Structured Multi-task Learning 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 molecular property prediction in a novel setting, where a relation graph between tasks is available. We first construct a dataset including around 400 tasks as well as a task relation graph. Then to better utilize such relation graph, we propose a method called SGNN-EBM to systematically investigate the structured task modeling from two perspectives. (1) In the *latent* space, we model the task representations by applying a state graph neural network (SGNN) on the relation graph. (2) In the *output* space, we employ structured prediction with the energy-based model (EBM), which can be efficiently trained through noise-contrastive estimation (NCE) approach. Empirical results justify the effectiveness of SGNN-EBM. Code is available on this GitHub repository.

1 Introduction

Predicting the properties of molecules (*e.g.*, binding affinity with proteins, toxicity, ADME property) is a fundamental problem in drug discovery. Recently, we witness many successes of deep neural networks for molecular property prediction Dahl et al. (2014); Unterthiner et al. (2014); Ramsundar et al. (2015, 2017);

Wu et al. (2018); Liu et al. (2018, 2019c); Hu et al. (2019b); Rong et al. (2020); Alnammi et al. (2021); Liu et al. (2021). In particular, molecules are represented as molecular graphs, and graph neural networks Kipf and Welling (2016)—which are neural network architectures specifically designed for graphs—are utilized for learning molecular representations. These neural networks are then usually trained with a set of labeled molecules. However, one big limitation for property prediction in drug discovery is that the labeled data are very limited, 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 tasks. Indeed, multi-task learning has been successfully applied to different domains and applications such as natural language understanding Søgaard and Goldberg (2016); Wang et al. (2021), computer vision Misra et al. (2016); Lu et al. (2017), and speech recognition Zhang et al. (2017); Jain et al. (2018). In general, the essential idea of these works is to infer the relation among tasks. For example, Lu et al. (2017) studied the hierarchical structure of different tasks; some more recent works Liu (2018); Yu et al. (2020); Wang et al. (2021) tried to infer the pairwise relation between tasks based on the gradients or loss of the tasks. There are also some recent work on multi-task learning for molecular property prediction Dahl et al. (2014); Ramsundar et al. (2015, 2017); Wu et al. (2018); Liu (2018); Liu et al. (2019c), which have shown very promising results. However, drug discovery possesses certain attributes distinguishable from other domains, making it more challenging and interesting. (1) There is rich information in chemistry and biology domain, *e.g.*, the task relation if we are referring molecules as data and corresponding biological effects as the tasks. Then the question is how to better utilize such domain knowledge. (2) The number of molecules for each task is comparatively small, and merging data from different tasks may lead to a severe

data sparsity issue (an example in Section 4), which adds more obstacles for learning.

In this paper, we study multi-task learning for molecular property prediction in a different setting, where a relation graph between tasks is explicitly given via domain knowledge. We first construct a large-scale dataset (w.r.t. the task number) by combining the chemical database of bioactive molecules (ChEMBL Mendez et al. (2018)) and the protein-protein interaction graph (STRING Szklarczyk et al. (2019)). Specifically, we define a binary classification task based on an assay in ChEMBL, which measures the biological effects of molecules over a set of proteins. The relationship between different tasks are defined according to the relation of their associated sets of proteins, which can be inferred according to the protein-protein interaction graph in STRING. Finally, we are able to construct a large-scale dataset with 13,004 molecules and 382 tasks, together with the corresponding task relation graph.

With this constructed dataset, we propose a **novel** research problem: *How to do structured multi-task learning with an explicit task relation graph?* Our proposed solution is SGNN-EBM, which models the structured task information in both the *latent* and *output* space. More specifically, a state graph neural network (SGNN) can learn effective task representations by utilizing the relation graph, where the learnt representations effectively capture the similarities between tasks in the latent space. However, given a molecule, its labels are predicted independently for each task, which ignores the task dependency, *i.e.*, the dependency in the output space. Therefore, we further introduce formulating multi-task learning as structured prediction Belanger and McCallum (2016) problem, and apply an energy-based model (EBM) to model the joint distribution of the labels in the task space. Our proposed solution, coined SGNN-EBM, combines the advantages of both by adopting SGNN into the energy function in EBM, which provides higher capacity for structured task modeling. As training SGNN-EBM is generally computationally expensive, we deploy the noise contrastive estimation (NCE) Gutmann and Hyvärinen (2010) for effective training, which trains a discriminator to distinguish the observed examples and examples sampled from a noise distribution.

Our major contributions include (1) To our best knowledge, we are the **first** to propose doing multi-task learning with an explicit task relation graph; (2) We construct a domain-specific multi-task dataset with relation graph for drug discovery; (3) We propose SGNN-EBM for task structured modeling in both the latent and output space; (4) We achieve consistently better performance using SGNN-EBM.

2 Related Work

In the multi-task learning (MTL) literature, there are two fundamental problems: (1) how to learn the relation among tasks, and (2) how to model the task relation once available. Existing works on MTL *merely* focus on the first question, which can be roughly classified into two categories: architecture-specific MTL and architecture-agnostic MTL.

Architecture-specific MTL aims at designing special architecture to better transfer knowledge between tasks. Fully-adaptive network Lu et al. (2017) dynamically groups similar tasks in a hierarchical structure. Cross-stitch network Misra et al. (2016) applies multiple cross-stitch units and Bypass network Ramsundar et al. (2017) manipulates the architecture to model task relation. One drawback is that as the number of tasks grows, the requirement of computation memory increases linearly, which limits their application to large-scale setting (w.r.t. the task number).

Architecture-agnostic MTL provides a more general solution by learning to balance the tasks numerically. It has two components: a shared representation module and multiple task-specific prediction modules. Based on this framework, several methods have been proposed to learn a global linear task coefficient according to the optimization process, such as the the uncertainty Kendall et al. (2018), and task gradients and losses Chen et al. (2018); Liu et al. (2019b,c). The learnt linear vector is then applied on the task-specific predictors. Instead of learning such linear vector, one alternative approach is to learn the pairwise task relation. RMTL Liu (2018) first handles this by applying a reinforcement learning framework to reduce the gradient conflicts between tasks. PCGrad and Grad-Vac Yu et al. (2020); Wang et al. (2021) follow the same motivation and use gradient projection. However, there is one drawback on the high computational cost, since the pair-wise computation grows quadratically with the number of tasks; thus they are infeasible for large-scale setting (w.r.t. the task number).

Molecular property prediction has witnessed certain successful applications with MTL Merck (2012); Dahl et al. (2014); Unterthiner et al. (2014); Ramsundar et al. (2015); Wu et al. (2018); Liu (2018); Liu et al. (2018, 2019c) in terms of the robust performance gain. Furthermore, Lee and Kim (2019) finds that similarity within a target group significantly affects the performance of MTL on molecular binding prediction, revealing the importance of utilizing the task relation in drug discovery. However, all the aforementioned MTL methods do not possess the knowledge of the

task relation and thus the main focus is to learn it in an architecture-specific or architecture-agnostic manner. While in this work, the task relation is given, and our focus moves to how to better model the structured task information in the MTL setting.

3 Problem Definition & Preliminaries

3.1 Problem Definition

Molecular Graph and Property Prediction. In molecular property prediction tasks, each data point $\boldsymbol{x}$ is a molecule, which can be naturally viewed as a *topological graph*, where atoms and bonds are nodes and edges accordingly. For each molecule $\boldsymbol{x}$, we want to predict T biological or physical *properties* Wu et al. (2018), where each property corresponds to one *task*. For notation, we want to predict $\boldsymbol{y} = \{y_0, y_1, ..., y_{T-1}\}$ for each molecule $\boldsymbol{x}$. Each task corresponds to C classes if it is a classification problem; and specifically in this work, we will be targeting at the binary tasks, *i.e.*, $C = 2$ and $y_i \in \{0, 1\}, \forall i \in \{0, 1, \cdots, T-1\}$.

Multi-Task Learning (MTL). Due to the inherent data scarcity issue in drug discovery Ramsundar et al. (2015); Wu et al. (2018); Mayr et al. (2018); Hu et al. (2019a), training an independent model for each task often yields inferior performance. In practice Mayr et al. (2018), a more effective and widely-adopted approach is *multi-task learning (MTL)*, which tries to optimize multiple tasks simultaneously.

Task Relation Graph. A *task relation graph* is $\mathcal{G} = (V, E)$, where V is the node set of tasks and E are the corresponding edges between tasks. Here we add a linkage between two tasks if they are closely related. Thus, this relation graph can effectively complement the information sparsity of the labeled data for different tasks. More information on the task relation graph $\mathcal{G}$ will be introduced in Section 4.

Structured Task Modeling. In this paper, we propose a *novel* research problem for MTL: how to do *structured task modeling* when the task relation graph is explicitly provided. Specifically, given a molecular graph $\boldsymbol{x}$, our goal is to jointly predict its labels for T tasks $\boldsymbol{y} = \{y_0, y_1, ..., y_{T-1}\}$ with a task relation graph $\mathcal{G}$. In other words, we aim to model $p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{G})$.

3.2 Preliminaries

Graph Neural Network (GNN) is a powerful tool in modeling structured data, like molecular graph and task relation graph. Gilmer et al. (2017) first proposes a general GNN framework called *message passing neural network (MPNN)*. Following this, recent works have explored how to model the complex structured data like molecular graph Duvenaud et al. (2015); Rong et al. (2020); Liu et al. (2019a); Demirel et al. (2021); Ying et al. (2021) and knowledge graph Kipf and Welling (2016); Xu et al. (2018). Typically for the node-level prediction, GNN models predict the node labels independently, and this limits the learning power of GNN to model the joint distribution of labels.

Energy-Based Model (EBM) uses a parametric energy function $E_\phi(\boldsymbol{x}, \boldsymbol{y})$ to fit the data distribution LeCun et al. (2006). The energy function induces a density function with the Boltzmann distribution. Formally, the probability of $p_\phi(\boldsymbol{y}|\boldsymbol{x})$ can be written as:

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

where $E_\phi(\boldsymbol{x}, \boldsymbol{y})$ is the energy function, with which EBM is allowed to model the structured output space. $Z_\phi(\boldsymbol{x}) = \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}'))$ 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 cardinality in $|\mathcal{Y}| = 2^T$. We will discuss how to cope with this issue for learning and inference in Section 5.

4 Dataset with Explicit Task Relation

In this section, we describe how to construct the molecular property prediction dataset together with an explicit task relation graph. The *task* here refers to a binary classification problem on a ChEMBL assay Mendez et al. (2018), which measures certain biological effects of molecules, *e.g.*, toxicity, inhibition or activation of proteins or whole cellular processes, etc. We focus on tasks that target at proteins, because easily available protein-protein interaction (PPI) data can serve as an available resource for task relation extraction.

Our dataset is based on the *Large Scale Comparison (LSC)* dataset proposed by Mayr et al. (2018), which is filtered from the ChEMBL-20 database Mendez et al. (2018). We account for a subset of 725 tasks which are protein-targeting. For each of these tasks, we collect the UniProt IDs Consortium (2018) of the targeted proteins and combine all of them into a UniProt ID set. We then query the STRING database Szklarczyk et al. (2019) to obtain PPI scores for all pairs of proteins in the set. With the collected PPI scores, we then heuristically define the edge weights w_{ij}, *i.e.*, task relation score, for task t_i and t_j in the task relation graph to be $\max\{\text{PPI}(s_i, s_j) : s_i \in S_i, s_j \in S_j\}$, where S_i denotes the protein set of task t_i. Therefore, the task relation graph proposed has a high quality to reveal the actual pharmaceutical effects for the molecular drugs.
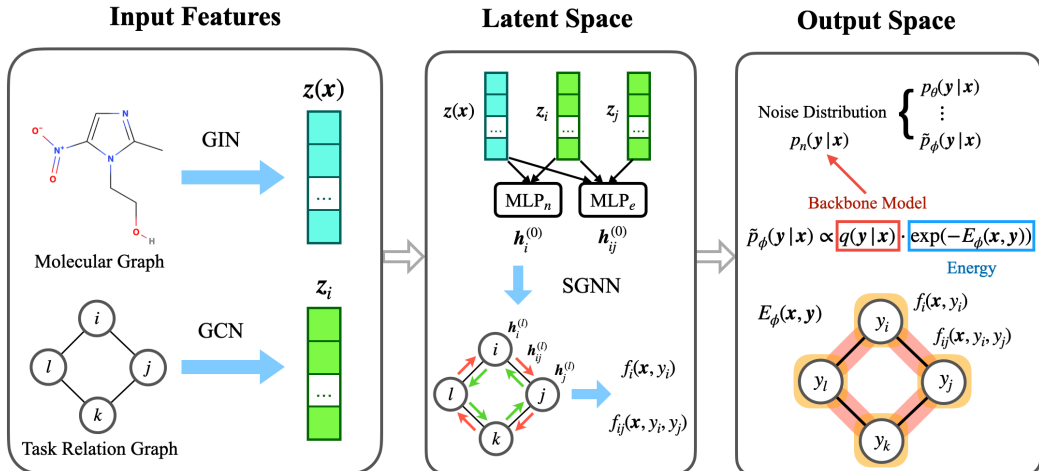
Figure 1: Pipeline of SGNN-EBM. We first obtain molecule and task embedding via GIN and GCN. Then, they are used to learn the latent representation for each task via a GNN model in the latent space. In SGNN-EBM, an SGNN model is used to model the task relation graph in the latent space and EBM learns the task distribution in the output space. The likelihood also applies the energy tilting term, which takes the same empirical distribution as the noise distribution for NCE.

Table 1: Statistics about 3 benchmark datasets with explicit task relation, filtered by 3 thresholds. Threshold means the number of non-missing labels for each molecule/task.

Threshold	# Molecules	# Tasks	Sparsity
10	13,004	382	5.76%
50	932	152	66.70%
100	518	132	92.87%

As the experiment-based LSC dataset is very sparsely-labeled - only 0.78% of elements of the molecule-task matrix have a label of *active* or *inactive*, we densify the molecule-task label matrix by iteratively filtering out molecules and tasks whose number of labels is lower than a certain threshold. By setting the threshold value to 10, 50 and 100, we obtain 3 benchmark datasets with different level of data sparsity. The statistics of the benchmark datasets are listed in Table 1, and more detailed dataset generation procedure can be found in Appendix A.

5 Method: Structured Task Modeling

5.1 Overview

The mainstream multi-task learning (MTL) methods Kendall et al. (2018); Chen et al. (2018); Liu et al. (2019c); Yu et al. (2020) typically learn the task relation implicitly, which can guide to balance tasks during training. While in this paper, we focus on a novel setting where the *task relation graph* is explicitly given and the goal is to better model such relation graph. We

first propose a dataset with an explicit task relation graph in Section 4, then in this section, we introduce two *structured MTL* approaches to modeling the task relation in the *latent* and *output* space respectively.

In the latent space, we propose to learn effective task representations with a State GNN (SGNN) on the task relation graph so that the learnt representations can capture the similarity between tasks. The property y_i in each task i can be independently predicted with the molecule information and its own task representation. More specifically, we can define the distribution as:

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}, \mathcal{G}) = \prod_{i=0}^{T-1} p_\theta(y_i|\boldsymbol{x}, \mathcal{G}), \qquad (2)$$

where $p_\theta(y_i|\boldsymbol{x}, \mathcal{G})$ is the prediction on the i-th task. We present this method in section 5.2. More detailed description of GNN can be found in Appendix D.

One limitation of the SGNN is that it ignores the dependency between task labels y_i. To handle this issue, we further propose to model the task dependency in the output space and solve it under the energy-based model (EBM) framework, as a structured prediction problem. The joint distribution of $\boldsymbol{y}$ can be modeled with EBM as:

$$p_\phi(\boldsymbol{y}|\boldsymbol{x}, \mathcal{G}) = \frac{\exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}; \mathcal{G}))}{Z_\phi}, \qquad (3)$$

where $E_\phi(\boldsymbol{x}, \boldsymbol{y}; \mathcal{G})$ is the energy function with flexible format. The noise contrastive estimation (NCE) is used to learn the EBM efficiently, and an outline of these methods is depicted in Figure 1.

Then we combine the advantages of both approaches by accounting the SGNN for energy function in EBM. Thus we are able to model the task relation in both the latent and output space, and we name this method as **SGNN-EBM** for solving structured MTL problems.

5.2 Modeling Task Relation in Latent Space

We propose *State GNN (SGNN)* to model the task relation in the latent space. The task relation is implicitly encoded in the learnt representations, and the final predictions are made independently for each task. We illustrate the pipeline of this model as follows.

Node- and Edge-Level Inputs. We first encode the molecules and tasks into the *embedding* space. For molecules, we adopt graph isomorphism network (GIN) Xu et al. (2018), and the molecule embedding is $z(x) \in \mathbb{R}^{d_m}$, where d_m is the embedding dimension. Then for tasks, we use one-hot encodings (w.r.t. the task index) and pass them through a graph convolutional network (GCN) Kipf and Welling (2016) to get task embedding: $z^{(i)} \in \mathbb{R}^{d_e}, \forall i \in \{0, 1, \ldots, T-1\}$, where d_t is the task embedding dimension. More details of GIN and GCN can be found in Appendices B and C. Given the molecule and task embeddings, we will use them to construct the node- and edge-level inputs to SGNN as:

$$
\begin{aligned}
h_i^{(0)}(x) &= \text{MLP}_n^{(0)}(z(x) \oplus z^{(i)}) \\
h_{ij}^{(0)}(x) &= \text{MLP}_e^{(0)}(z(x) \oplus z^{(i)} \oplus z^{(j)}),
\end{aligned}
\tag{4}
$$

where $\oplus$ is the concatenation of two tensors. $\text{MLP}_n^{(0)} : \mathbb{R}^{d_m+d_t} \to \mathbb{R}^{C \times d}$ and $\text{MLP}_e^{(0)} : \mathbb{R}^{d_m+2d_t} \to \mathbb{R}^{C \times C \times d}$ are two multi-layer perceptron (MLP) layers, operating on the node- and edge-level respectively. d is the dimension of the latent representation and $C = 2$ is the class number, and it also represents the states on each node and edge in SGNN. The node- and edge-level inputs in Equation (4) will then be fed to SGNN.

State GNN (SGNN). Different from the mainstream GNN models, SGNN has C and $C \times C$ states on each node and edge respectively, where each state delegates the representation for the corresponding label. Concretely, every node state represents the task w.r.t. the corresponding label, and edge state is composed of the pair-wise states from the two endpoint nodes. Thus, the representation for each node and edge state is defined as:

$$
\begin{aligned}
h_i^{(0)}(x, y_i) &= h_i^{(0)}(x)[y_i] \\
h_{ij}^{(0)}(x, y_i, y_j) &= h_{ij}^{(0)}(x)[y_i, y_j].
\end{aligned}
\tag{5}
$$

In this way, the representations of nodes and edges can well capture the information of each node label as well as the pairwise labels on an edge.

Such state-level view builds up the smallest granularity in SGNN. For example, during **message-passing propagation**, the key function in SGNN, only information with the same state will be exchanged between nodes and edges. Specifically, the propagation on the l-th layer is:

$$
\begin{aligned}
h_i^{(l+1)}&(x, y_i) = \\
&\text{MPNN}_n^{(l+1)}\Big(h_i^{(l)}(x, y_i), \big\{h_{ij}^{(l)}(x, y_i, y_j) \mid \forall j, y_j\big\}\Big) \\
h_{ij}^{(l+1)}&(x, y_i, y_j) = \\
&\text{MPNN}_e^{(l+1)}\Big(h_i^{(l)}(x, y_i), h_j^{(l)}(x, y_j), h_{ij}^{(l)}(x, y_i, y_j)\Big),
\end{aligned}
\tag{6}
$$

where MPNN stands for the message-passing neural network layer Gilmer et al. (2017). MPNN_n is doing node aggregation by gathering information from edges with the same node state y_i; MPNN_e stores the messages for each state pair (y_i, y_j) with the corresponding state information from the nodes. After repeating Equation (6) L times, we obtain the latent representation for each task given the molecule.

Independent Label Prediction. Finally, we make predictions for each task independently as Equation (2). For each task i, we first get the node representation by concatenating the two state representations, after which we apply a readout function R:

$$
f_i(x) = R(\{h_i^{(l)}(x, 0) \oplus h_i^{(l)}(x, 1) \mid l = 1, \ldots, L\}), \tag{7}
$$

where $R : \mathbb{R}^{2dL} \to \mathbb{R}$ is an MLP layer. Because $C = 2$ is the binary classification, the label distribution is defined via a sigmoid function, *i.e.*, $p(y_i = 1|x, \mathcal{G}) = \text{sigmoid}(f_i(x))$. The loss function is the binary cross entropy function over all T tasks:

$$
\mathcal{L} = \sum_{i=0}^{T-1} \log p(y_i|x, \mathcal{G}). \tag{8}
$$

Despite the effectiveness of learning task representations, SGNN fails to directly model the task dependency when making predictions as different task labels are predicted separately. To address this issue, we next propose a general method for modeling the task dependency under the structured prediction framework, which is able to predict task labels collectively to improve the result.

5.3 Modeling Task Relation in Output Space

The aforementioned MTL methods are predicting each task independently. However, there also exists a task distribution in the output space, *i.e.*, $p(y = y_0, y_1, \ldots, y_{T-1}|x)$. 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:

$$E_\phi(\boldsymbol{x}, \boldsymbol{y}) = -\sum_{i=0}^{T-1} f_i(\boldsymbol{x}, y_i) - \lambda \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j), \quad (9)$$

where λ is a weighting coefficient. Thus the conditional probability under the EBM framework is defined as:

$$p_\phi(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\left(\sum_i f_i(\boldsymbol{x}, y_i) + \sum_{ij} f_{ij}(\boldsymbol{x}, y_i, y_j)\right)}{Z_\phi}. \quad (10)$$

Activation Function. We apply the activation function $\sigma(\cdot) = \log(\mathrm{softmax}(\cdot))$ on the first- and second-order factors. Then the readout function is $\tilde{R}(\cdot) = \log(\mathrm{softmax}(\mathrm{MLP}(\cdot)))$, where the softmax function is applied on the label/state space of each task and each task pair. The softmax function normalizes the scores of different label candidates, allowing us to compare them in the same range between 0 and 1. The logarithm function further scales the energy to 0 to ∞, which is a common practice in EBM.

Energy Tilting Term. We have introduced EBM to model task relations in output space. However, directly training the energy-based model is still a challenging problem. To alleviate this issue, we leverage the energy tilting term from Dai et al. (2014); Xie et al. (2016); Nijkamp et al. (2020); Arbel et al. (2020), which takes EBM in the form of a correction or an exponential tilting of a pre-trained backbone model $q(\boldsymbol{y}|\boldsymbol{x})$. The pre-trained backbone model acts as a base model, and the energy function $\exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))$ tries to tilt the base model for better results, yielding an integrated model as: $\tilde{p}_\phi(\boldsymbol{y}|\boldsymbol{x}) \propto q(\boldsymbol{y}|\boldsymbol{x}) \cdot \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))$, where the integrated model $\tilde{p}_\phi(\boldsymbol{y}|\boldsymbol{x})$ is named the *energy tilting distribution*. We will illustrate how to combine this energy tilting term in the learning and inference below.

5.4 SGNN-EBM

Then we will combine the structured modeling on both latent and output space together. As mentioned before, the energy function in EBM can have flexible formulation LeCun et al. (2006); thus, we may as well parameterize it by adopting the node- and edge-level representation from SGNN. With minor modifications we have:

$$\begin{aligned} f_i(\boldsymbol{x}, y_i) &= \tilde{R}(\{\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) \mid l = 1, \ldots, L\}) \\ f_{ij}(\boldsymbol{x}, y_i, y_j) &= \tilde{R}(\{\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) \mid l = 1, \ldots, L\}), \end{aligned} \quad (11)$$

where $\tilde{R} : \mathbb{R}^{dL} \to \mathbb{R}$ is a readout function defined as $\tilde{R} = \sigma(\mathrm{MLP}(\cdot))$ and $\sigma(\cdot)$ is the activation function.

Equation (11) is mapping the node and edge representations to scalars (or energies) indiced with the corresponding node and edge label.

As the number of message-passing layers L increases, the SGNN-based energy function (Equation (11)) can be seen as a general form to capture the higher-order dependency. However, according to the energy function decomposition in Equation (9), only first- and second-order factors are considered during the EBM learning and inference. This discrepancy may raise some potential concern, and we carry on an ablation study in Section 6.3, where we empirically prove that slightly increasing L can be beneficial for the generalization performance. Yet, this is still worth further exploration in the future.

In the following sections, we will introduce how to do NCE learning and Gibbs sampling inference for our proposed SGNN-EBM model.

5.4.1 Learning

The learning process aims at optimizing ϕ to maximize the data likelihood. However, the problem is nontrivial as the partition function Z_ϕ is intractable. Our approach addresses this by using noise contrastive estimation (NCE) Gutmann and Hyvärinen (2010), which casts the problem of maximizing log-likelihood into a contrastive learning task. We first take the normalization constant Z_ϕ in Equation (1) as a learned scalar parameter. Then we transform the EBM learning into a binary classification problem by maximizing the following objective:

$$\begin{aligned} \mathcal{L}_{NCE} = &\mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} \\ &+ \mathbb{E}_{\boldsymbol{y} \sim p_{\mathrm{data}}} \log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})}, \end{aligned} \quad (12)$$

where p_{data} is the underlying data distribution, p_ϕ is the model distribution to approximate data distribution, and p_n is a noise distribution, whose samples serve as negative examples in the contrastive learning objective. Ideally, p_ϕ will be trained to approximate p_{data} for any noisy distribution. Yet in practice, the noise distribution should be close to the data distribution to facilitate the mining of hard negative samples. In addition Mnih and Teh (2012), given an expressive energy function, we can fix $Z_\phi = 1$ and the resulting learned EBM will be self-normalized.

NCE with Tilting Term. The above objective function seems complicated. Nevertheless, it will become more concise as we 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 (12). With the self-normalized partition function, the NCE learning with

Table 2: Main MTL results. All datasets are split into 8-1-1 for train, valid, and test respectively. For each method, we run 5 seeds and report the mean and standard deviation. The best performance is **highlighted**.

Method	p_n	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	–	72.49 ± 0.53	79.68 ± 0.98	78.71 ± 1.93
GradNorm	–	75.17 ± 0.77	79.46 ± 1.27	78.75 ± 1.60
DWA	–	72.45 ± 1.31	79.35 ± 0.68	78.21 ± 2.31
LBTW	–	75.21 ± 0.49	79.52 ± 0.56	79.07 ± 0.99
SGNN	–	77.90 ± 0.88	79.67 ± 0.87	80.19 ± 0.67
SGNN-EBM	SGNN (Fixed)	78.04 ± 0.73	80.34 ± 1.08	80.48 ± 1.93
SGNN-EBM	SGNN (Adaptive)	$\mathbf{78.35 \pm 1.07}$	$\mathbf{80.54 \pm 1.02}$	$\mathbf{81.15 \pm 0.59}$

energy tilting term can be written as:

$$
\begin{aligned}
\tilde{\mathcal{L}}_{NCE} = &\ \mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{1}{1 + \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))} \\
&+ \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \log \frac{1}{1 + \exp(E_\phi(\boldsymbol{x}, \boldsymbol{y}))}.
\end{aligned}
\tag{13}
$$

In this new objective function, we only need to draw samples from the noise distribution without computing their density, which is easy to operate. More detailed derivations are attached in Appendix F.

The Choice of Noise Distribution. One key component in NCE training is the choice of the noise distribution, p_n. NCE works for any given noise distribution, yet the algorithm empirically converges faster if the noise distribution p_n can stay close to the model distribution p_ϕ Song and Kingma (2021). In the experiment, we propose two options for selecting the noise distributions. (1) We use a pre-trained model to be a *fixed* noise distribution, *e.g.*, the SGNN proposed in Section 5.2 and $p_n = p_\theta$. (2) We adopt an *adaptive* noise distribution, and start with a pre-trained model. The difference is that after training with this pre-trained noise distribution for a few epochs, we will gradually update the noise distribution with our learned model, *i.e.*, updating p_n with the latest $\tilde{p}_\phi$. The second idea aligns well with the curriculum learning Bengio et al. (2009), a learning process starting with easy data to hard data. Thus another way to interpret the adaptive noise distribution is that, we start with a simple distribution (from a pre-trained model distribution) and gradually using harder distribution (from the latest model distribution). We investigate the effect on the choices of noise distributions for NCE learning in the ablation study in Section 6.2.

Imputation for Missing Labels. For the SGNN-EBM training proposed in Section 5.4, we use the task distribution for predicting each data point, $p_\phi(\boldsymbol{y}|\boldsymbol{x})$, but some tasks do not have valid labels due to the label sparsity, as discussed in Sections 1 and 4. In SGNN-EBM, we propose to use the backbone model, q, to fill in the missing labels so as to calculate the

probability. This strategy shares similar idea to the EM algorithm Neal and Hinton (1998), which allows us to maximize a variational lower bound of the data likelihood. Empirically, experiment results help support this imputation strategy, yet, this is still work investigating in the future.

5.4.2 Inference

The inference procedure aims at computing the marginal distribution for each task, which can be further utilized for the label prediction for each task. The main challenge is how to calculate the intractable partition function during inference. We propose to approximate the distribution via Gibbs sampling Geman and Geman (1984). Gibbs sampling is a classic MCMC-based inference method and the core idea is to generate samples by sweeping through each variable to a sample with the remaining variables fixed.

To adopt Gibbs sampling in our setting, for each data and T labels, $(\boldsymbol{x}, y_0, \dots, y_{T-1})$, we iteratively sample label for each task with other labels fixed. The update function at each iteration is:

$$
\begin{aligned}
&p_\phi(y_i | \boldsymbol{y}_{-i}, \boldsymbol{x}) \\
&= \frac{\exp\left(f_i(\boldsymbol{x}, y_i) + \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j)\right)}{\sum_{y_i=0}^{C-1} \exp\left(f_i(\boldsymbol{x}, y_i) + \sum_{\langle i,j \rangle \in \mathcal{G}} f_{ij}(\boldsymbol{x}, y_i, y_j)\right)},
\end{aligned}
\tag{14}
$$

where $\boldsymbol{y}_{-i}$ denotes all T task labels except the i-the task. Then we take this as the tilting term, and 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.

6   Experiment Results

6.1   Main Results

Baselines. As described in Section 2, the memory cost of architecture-specific MTL methods (*e.g.*, Bypass network) is $O(T)$, and pair-wise architecture-agnostic MTL methods (RMTL Liu (2018), PC-Grad Yu et al. (2020), GradVac Wang et al. (2021)) have $O(T^2)$ time complexity. Both are infeasible in the

Table 3: The effect of different noise distributions p_n in NCE. Here all the noise distributions are fixed.

Method	p_n	ChEMBL 10	ChEMBL 50	ChEMBL 100
MTL	–	74.83 ± 0.61	79.37 ± 1.76	77.78 ± 1.59
UW	–	72.49 ± 0.53	79.68 ± 0.98	78.71 ± 1.93
GradNorm	–	75.17 ± 0.77	79.46 ± 1.27	78.75 ± 1.60
DWA	–	72.45 ± 1.31	79.35 ± 0.68	78.21 ± 2.31
LBTW	–	75.21 ± 0.49	79.52 ± 0.56	79.07 ± 0.99
SGNN	–	77.90 ± 0.88	79.67 ± 0.87	80.19 ± 0.67
SGNN-EBM	Uniform	58.66 ± 4.65	73.55 ± 0.61	75.49 ± 1.64
SGNN-EBM	MTL	75.71 ± 0.41	79.96 ± 1.41	78.41 ± 1.37
SGNN-EBM	UW	74.36 ± 0.87	80.26 ± 0.67	79.12 ± 1.79
SGNN-EBM	GradNorm	75.83 ± 0.73	80.18 ± 1.04	79.34 ± 1.31
SGNN-EBM	DWA	75.22 ± 1.16	80.18 ± 0.74	79.01 ± 1.94
SGNN-EBM	LBTW	76.16 ± 0.54	80.04 ± 0.50	79.68 ± 0.93
SGNN-EBM	SGNN	78.04 ± 0.73	80.34 ± 1.08	80.48 ± 1.93

large-scale MTL setting (w.r.t. the number of tasks), so we exclude them in the experiments. For the baseline methods, we include standard single-task learning (STL), standard multi-task learning (MTL), Uncertainty Weighing (UW) Kendall et al. (2018), Grad-Norm Chen et al. (2018), Dynamic Weight Average (DWA) Liu et al. (2019b), and Loss-Balanced Task Weighting (LBTW) Liu et al. (2019c).

Our Methods. We first test SGNN, which *only* models the task relation graph in the latent space. On the other hand, EBM is very sensitive to the noise distribution, leading to unstable performance. Thus we will not test it separately as SGNN, and two following ablation studies can reveal more insights for it. Then we test our main proposal, SGNN-EBM. SGNN-EBM models the task relation graph in both the latent and output space under the EBM framework, where the energy function is defined as the SGNN. We explore two noise distributions in the NCE learning steps: (2.1) the first is a fixed pre-trained SGNN, $p_n = p_\theta$; (2.2) the second is taking the pre-trained SGNN, $p_n = p_\theta$, 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 MTL 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 settings like molecular property prediction in drug discovery.

Observation. We adopt the proposed dataset with three thresholds introduced in Section 4 for experiments. The main results are in Table 2. First we can see all the MTL methods are better than the STL, which matches with the common acknowledgement 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 us-

ing SGNN reaches a good performance compared to all MTL baselines, while combining it with the EBM in the output space, *i.e.*, SGNN-EBM, can reach the best performance on all datasets. For the two SGNN-EBM models, they are consistently better than the SGNN model, while adaptively updated noise distribution can reach best performance. All these observations deliver an important message: structured task modeling is useful in MTL, and SGNN-EBM is an effective solution in achieving this goal.

6.2 Ablation Study 1: The Effect of p_n

In the NCE learning of EBMs, the performance highly depends on the noise distribution p_n. In Table 2 we show that the best method is SGNN-EBM with SGNN as both the energy function and noise distribution. Indeed we can take one uniform distribution and all pre-trained models (prior distribution) as the noise distribution, and we show that NCE-based structured prediction can obtain consistent performance gain when comparing to the corresponding prior distribution.

As in Table 3, the improvement by structured prediction is not huge but consistent on all datasets: for each pre-trained model, its SGNN-EBM counterpart can consistently improve the performance by taking it as a prior distribution in NCE learning. Such consistency consolidates the effectiveness of our solution.

6.3 Ablation Study 2: The Effect of L

Table 4: The Effect of Layer Number in SGNN.

# layer	ChEMBL 10	ChEMBL 50	ChEMBL 100
0	77.45 ± 1.03	80.63 ± 0.80	80.82 ± 2.09
2	77.56 ± 1.00	80.78 ± 0.85	81.13 ± 2.04
4	76.98 ± 0.91	80.42 ± 0.82	81.06 ± 2.09

We test SGNN-EBM* with $L = 0, 2, 4$ with all the other hyper-parameters fixed, where L is the number of layers in GNN. In the NCE learning, we are adapting the noise distributions from a pre-trained SGNN

model, p_θ. The parameter L reflects that each node (molecule-task) in the graph aggregates features from its L-hop neighborhood.

As observed in Table 4, the SGNN-EBM improves the performance slightly with larger L in SGNN owing to the ability to model longer-term dependencies among labels. However, as L increases, the performance will drop instead. One possible explanation is that the inference method, Gibbs Sampling, defined in Section 5.3 only considers first- and second-order factors, thus it fails to capture the long-term dependencies.

7 Conclusion and Future Direction

In this paper, we propose a novel research problem of MTL for molecular property prediction with an explicit task relation graph. We propose a novel approach to modeling the task relations in both the *latent* and *output* space. Experimental results demonstrate that SGNN-EBM outperforms competitive baselines.

We want to highlight that SGNN-EBM can fit to broad MTL problems, as long as the explicit task relation is accessible. But as the first step along this direction, we would like to start from a modest setting with assurance from the oracle, like explicit task relation from drug discovery domain. In addition, structured task modeling opens a new and promising research venue. For example, some MTL methods (RMTL Liu (2018), GradVac Wang et al. (2021)) are able to extract the pairwise similarity to compose a task relation graph; yet, this view point is unexplored and would be interesting to combine with SGNN-EBM as the next step.

Acknowledgements

References

Alnammi, M., Liu, S., Ericksen, S. S., Ananiev, G. E., Voter, A. F., Guo, S., Keck, J. L., Hoffmann, F. M., Wildman, S. A., and Gitter, A. (2021). Evaluating scalable supervised learning for synthesize-on-demand chemical libraries.

Arbel, M., Zhou, L., and Gretton, A. (2020). Generalized energy based models. *arXiv preprint arXiv:2003.05033*.

Belanger, D. and McCallum, A. (2016). Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992. PMLR.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803.

Consortium, T. U. (2018). UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515.

Dahl, G. E., Jaitly, N., and Salakhutdinov, R. (2014). Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*.

Dai, J., Lu, Y., and Wu, Y.-N. (2014). Generative modeling of convolutional neural networks. *arXiv preprint arXiv:1412.6296*.

Demirel, M. F., Liu, S., Garg, S., and Liang, Y. (2021). An analysis of attentive walk-aggregating graph neural networks. *arXiv preprint arXiv:2110.02667*.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28:2224–2232.

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

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Gutmann, M. and Hyvärinen, A. (2010). 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.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2019a). Pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2019b). Strategies for

pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Jain, A., Upreti, M., and Jyothi, P. (2018). Improved accented speech recognition using accent embeddings and multi-task learning. In *Interspeech*, pages 2454–2458.

Kendall, A., Gal, Y., and Cipolla, R. (2018). 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.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Lee, K. and Kim, D. (2019). In-silico molecular binding prediction for human drug targets using deep neural multi-task learning. *Genes*, 10(11):906.

Liu, S. (2018). Exploration on deep drug discovery: Representation and learning. *Master's Thesis*, TR1854.

Liu, S., Alnammi, M., Ericksen, S. S., Voter, A. F., Ananiev, G. E., Keck, J. L., Hoffmann, F. M., Wildman, S. A., and Gitter, A. (2018). Practical model selection for prospective virtual screening. *Journal of chemical information and modeling*, 59(1):282–293.

Liu, S., Demirel, M. F., and Liang, Y. (2019a). N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8464–8476. Curran Associates, Inc.

Liu, S., Johns, E., and Davison, A. J. (2019b). End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880.

Liu, S., Liang, Y., and Gitter, A. (2019c). Loss-balanced task weighting to reduce negative transfer in multi-task learning.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. (2021). Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*.

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. (2017). 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.

Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wegner, J. K., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. (2018). Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M., Mosquera, J., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C., Segura-Cabrera, A., Hersey, A., and Leach, A. (2018). ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940.

Merck (2012). Merck molecular activity challenge. *https://www.kaggle.com/c/MerckActivity*.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Nijkamp, E., Gao, R., Sountsov, P., Vasudevan, S., Pang, B., Zhu, S.-C., and Wu, Y. N. (2020). Learning energy-based model with flow-based backbone by neural transport mcmc. *arXiv preprint arXiv:2006.06897*.

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., and Pande, V. (2015). Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*.

Ramsundar, B., Liu, B., Wu, Z., Verras, A., Tudor, M., Sheridan, R. P., and Pande, V. (2017). Is multitask deep learning practical for pharma? *Journal of chemical information and modeling*, 57(8):2068–2076.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. (2020). Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33.

Søgaard, A. and Goldberg, Y. (2016). 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. Association for Computational Linguistics.

Song, Y. and Kingma, D. P. (2021). How to train your energy-based models. *arXiv preprint arXiv:2101.03288.*

Szklarczyk, D., Gable, A. L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N. T., Morris, J. H., Bork, P., et al. (2019). String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613.

Unterthiner, T., Mayr, A., Klambauer, G., Steijaert, M., Wegner, J. K., Ceulemans, H., and Hochreiter, S. (2014). Deep learning as an opportunity in virtual screening. *Advances in neural information processing systems*, 27.

Wang, Z., Tsvetkov, Y., Firat, O., and Cao, Y. (2021). Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. (2016). A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826.*

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020). Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782.*

Zhang, Y., Zhang, P., and Yan, Y. (2017). Attention-based lstm with multi-task learning for distant speech recognition. In *Interspeech*, pages 3857–3861.

Supplementary Material:
Structured Multi-task Learning for Molecular Property Prediction

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 Mayr et al. (2018), 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 Hu et al. (2019b). 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 $CH_3NH_3^+Cl^-$, we retain only the organic compound after removing HCl, in this case 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 Mendez et al. (2018) and STRING Szklarczyk et al. (2019) databases. The details are as follows:

1. The LSC dataset Mayr et al. (2018) 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 Consortium (2018) 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 $\max\{PPI(s_i, s_j) : s_i \in S_i, s_j \in S_j\}$, 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.

The statistics of the resulting dataset with three thresholds can be found at Table 1.

B GIN for Molecule Embedding

The Graph Isomorphism Network (GIN) is proposed in Xu et al. (2018). 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 Hu et al. (2019b). With this customized GIN as the base model, plus pre-training techniques, Hu et al. (2019b) 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 3, 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:

$$z_i^{(k+1)} = \text{MLP}_{\text{atom}}^{(k+1)}\Big(z_i^{(k)} + \sum_{j \in \mathcal{N}(i)} \big(z_j^{(k)} + \text{MLP}_{\text{bond}}^{(k+1)}(E_{ij})\big)\Big), \tag{15}$$

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:

$$z(\boldsymbol{x}) = \frac{1}{N} \sum_i z_i^{(K)} \tag{16}$$

C GCN for Task Embedding

We use graph convolutional network (GCN) Kipf and Welling (2016) 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 $\boldsymbol{e}_i \in \mathbb{R}^{d_t \times 1}, \forall i \in \{0, 1, \dots, T-1\}$, where d_t is the task embedding dimension. $\boldsymbol{E} = \{\boldsymbol{e}_0, \boldsymbol{e}_1, \dots, \boldsymbol{e}_{T-1}\}^T \in \mathbb{R}^{T \times d_t}$ 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 $\boldsymbol{z}^{(i)} = \text{GCN}(\boldsymbol{E})_i, \forall i \in \{0, 1, \dots, T-1\}$.

D SGNN for Modeling Latent Space

In this section, we give a detailed illustration of our proposed State GNN (SGNN) model in Section 5.2. The general pipeline is shown in Figure 2.

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

$$\begin{aligned}
\boldsymbol{h}_i^{(0)}(\boldsymbol{x}) &= \text{MLP}_n^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)}) \\
\boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}) &= \text{MLP}_e^{(0)}(\boldsymbol{z}(\boldsymbol{x}) \oplus \boldsymbol{z}^{(i)} \oplus \boldsymbol{z}^{(j)}),
\end{aligned} \tag{17}$$

and as discussed in Section 5.2, the biggest difference between SGNN and the mainstream GNN models is that in SGNN, each node and each edge has two and four state respectively, where each state of a node/edge is the representation for the corresponding 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:

$$\begin{aligned}
\boldsymbol{h}_i^{(0)}(\boldsymbol{x}, y_i) &= \boldsymbol{h}_i^{(0)}(\boldsymbol{x})[y_i] \\
\boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x}, y_i, y_j) &= \boldsymbol{h}_{ij}^{(0)}(\boldsymbol{x})[y_i, y_j].
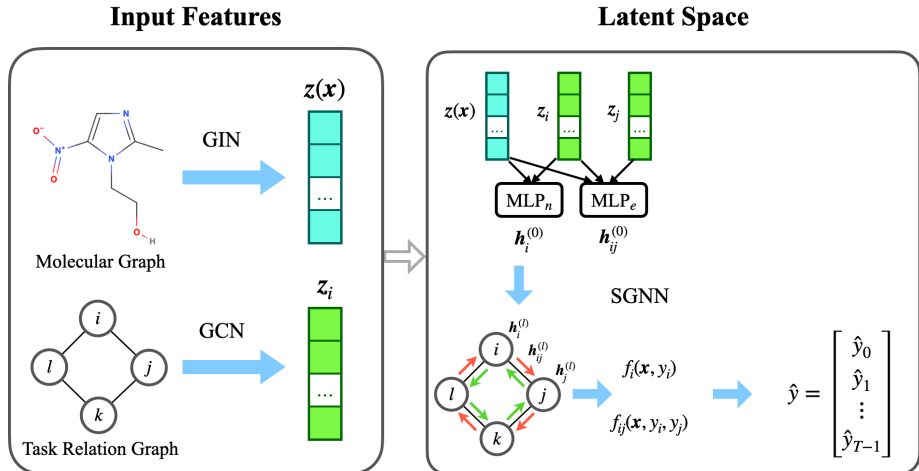\end{aligned} \tag{18}$$

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.

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 5.2, for the node-level propagation we have:

$$
\begin{aligned}
\boldsymbol{h}_i^{(l+1)}(\boldsymbol{x}, y_i) &= \mathrm{MPNN}_n^{(l+1)}\Big(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \{\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) \mid \forall j, y_j\}\Big) \\
&= \mathrm{MLP}_n^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) + \sum_{j \in \mathbb{N}(i)} \sum_{y_j=0}^{C-1} \boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j)\right),
\end{aligned}
\tag{19}
$$

and for the edge-level propagation, we have:

$$
\begin{aligned}
\boldsymbol{h}_{ij}^{(l+1)}(\boldsymbol{x}, y_i, y_j) &= \mathrm{MPNN}_e^{(l+1)}\Big(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i), \boldsymbol{h}_j^{(l)}(\boldsymbol{x}, y_j), \boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j)\Big) \\
&= \mathrm{MLP}_e^{(l+1)}\left(\boldsymbol{h}_{ij}^{(l)}(\boldsymbol{x}, y_i, y_j) + \mathrm{MLP}_a^{(l+1)}\left(\boldsymbol{h}_i^{(l)}(\boldsymbol{x}, y_i) + \boldsymbol{h}_j^{(l)}(\boldsymbol{x}, y_j)\right)\right),
\end{aligned}
\tag{20}
$$

where $\mathrm{MLP}_n^{(l+1)}(\cdot)$, $\mathrm{MLP}_e^{(l+1)}(\cdot)$ and $\mathrm{MLP}_a^{(l+1)}(\cdot)$ 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

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 Xu et al. (2018), and we follow the hyperparameter used in Hu et al. (2019b). The base graph neural network for task embedding is GCN Kipf and Welling (2016). We have more detailed description of GIN and GCN in Appendices B and C. The hyperparameter tuning for all baseline methods and SGNN base models in Appendix E.1.

E.1   Hyperparameter Tuning

We list the hyperparameters for baselines models and our proposed models in Table 5, including MTL, UW Kendall et al. (2018), GradNorm Chen et al. (2018), Dynamic Weight Average (DWA) Liu et al. (2019b), and Loss-Balanced Task Weighting (LBTW) Liu et al. (2019c), SGNN in Section 5.2, SGNN-EBM in Section 5.4.

Table 5: Hyperparameters for baselines and our models.

Model	Hyperparameters	Values
MTL	Epochs	[100, 200]
UW	Epochs	[100, 200]
GradNorm	Epochs α	[100, 200] [0.1, 0.2, 0.5]
DWA	Epochs T	[100, 200] [0.2]
LBTW	Epochs α	[100, 200] [0.1, 0.2, 0.5]
SGNN	Epochs d # GIN Layer # GCN Layer # SGNN Layer	[200, 500] [50, 100] [5] [0, 2] [2]
SGNN-EBM	Epochs Fixed-Noise Distribution Epochs d # GIN Layer # GCN Layer # SGNN Layer λ	[200, 500] [200, 300, 400, 1000] [50, 100] [5] [0, 2] [0, 2, 4] [0.1, 1]

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 5.3. When applying the backbone model for noise distribution, *i.e.*, $p_n = q$, and adopting the self-normalization ($Z = 1$), the loss can be rewritten as:

$$
\begin{aligned}
\hat{\mathcal{L}}_{NCE} &= \mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_\phi(\boldsymbol{y}|\boldsymbol{x})} \\
&= \mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_n(\boldsymbol{y}|\boldsymbol{x}) \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))} + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \log \frac{p_n(\boldsymbol{y}|\boldsymbol{x}) \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))}{p_n(\boldsymbol{y}|\boldsymbol{x}) + p_n(\boldsymbol{y}|\boldsymbol{x}) \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))} \quad (21) \\
&= \mathbb{E}_{\boldsymbol{y} \sim p_n} \log \frac{1}{1 + \exp(-E_\phi(\boldsymbol{x}, \boldsymbol{y}))} + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \log \frac{1}{1 + \exp(E_\phi(\boldsymbol{x}, \boldsymbol{y}))}.
\end{aligned}
$$

For more detailed derivations, please check Song and Kingma (2021); Liu et al. (2021).