Multi-task Learning with Domain Knowledge for Molecular Property Prediction

Shengchao Liu[1,2], **Meng Qu**[1,2], **Zuobai Zhang**[1,2], **Huiyu Cai**[1,2], **Jian Tang**[1,3,4]
[1]Mila [2]Université de Montréal [3]HEC Montréal [4]CIFAR AI Chair

Abstract

Multi-task learning for molecular property prediction is becoming increasingly important in drug discovery. However, in contrast to other domains, the performance of multi-task learning in drug discovery is still not satisfying as the number of labeled data for each task is too limited, which calls for additional data to complement the data scarcity. In this paper, we study multi-task learning for molecule property prediction in a different setting, where a relation graph between different tasks is available. We first extract a dataset including 400 tasks as well as a relation graph between different tasks. Then we systematically investigate modeling the relation between different tasks: (1) in the *latent* space by learning effective task representations on the task relation graph; (2) and in the *output* space via structured prediction methods (e.g., energy-based methods). Empirical results prove the effectiveness of our proposed approaches.

1 Introduction

Predicting the properties of molecules is a fundamental problem in drug discovery. Recently, we witnessed many successes of deep neural networks for molecular property prediction [4, 29, 24, 25, 31, 16, 18, 10, 26]. In particular, molecules are represented as molecular graphs, and graph neural networks (GNNs) [13] are utilized for learning molecular representations for predicting their properties. However, one big limitation for property prediction in drug discovery is that the labeled data are scarce, since they are very expensive and time-consuming to obtain. As a result, how to minimize the number of labeled data needed for effective molecular property prediction has long been a challenge in drug discovery.

One promising direction is multi-task learning, which tries to train multiple tasks (or properties) simultaneously so that the supervision or knowledge can be shared across different tasks. Indeed, multi-task learning has been successfully applied to different domains and applications such as natural language understanding [27, 30], computer vision [23, 20], speech recognition [34, 11], and recently to drug discovery [25, 18]. In general, the essential idea of these works is to infer the relation among different tasks, like learning a linear vector for task weights [2, 18] or pairwise task relation [15, 33, 30]. There are also some recent works on multi-task learning for molecular property prediction [4, 24, 31, 15, 18], which have shown very promising results compared to single task learning.

In this paper, we study multi-task learning for molecular property prediction in a different setting, where a relation graph between different tasks is explicitly given. We first construct a dataset by combining the chemical database ChEMBL [22] and the protein-protein interaction graph STRING [28]. ChEMBL can be formulated as a 2-dimensional matrix where each row is a molecule and each column is a task (biological effect), and each binary element in this matrix measures the interaction for that molecule-protein pair. The relationship between tasks are defined according to the relation of their associated sets of proteins, which can be inferred according to the protein-protein interaction

NeurIPS 2021 AI for Science Workshop.

graph in STRING. Finally, we are able to construct a dataset with 13,004 molecules and 382 different tasks, together with the task relation graph.

With this constructed dataset, we further propose to model the relation between different tasks in both the *latent* and *output* space. For the latent space representation, one can learn effective task representations by applying graph neural networks to the task relation graph. These task representations effectively capture the task similarities, which are further used for predictions in different tasks. Besides, for each molecule, its biological interactions with various tasks are dependent; thus we further formulate this as a problem of *structured prediction* [1] and propose an energy-based model (EBM) to model the joint distribution of the labels of a given molecule in different tasks. As training EBMs is computational expensive in general, we deploy the noise contrastive estimation (NCE) [9] for approximation, which turns the intractable integration problem into a simple classification: if the data is from the actual task distribution or from a noise distribution.

Related Work. In multi-task learning (MTL), there are two key problems: (1) how to learn the relation among tasks, and (2) how to model this task relation for improvement. Existing works on multi-task learning mainly focus on the first question, and can be classified into two categories: the architecture-specific methods [20, 23, 25] design specific neural network architecture, like task grouping using clustering; the architecture-agnostic methods [12, 2, 19, 18, 15, 33, 30] are modeling a linear task weight using certain optimization measure like loss and gradient. However, there is one limitation of the existing MTL methods: they neglect the effect of modeling the task relation. In our work, we circumvent the first problem by extracting an explicit task relation graph, and mainly target at the second problem: how to better model the task relation.

2 Problem Definition and Preliminaries

In this paper, given a molecular graph $\boldsymbol{x}$, our goal is to jointly predict its properties in T different tasks $\boldsymbol{y} = \{y_0, y_1, ..., y_{T-1}\}$ with a task relation graph $\mathcal{G}$. Besides, a **special** setting of our work is that a relation graph between different tasks $\mathcal{G} = (V, E)$ is given, and we will also take this into the modeling process. Here V is the set of tasks, E are the edges between different tasks. More information on the task relation graph $\mathcal{G}$ is included in Appendix A.

Graph Neural Networks (GNN). GNN is a powerful tool in capturing the structure information in the data. Gilmer et al. propose a general framework called message passing neural network (MPNN) in [8]. Following this, more recent works have explored how to model the complex structured data like molecular graph [5, 17, 26].

Energy-Based Models (EBMs). EBMs [14] use a parametric energy function $E_\phi(\boldsymbol{x}, \boldsymbol{y})$ to fit the data distribution. Formally, the probability of $p_\phi(\boldsymbol{y}|\boldsymbol{x})$ can be written as:

$$p_\phi(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))}{Z_\phi}, \tag{1}$$

where $E_\phi(\boldsymbol{x}, \boldsymbol{y})$ is the energy function, and $Z_\phi = \int_{\boldsymbol{y}' \in \mathcal{Y}} \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}'))$ is the partition function. Here $\mathcal{Y} = \{0, 1\}^T$ is the label space, and the partition function is computationally intractable due to the high dimensionality of the label space. We will discuss how to cope with this issue in Section 3.2.

3 Method and Primary Results

The existing mainstream multi-task learning methods [12, 2, 18, 33] learn the task relation implicitly, which can help balance optimization for each task during training. While in this paper, we focus on a different setting where the task relation graph is explicitly given. We propose two modules to model the task relation in *latent* and *output* space respectively.

3.1 Modeling Task Relation in the Latent Space

We propose a GNN to model the task relation. The task relation is implicitly encoded in the task representations, and the final predictions are made independently for each task. We first embed the molecule into the feature space as $\boldsymbol{z}(\boldsymbol{x}) \in \mathbb{R}^{d_m}$ using Graph Isomorphism Network (GIN) [32], where d_m is the latent dimension. Then to embed each task, we use one-hot encoding, and by

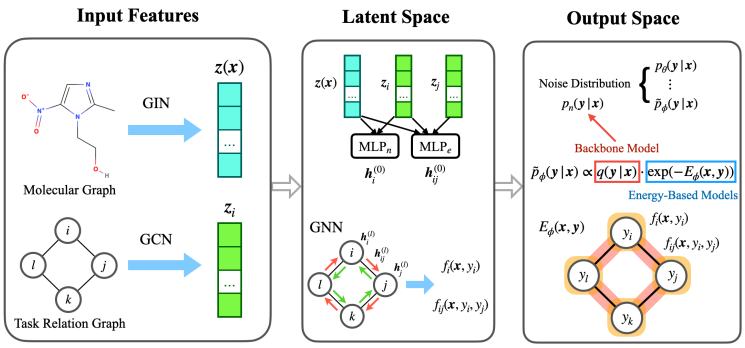

Figure 1: Pipeline of EBM-GNN. We first obtain molecule and task embedding via GIN and GCN. Then, they are used to learn the latent representation for each task via a GNN model in the latent space. In EBM-GNN, a GNN model is used to model the task relation graph in the latent space and EBM learns the task distribution in the output space. The ultimate likelihood applies the energy tilting term: $\hat{p}_{\theta,\phi}(\boldsymbol{y}|\boldsymbol{x})$.

passing it through graph convolutional network (GCN) [13], we get the explicit task representation $\boldsymbol{z}^{(i)} \in \mathbb{R}^{d_t \times 1}, \forall i \in \{0, 1, \ldots, T-1\}$, where d_t is the task embedding dimension. More detailed discussions are in Appendices B and C.

Given the molecule and task representations ($\boldsymbol{z}(\boldsymbol{x})$ and $\boldsymbol{z}^{(i)}$), we then model the task relation using another GNN, and the node- and edge-level inputs are:

$$\boldsymbol{h}_i^{(0)}(\boldsymbol{x}) = \text{MLP}_n^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)}), \qquad \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}) = \text{MLP}_e^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)} \oplus \boldsymbol{z}^{(j)}), \tag{2}$$

where $\oplus$ is the concatenation operation of two tensors. The two mapping functions $\text{MLP}_n^{(0)}(\cdot) : \mathbb{R}^{d_m+d_t} \to \mathbb{R}^{C \times d}$ and $\text{MLP}_e^{(0)}(\cdot) : \mathbb{R}^{d_m+2d_t} \to \mathbb{R}^{C \times C \times d}$ are two multi-layer perceptrons (MLP) on the node- and edge-level respectively. $C = 2$ is for binary task and d is the latent dimension. Different from existing GNN models where there is only one state for each node and edge, we have C node states and $C \times C$ edge states. The representation for each node and edge state is defined as:

$$\boldsymbol{h}_i^{(0)}(\boldsymbol{x}, y_i) = \boldsymbol{h}_i^{(0)}(\boldsymbol{x})[y_i], \qquad \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}, y_i, y_j) = \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x})[y_i, y_j]. \tag{3}$$

With the inputs defined above, we then define the message-passing function in this task relation GNN. As we have multiple states for each node and edge, the aggregation function will consider the states accordingly. The propagation on the l-th layer is:

$$\begin{aligned} \boldsymbol{h}_i^{(l+1)}(\boldsymbol{x}, y_i) &= \text{MPNN}_n^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \left\{\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) \mid \forall j, y_j\right\}\right) \\ \boldsymbol{h}_{ij}^{(l+1)}(\boldsymbol{x}, y_i, y_j) &= \text{MPNN}_e^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \boldsymbol{h}_j^{(l)}(\boldsymbol{x}, y_j), \boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j)\right), \end{aligned} \tag{4}$$

where MPNN_n and MPNN_e are MPNN layers defined on the nodes and edges on the $(l+1)$-th layer respectively. Iterating Equation (4) L times can give the latent representation for each molecule-task pair. The detailed structure of MPNN is described in Appendix D.

Finally, we make predictions for each task independently. We first get the node representation by concatenating the two state representations, after which we apply a readout function R:

$$f_i(\boldsymbol{x}) = R(\{\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, 0) \oplus \boldsymbol{h}_i^{(l)}(\boldsymbol{x}, 1) \mid l = 1, \ldots, L\}), \tag{5}$$

where $R : \mathbb{R}^{2dL} \to \mathbb{R}$ is an MLP. Because $C = 2$ for binary classification, the label distribution is $p(y_i = 1|\boldsymbol{x}, \mathcal{G}) = \text{sigmoid}(f_i(\boldsymbol{x}))$, and the loss function is the binary cross entropy over all T binary tasks $\mathcal{L} = \sum_{i=0}^{T-1} \log p(y_i|\boldsymbol{x}, \mathcal{G})$.

3.2 Modeling Task Relation in the Output Space

The multi-task learning methods discussed so far are predicting each task independently. However, there also exists a task distribution over multiple tasks, *i.e.*, $p(\boldsymbol{y} = y_0, y_1, \ldots, y_{T-1}|\boldsymbol{x})$. In this subsection, we propose to apply an energy-based model (EBM) to inject the prior knowledge about task dependency and model it with joint task distribution.

We define the energy function as the summation of first-order (node) and second-order (edge) factors on the graph $E_\phi(\boldsymbol{x}, \boldsymbol{y}) = -\sum_{i=0}^{T-1} f_i(\boldsymbol{x}, y_i) - \lambda \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j)$, where λ is the coefficient between first- and second-order factors. The conditional probability under the EBM framework is defined as:

$$p_\phi(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\Big(\sum_i f_i(\boldsymbol{x}, y_i) + \sum_{ij} f_{ij}(\boldsymbol{x}, y_i, y_j)\Big)}{Z_\phi}. \tag{6}$$

The energy function in EBM can have flexible formulation [14]. Following Section 3.1, we define our energy function as $f_i(\boldsymbol{x}, y_i) = \tilde{R}(\{\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) \mid l = 1, \ldots, L\})$, $f_{ij}(\boldsymbol{x}, y_i, y_j) = \tilde{R}(\{\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) \mid l = 1, \ldots, L\})$, where $\tilde{R} : \mathbb{R}^{dL} \to \mathbb{R}$ is a readout function defined as $\tilde{R} = \sigma(\text{MLP}(\cdot))$ and $\sigma(\cdot)$ is the activation function. These two equations are mapping the node and edge representations to scalars (or energies) indexed with the corresponding node and edge label.

Learning. We apply the noise contrastive estimation (NCE) [9] for EBM learning, which casts the problem of maximizing log-likelihood into a binary classification task. It first takes the normalization constant Z_ϕ in Equation (1) as a learned scalar parameter, then it transforms the EBM learning into a binary classification problem by maximizing the following objective:

$$\mathcal{L}_{NCE} = \mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})}, \tag{7}$$

where p_{data}, p_ϕ and p_n denote the data distribution, the model distribution and the noise distribution over tasks respectively. In practice, the noise distribution should be close to the data distribution to facilitate the mining of hard negative samples. We also combine the energy tilting term into NCE learning. We apply the backbone model for the noise distribution, *i.e.*, $p_n = q$, and replace the energy tilting term into Equation (7). More detailed derivations are attached in Appendix F.

Inference. The inference procedure aims at computing the marginal distribution for each task, and the main challenge is how to calculate the intractable partition function. Here we use Gibbs sampling [7]: it is a standard method and the core idea is to generate samples by sweeping through each variable to a sample with the remaining variables fixed. For each data point $(\boldsymbol{x}, y_0, \ldots, y_{T-1})$, we iteratively sample label for each task with other task labels fixed. The update function at each iteration is:

$$p_\phi(y_i|\boldsymbol{y}_{-i}, \boldsymbol{x}) = \frac{\exp\big(f_i(\boldsymbol{x}, y_i) + \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j)\big)}{\sum_{y_i=0}^{C-1} \exp\big(f_i(\boldsymbol{x}, y_i) + \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j)\big)}, \tag{8}$$

where $\boldsymbol{y}_{-i}$ denotes all T task labels except the i-the task. And taking this as the tilting term, we apply $\tilde{p}(\boldsymbol{y}|\boldsymbol{x}) = p_\phi(\boldsymbol{y}|\boldsymbol{x}) \cdot q(\boldsymbol{y}|\boldsymbol{x})$ for sampling. To accelerate the convergence of Gibbs sampling, we take the backbone model for initial distribution.

Table 1: Dataset is split into 8-1-1 for train, valid, and test. For each method, we run 5 seeds and report the mean and standard deviation. The best performance is **highlighted**.

Method	ChEMBL 10	ChEMBL 50	ChEMBL 100
STL	71.67 ± 0.64	73.57 ± 1.20	70.81 ± 1.28
MTL	74.83 ± 0.61	79.37 ± 1.76	77.78 ± 1.59
UW [12]	72.49 ± 0.53	79.68 ± 0.98	78.71 ± 1.93
GradNorm [2]	75.17 ± 0.77	79.46 ± 1.27	78.75 ± 1.60
DWA [19]	72.45 ± 1.31	79.35 ± 0.68	78.21 ± 2.31
LBTW [18]	75.21 ± 0.49	79.52 ± 0.56	79.07 ± 0.99
GNN (ours)	77.72 ± 0.66	79.69 ± 1.07	80.19 ± 2.01
EBM-GNN (ours)	$\mathbf{78.04 \pm 0.73}$	$\mathbf{80.34 \pm 1.08}$	$\mathbf{80.48 \pm 1.93}$

Primary results. We use the proposed dataset with three thresholds introduced in Appendix A for experiments, and report the average ROC-AUC over all T binary tasks in Table 1. We can see all the multi-task learning methods are better than the single-task learning, which matches with the common belief that the joint learning can improve the overall performance. Then for our proposed methods, we can see that modeling task relation in the latent space using GNN reaches a good performance compared to all multi-task learning baselines, while combining it with the EBM in the output space, *i.e.*, EBM-GNN, can reach the best performance on all datasets. Due to the space limitation, we put the detailed descriptions of baselines and our models in Appendix E.

References

[1] David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992. PMLR, 2016.

[2] 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, 2018.

[3] The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 11 2018.

[4] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28:2224–2232, 2015.

[6] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[7] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[9] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

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

[11] Abhinav Jain, Minali Upreti, and Preethi Jyothi. Improved accented speech recognition using accent embeddings and multi-task learning. In *Interspeech*, pages 2454–2458, 2018.

[12] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[15] Shengchao Liu. Exploration on deep drug discovery: Representation and learning. *Master's Thesis*, TR1854, 2018.

[16] Shengchao Liu, Moayad Alnammi, Spencer S Ericksen, Andrew F Voter, Gene E Ananiev, James L Keck, F Michael Hoffmann, Scott A Wildman, and Anthony Gitter. Practical model selection for prospective virtual screening. *Journal of chemical information and modeling*, 59(1):282–293, 2018.

[17] Shengchao Liu, Mehmet Furkan Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *arXiv preprint arXiv:1806.09206*, 2018.

[18] Shengchao Liu, Yingyu Liang, and Anthony Gitter. Loss-balanced task weighting to reduce negative transfer in multi-task learning. 2019.

[19] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.

[20] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343, 2017.

[21] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018.

[22] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 11 2018.

[23] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[24] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[25] Bharath Ramsundar, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P Sheridan, and Vijay Pande. Is multitask deep learning practical for pharma? *Journal of chemical information and modeling*, 57(8):2068–2076, 2017.

[26] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany, August 2016. Association for Computational Linguistics.

[28] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613, 2019.

[29] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Deep learning as an opportunity in virtual screening. *Advances in neural information processing systems*, 27, 2014.

[30] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2021.

[31] 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.

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[33] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[34] Yu Zhang, Pengyuan Zhang, and Yonghong Yan. Attention-based lstm with multi-task learning for distant speech recognition. In *Interspeech*, pages 3857–3861, 2017.

A Dataset Generation

We propose a multi-task learning dataset with explicit task relation for the molecular property prediction. This new dataset is built on the Large Scale Comparison (LSC) dataset [21], and we list the three main steps in Appendices A.1 to A.3.

A.1 Filtering molecules

Among 456,331 molecules in the LSC dataset, 969 are filtered out following the pipeline in [10]. Here we describe the detailed filtering process, and the molecules filtered out in each step.

1. Discard the 22 `None`s in the compound list.
2. Filter out the 9 molecules with ≤ 2 non-H atoms.
3. Retain only the largest molecule in the SMILES string. E.g. if the compound is a organic hydrochloride, say $\mathrm{CH_3NH_3^+Cl^-}$, we retain only the organic compound after removing HCl, in this case $\mathrm{CH_3NH_2}$.
4. Filter out 929 molecules with molecular weight < 50 and 9 with m.w. > 900.

A.2 Querying the PPI scores

Then we obtain the PPI scores by quering the ChEMBL [22] and STRING [28] databases. The details are as follows:

1. The LSC dataset [21] gives the ChEMBL ID for each assay. We use the assay id to query the ChEMBL database by visiting `https://www.ebi.ac.uk/chembl/api/data/assay/[assay_id]` for target ID. We then query the ChEMBL database by visiting `https://www.ebi.ac.uk/chembl/api/data/ target/[target_id]` for UniProt [3] information. We save all the uniprots related to each target in a list. We discard the 585 assays with no associated uniprot, and confirm that all remaining assays are targeting human proteins.
2. Next, we query the STRING database for the corresponding STRING ID for each of the 2,670 uniprots by visiting `https://string-db.org/api/xml/get_string_ids?identifiers=[uniprot]`. 40 uniprots do not have corresponding StringIDs, so we discard them. The String ID list is then sent to `https://string-db.org/ api/tsv-no-header/network` via a POST request to obtain 12,356 human PPI scores.

A.3 Constructing the Task Relation Graph

Finally, we calculate the edge weights w_{ij}, *i.e.*, task relation score, for task t_i and t_j in the task relation graph to be $\max\{\mathrm{PPI}(s_i, s_j) : s_i \in S_i, s_j \in S_j\}$, where S_i denotes the protein set of task t_i. The resulting task relation graph has 1,310 nodes and 9,172 edges with non-zero weights. Note that 96% of the protein-targeted tasks only target a single protein, for which the relation score of these tasks is exactly the PPI score between their target proteins. We then densify the dataset via the following filtering process:

1. We filter out all isolated tasks.
2. We define a threshold τ and iteratively filter out molecules with number of labels below τ, tasks with number of labels below τ, and tasks with number of positive or negative labels below 10. We repeat this until no molecule or task is filtered out.

B GIN for Molecule Embedding

The Graph Isomorphism Network (GIN) is proposed in [32]. It was originally proposed for the simple graph structured data, where each node has one discrete label and no extra edge information is provided. Here we adopt a customized GIN from a recent paper [10]. With this customized GIN as the base model, plus pre-training techniques, [10] can reach the state-of-the-art performance on several molecular property prediction tasks. Thus we adopt this customized GIN model in our work.

Following the notation in Section 2, each molecule is represented as a molecular graph, *i.e.*, $\boldsymbol{x} = (X, E)$, where X and E are feature matrices for atoms and bonds respectively. Suppose for one molecule, we have n atoms and m edges. The message passing function is defined as:

$$z_i^{(k+1)} = \text{MLP}_{\text{atom}}^{(k+1)}\Big(z_i^{(k)} + \sum_{j \in \mathcal{N}(i)} \big(z_j^{(k)} + \text{MLP}_{\text{bond}}^{(k+1)}(E_{ij})\big)\Big), \tag{9}$$

where $z_0 = X$ and $\text{MLP}_{\text{atom}}^{(k+1)}$ and $\text{MLP}_{\text{bond}}^{(k+1)}$ are the $(l+1)$-th MLP layers on the atom- and bond-level respectively. Repeating this for K times, and we can encode K-hop neighborhood information for each atom in the molecular data, and we take the last layer for each node/atom representation. The graph representation is the mean of the node representation, *i.e.*, the molecule representation in this paper:

$$z(\boldsymbol{x}) = \frac{1}{N} z_i^{(K)} \tag{10}$$

C GCN for Task Embedding

We use graph convolutional network (GCN) [13] for the task embedding. For the i-th task, we first get its one-hot encoding and then pass it through an embedding layer, with the output denoted as $\boldsymbol{e}_i \in \mathbb{R}^{d_t \times 1}, \forall i \in \{0, 1, \ldots, T-1\}$, where d_t is the task embedding dimension. $\boldsymbol{E} = \{\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{T-1}\}^T \in \mathbb{R}^{T \times d_t}$ is the initial embedding matrix for T tasks. Then we pass $\boldsymbol{E}$ through a GCN and the output embedding for the i-th task is $\boldsymbol{z}^{(i)} = \text{GCN}(\boldsymbol{E})_i, \forall i \in \{0, 1, \ldots, T-1\}$.

D GNN for Modeling Latent Space

In this section, we give a detailed illustration of our proposed GNN model in Section 3.1. The general pipeline is shown in Figure 2.

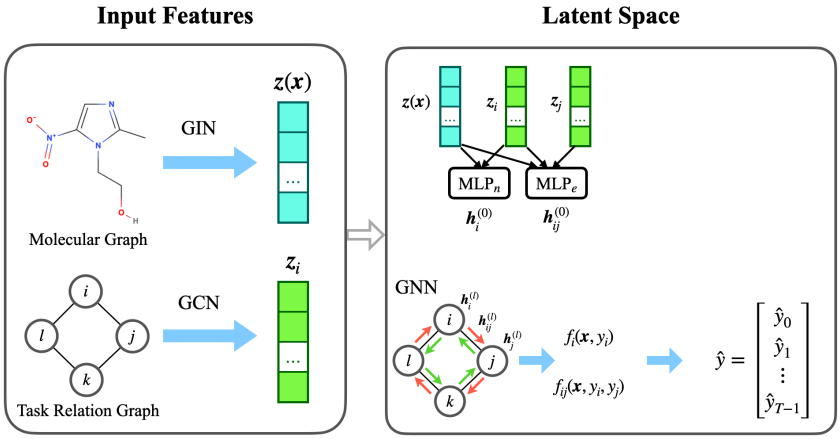

Figure 2: Pipeline of GNN. We first obtain molecule and task embedding via GIN and GCN. Then they are concatenated and passed through a GNN to better learn the task representation. The final prediction for each task is predicted independently on each node representation.

First let us quickly review the node- and edge-level inputs:

$$\begin{aligned} \boldsymbol{h}_i^{(0)}(\boldsymbol{x}) &= \text{MLP}_n^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)}) \\ \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}) &= \text{MLP}_e^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)} \oplus \boldsymbol{z}^{(j)}), \end{aligned} \tag{11}$$

and as discussed in Section 3.1, the biggest difference between our GNN model and the mainstream GNN models is that each node and each edge has two and four states respectively, where each state represents a label.

Recall that in this task relation graph, y_i is the label for the i-th task, and it has two values; similarly for each edge $\langle y_i, y_j \rangle$ has four labels with a simple combination. Thus the representations for node label y_i and edge label $\langle y_i, y_j \rangle$ are as follows:

$$\begin{aligned} \boldsymbol{h}_i^{(0)}(\boldsymbol{x}, y_i) &= \boldsymbol{h}_i^{(0)}(\boldsymbol{x})[y_i] \\ \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}, y_i, y_j) &= \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x})[y_i, y_j]. \end{aligned} \tag{12}$$

With the node and edge inputs, we can then define the message-passing propagation. Notice that here we are propagating on both the node- and edge-levels. Following the notations in Section 3.1, for the node-level propagation we have:

$$\begin{aligned} \boldsymbol{h}_i^{(l+1)}(\boldsymbol{x}, y_i) &= \text{MPNN}_n^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \{\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) \mid \forall j, y_j\}\right) \\ &= \text{MLP}_n^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) + \sum_{j \in \mathbb{N}(i)} \sum_{y_j=0}^{C-1} \boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j)\right), \end{aligned} \tag{13}$$

and for the edge-level propagation, we have:

$$\begin{aligned} \boldsymbol{h}_{ij}^{(l+1)}(\boldsymbol{x}, y_i, y_j) &= \text{MPNN}_e^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \boldsymbol{h}_j^{(l)}(\boldsymbol{x}, y_j), \boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j)\right) \\ &= \text{MLP}_e^{(l+1)}\left(\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) + \text{MLP}_a^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) + \boldsymbol{h}_j^{(l)}(\boldsymbol{x}, y_j)\right)\right), \end{aligned} \tag{14}$$

where $\text{MLP}_n^{(l+1)}(\cdot)$, $\text{MLP}_e^{(l+1)}(\cdot)$ and $\text{MLP}_a^{(l+1)}(\cdot)$ are MLP layers defined on the node-level, edge-level, and in the aggregation function from nodes to edges. All three MLP layers are mapping functions defined on $\mathbb{R}^d \to \mathbb{R}^d$.

E Training Details

Baselines As discussed before, the memory cost of architecture-specific multi-task learning methods (Bypass network) is $O(T)$, while pair-wise architecture-agnostic multi-task learning methods (RMTL, PCGrad, GradVac) have $O(T^2)$ complexity. Both are infeasible in the deep multi-task learning setting, so we exclude these in the experiment. For the baseline methods, we include standard single-task learning (STL), standard multi-task learning (MTL), Uncertainty Weighing (UW) [12], GradNorm [2], Dynamic Weight Average (DWA) [19], and Loss-Balanced Task Weighting (LBTW) [18].

Our Methods We proposed two approaches in Section 3. (1) GNN models the task relation graph in the latent space, as proposed in Section 3.1. (2) EBM-GNN is proposed in Section 3.2. It models the task relation graph in both the latent and output space under the EBM framework, where the energy function is defined as GNN. We explore two noise distributions in the NCE learning steps: (2.1) the first is a fixed pre-trained GNN, $p_n = p_\theta$; (2.2) the second is taking the pre-trained GNN, $p_n = p_\theta$, as initial noise distribution, and then adaptively updating this noise distribution with the latest model distribution $p_n = \tilde{p}_\phi$. More training details can be found in Appendix E.

Evaluation We follow the mainstream evaluation metrics on multi-task learning for drug discovery, *i.e.*, the mean of ROC-AUC over all T tasks. ROC-AUC is ranking-based, thus it can better match with the class-imbalance tasks like molecular property prediction in drug discovery.

Reproducibility The sizes are actually quite small for ChEMBL-50 and ChEMBL-100, leading to the high variation. To alleviate this issue, for each method on each dataset, we run 5 times with different seeds, and report the mean and standard deviation. Another thing that may increase the unstable performance is the implementation of GNN: our code is built on PyTorch Geometric [6], which uses torch-scatter operation. This atomic operation is non-deterministic and has not been solved yet. Thus for reproducibility, we will provide the trained model weight after the paper is accepted.

To train our proposed model, we use Adam for optimization with learning rate 1e-3, and the batch size is 32 for ChEMBL-10 (due to the memory issue) and 128 for ChEMBL-50 and ChEMBL-100. We train 200 epochs on ChEMBL-10 (within 36 hours) and 500 epochs on ChEMBL-50 and ChEMBL-100 (within 2 hours). The base graph neural network for molecule representation is GIN [32], and we follow the hyperparameter used in [10]. The base graph neural network for task embedding is GCN [13]. We have more detailed description of GIN and GCN in Appendices B and C. The hyperparameter tuning for all baseline methods and GNN base models in Appendix E.1.

E.1 Hyperparameter Tuning

We list the hyperparameters for baselines models and our proposed models in Table 2, including MTL, UW [12], GradNorm [2], Dynamic Weight Average (DWA) [19], and Loss-Balanced Task Weighting (LBTW) [18], GNN in Section 3.1, EBM-GNN in Section 3.2.

F Noise Contrastsive Esitmation with Energy Tilting Term

Here we present the derivation of the training objective function of NCE learning with tilting term in section 3.2. When applying the backbone model for noise distribution, *i.e.*, $p_n = 1$, and adopting the self-normalization ($Z = 1$), the loss can be rewritten as:

$$\begin{aligned}
\hat{\mathcal{L}}_{NCE} &= \mathbb{E}_{\boldsymbol{y}\sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} \\
&= \mathbb{E}_{\boldsymbol{y}\sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_n(\boldsymbol{y}|\boldsymbol{x})\exp(-E_\phi(\boldsymbol{x},\boldsymbol{y}))} + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})\exp(-E_\phi(\boldsymbol{x},\boldsymbol{y}))}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_n(\boldsymbol{y}|\boldsymbol{x})\exp(-E_\phi(\boldsymbol{x},\boldsymbol{y}))} \\
&= \mathbb{E}_{\boldsymbol{y}\sim p_n} \log \frac{1}{1+\exp(-E_\phi(\boldsymbol{x},\boldsymbol{y}))} + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \log \frac{1}{1+\exp(E_\phi(\boldsymbol{x},\boldsymbol{y}))}.
\end{aligned} \tag{15}$$

Table 2: Hyperparameters for baselines and our models.

Model	Hyperparameters	Values
MTL	Epochs	[100, 200]
UW	Epochs	[100, 200]
GradNorm	Epochs	[100, 200]
	α	[0.1, 0.2, 0.5]
DWA	Epochs	[100, 200]
	T	[0.2]
LBTW	Epochs	[100, 200]
	α	[0.1, 0.2, 0.5]
GNN	Epochs	[200, 500]
	d	[50, 100]
	# GCN Layer	[0, 2]
	# GNN Layer	[2]
EBM-GNN	Epochs	[200, 500]
	Fixed-Noise Distribution Epochs	[200, 300, 400, 1000]
	d	[50, 100]
	# GCN Layer	[0, 2]
	# GNN Layer	[0, 2, 4]
	λ	[0.1, 1]