \{5, 10, 20\}$ updates of Θ, and use Adam optimizer with learning rate β of 0.001 to update $\mathbf{w}$. The code is available at `https://github.com/vishaldeyiiest/GraphTA`.

B.4 Additional Results with `GraphMVP-C` as the pretrained GNN

We compare our adaptation strategies with finetuning methods (`FT` and `GTOT`) and vanilla multi-task learning (`MTL`) that assigns equal weights to all auxiliary tasks. We present the results in Table 4. We observed that Vanilla `MTL` leads to drastic negative transfer. Our experiments show that adaptation strategies can help mitigate negative transfer to some extent. Similar to our observations for `Sup-C` in Table 2, both `GCS` and `GNS` consistently outperform `BLO`. However, the adaptation methods perform worse or achieve competitive performance compared to `FT` and `GTOT`.

Additionally, we observed that IG and MP tasks were less prone to gradient conflicts with the target task, unlike other auxiliary tasks (as demonstrated in Figure 2). Thus, IG and MP may benefit the

Table 4: Test ROC-AUC using $\mathcal{T}_a$={AM, EP, CP, IG, MP} and GraphMVP-C.

Method	SIDER	ClinTox	BACE	BBBP	Tox21	ToxCast	HIV	MUV
FT	**62.73**(0.40)	68.52(7.19)	**80.07**(0.98)	**70.92**(0.71)	73.71(0.61)	**64.13**(0.68)	**75.01**(1.25)	72.45(2.87)
GTOT	**60.83**(0.55)	**73.17**(5.63)	78.36(2.30)	62.09(1.70)	**74.30**(0.59)	**64.67**(0.11)	74.78(1.94)	72.63(0.83)
MTL	51.31(1.47)	52.84(5.23)	57.20(4.83)	51.08(2.02)	60.53(1.78)	55.02(0.45)	72.12(3.15)	72.26(7.09)
GCS	57.97(0.54)	**69.20**(1.59)	78.00(3.22)	68.27(1.06)	74.14(0.29)	60.87(0.63)	**74.80**(1.19)	73.49(0.21)
GNS	58.37(2.71)	65.91(7.41)	**80.98**(0.61)	**71.82**(0.41)	**74.87**(0.35)	63.97(0.16)	72.88(1.15)	**76.20**(2.28)
BLO	58.89(1.21)	67.26(1.60)	74.88(5.21)	68.18(0.64)	73.63(0.29)	61.93(0.87)	74.29(1.74)	**74.69**(1.34)

We report the mean (and standard deviation) over 3 different seeds with scaffold splitting. Best- and second best-performing models are in **bold** and **bold**. Tasks are presented in increasing order of data size.

target task better than the other auxiliary tasks. This could be likely because 1) these tasks are more closely related to the target task, or 2) these tasks are more challenging than EP and CP tasks (which only capture some local patterns), thereby forcing the GNNs to learn more robust and generalizable features. Therefore, we compare different adaptation strategies using only IG and MP as auxiliary tasks in Table 5. Overall, compared to the previous setup, all adaptation methods exhibit better performance, especially on small-scale datasets. However, BLO still consistently performs worse than GCS and GNS, similar to the previous setup. Overall, GCS achieved an average relative improvement of 1.18% over FT, with notable improvements of 3.79% and 2.73% in ClinTox and HIV, respectively. Interestingly, we observed that our adaptation strategies also work relatively better than FT and GTOT for very large-scale datasets such as HIV and MUV.

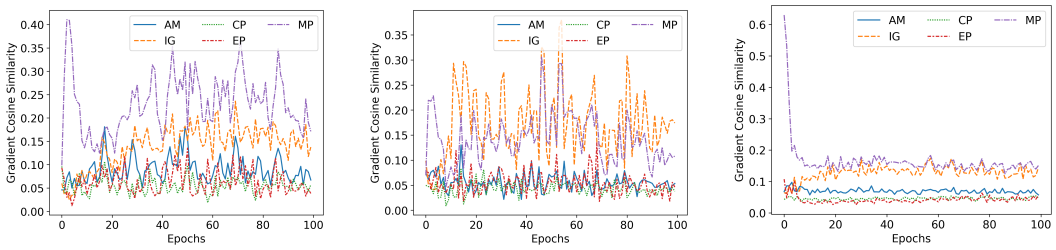
Table 5: Test ROC-AUC using $\mathcal{T}_a$={IG, MP} and GraphMVP-C.

Method	SIDER	ClinTox	BACE	BBBP	Tox21	ToxCast	HIV	MUV
FT	**62.73**(0.40)	68.52(7.19)	80.07(0.98)	70.92(0.71)	73.71(0.61)	**64.13**(0.68)	75.01(1.25)	72.45(2.87)
GTOT	60.83(0.55)	**73.17**(5.63)	78.36(2.30)	62.09(1.70)	74.30(0.59)	**64.67**(0.11)	74.88(1.94)	72.63(0.83)
MTL	60.29(1.87)	61.76(4.14)	78.41(0.50)	72.05(1.67)	73.75(0.40)	61.59(1.07)	75.19(0.53)	**74.18**(4.59)
GCS	**63.34**(0.34)	**71.43**(2.53)	**80.48**(1.73)	71.04(2.33)	**74.73**(0.16)	63.28(1.49)	**77.06**(0.86)	72.42(2.43)
GNS	61.62(1.44)	67.82(2.57)	**81.02**(1.42)	**72.26**(0.45)	**75.12**(0.43)	62.98(0.30)	**76.61**(1.50)	**76.94**(0.18)
BLO	60.94(0.72)	68.54(0.53)	79.92(0.57)	**72.71**(2.01)	73.86(0.63)	62.29(0.60)	74.75(0.76)	73.94(1.27)

Best- and second best-performing models are in **bold** and **bold**.

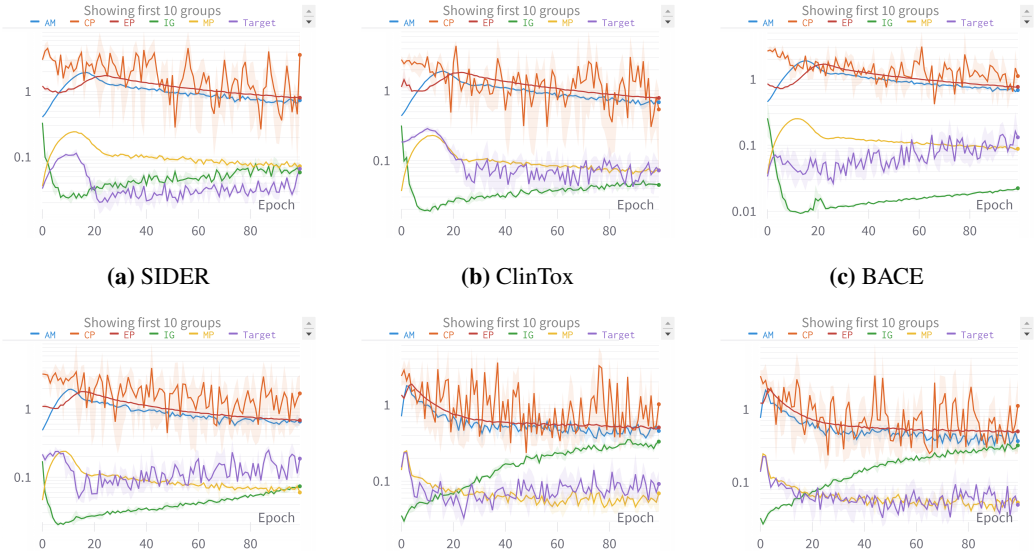
Compared to the results for Sup-C, using GraphMVP-C as the pretrained model substantially degrades the performance of all methods. This discrepancy could be due to the nature of auxiliary tasks and the pretraining objectives. The 2D-3D alignment pretraining objective in GraphMVP-C might lead to a feature representation that is optimized for capturing geometric information. Using auxiliary tasks, which are designed to capture 2D structural information, will likely result in spurious representations losing vital geometric information. Consequently, finetuning methods do not heavily alter the feature space and preserve most pretrained knowledge and thus perform better. On the other hand, Sup-C, pretrained on supervised and context prediction objectives, is more suitable for adaptation strategies leveraging 2D auxiliary tasks. These observations raise intriguing questions about the dependence of subsequent adaptation methods on the properties of pretrained models[44], warranting a more exhaustive exploration in future.

B.5 Additional Figures



(a) SIDER **(b)** BBBP **(c)** ToxCast

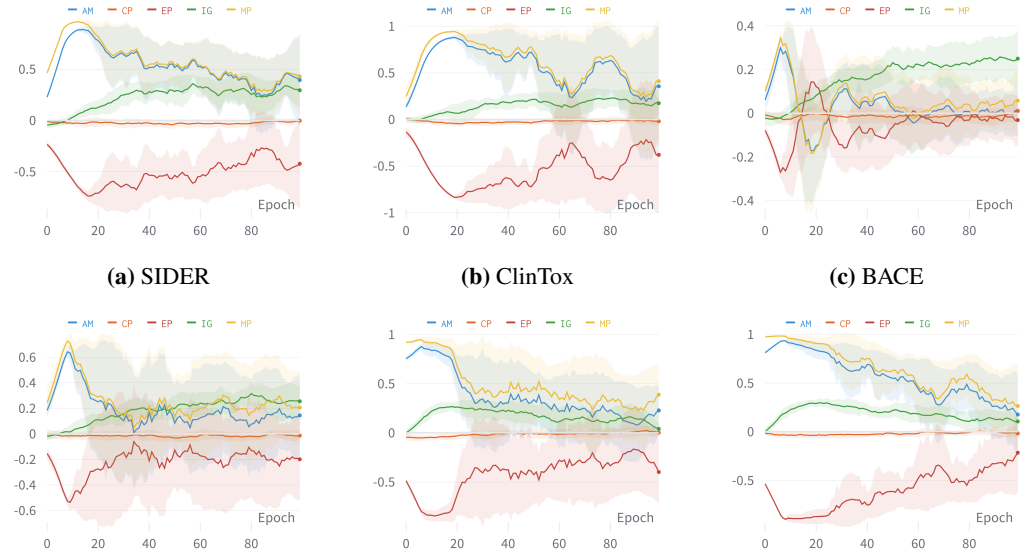
Figure 2: Target task gradient conflicts with almost all tasks except IG and MP. GraphMVP-C is adapted with all auxiliary tasks in a MTL setting.



(a) SIDER **(b)** ClinTox **(c)** BACE

(d) BBBP **(e)** Tox21 **(f)** ToxCast

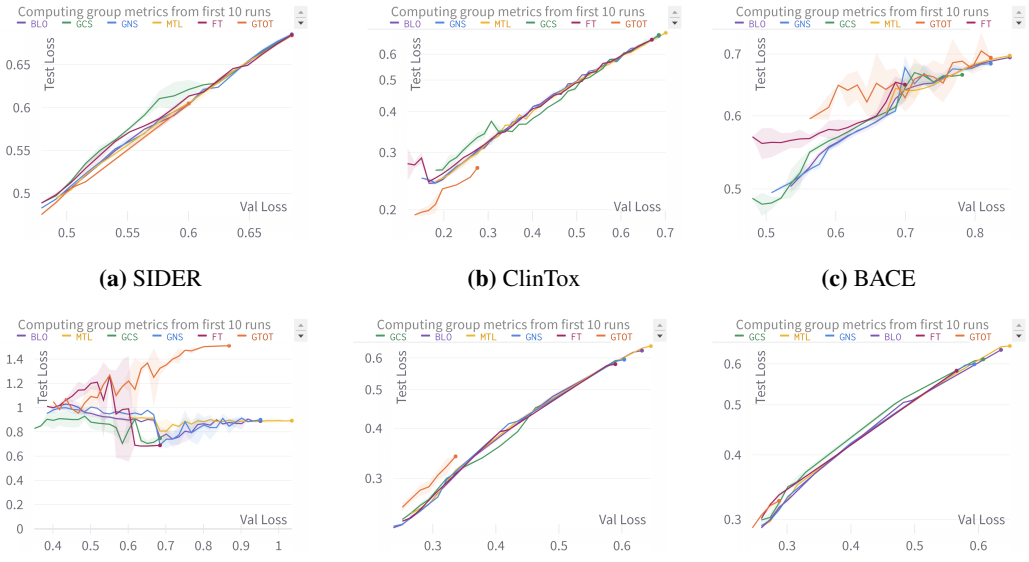
Figure 3: Large variations of scales among task gradients observed across multiple tasks. `Sup-C` is adapted with all auxiliary tasks in a `MTL` setting.



(a) SIDER **(b)** ClinTox **(c)** BACE

(d) BBBP **(e)** Tox21 **(f)** ToxCast

Figure 4: Target task gradient conflicts with EP and CP tasks. `Sup-C` is adapted with all auxiliary tasks in a `MTL` setting.



(a) SIDER **(b)** ClinTox **(c)** BACE

(d) BBBP **(e)** Tox21 **(f)** ToxCast

Figure 5: Error plots for experiments using all auxiliary tasks and Sup-C.

(a) SIDER **(b)** ClinTox **(c)** BACE

(d) BBBP **(e)** Tox21 **(f)** ToxCast

Figure 6: Error plots for experiments using {AM,IG,MP} and Sup-C.