# Pre-training Graph Neural Networks

**Weihua Hu**[1*], **Bowen Liu**[2*], **Joseph Gomes**[3], **Marinka Zitnik**[1],
**Percy Liang**[1], **Vijay S. Pande**[3], **Jure Leskovec**[1]
[1]Department of Computer Science, [2]Department of Chemistry, [3]Department of Bioengineering
Stanford University
{weihuahu,liubowen,joegomes,pande}@stanford.edu,
{marinka,pliang,jure}@cs.stanford.edu

## Abstract

Many applications of machine learning in science and medicine, including molecular property and protein function prediction, can be cast as problems of predicting some properties of graphs, where having good graph representations is critical. However, two key challenges in these domains are (1) extreme scarcity of labeled data due to expensive lab experiments, and (2) needing to extrapolate to test graphs that are structurally different from those seen during training. In this paper, we explore pre-training to address both of these challenges. In particular, working with Graph Neural Networks (GNNs) for representation learning of graphs, we wish to obtain node representations that (1) capture similarity of nodes' network neighborhood structure, (2) can be composed to give accurate graph-level representations, and (3) capture domain-knowledge. To achieve these goals, we propose a series of methods to pre-train GNNs at both the node-level and the graph-level, using both unlabeled data and labeled data from related auxiliary supervised tasks. We perform extensive evaluation on two applications, molecular property and protein function prediction. We observe that performing only graph-level supervised pre-training often leads to marginal performance gain or even can worsen the performance compared to non-pre-trained models. On the other hand, effectively combining both node- and graph-level pre-training techniques significantly improves generalization to out-of-distribution graphs, consistently outperforming non-pre-trained GNNs across 8 datasets in molecular property prediction (resp. 40 tasks in protein function prediction), with the average ROC-AUC improvement of 7.2% (resp. 11.7%).

## 1 Introduction

Many problems in scientific domains, such as chemistry and biology, can be cast as the prediction of some property of a graph. For example, in chemistry, predicting chemical properties such as toxicity of molecules is important to accelerate drug discovery, where molecules are naturally represented by molecular graphs [9, 23, 13, 21, 52, 56]. In biology, identifying the functionality of proteins is important to find proteins that associate with a certain disease, where proteins are represented by local protein-protein interaction (PPI) graphs [57, 55]. Supervised learning of graphs, especially with Graph Neural Networks (GNNs) [26, 15, 59, 53], has shown promising results in these domains [64, 56, 13, 60].

Despite the promise, there remain two key challenges in applying GNNs to these scientific domains: (1) the extreme scarcity of labeled data, and (2) out-of-distribution prediction, where the graphs in the training set can have very different structural properties from those in the test set. First, task-specific data labeling is a costly and time consuming procedure typically performed in wet lab environments. Consequently, conventional GNNs can easily overfit to the small training datasets. Second, many

*The two first authors made equal contributions.

Preprint. Under review.

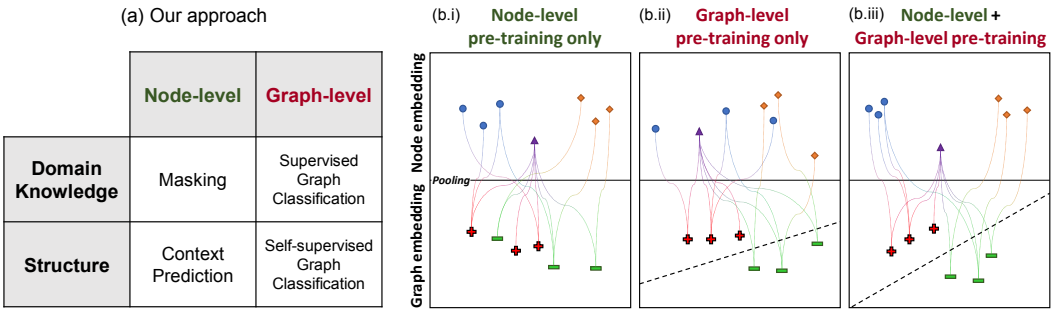



Figure 1: **(a)** Categorization of the pre-training methods for GNNs. Crucially, our methods, *i.e.*, Context Prediction, Masking, and graph-level supervised pre-training, cover both node-level and graph-level pre-training. **(b)** Node and graph embeddings obtained by different pre-training strategies. **(b.i)** When only node-level pre-training is used, nodes of different shapes (semantically different nodes) can be well separated, however, the node embeddings are not composable, and thus resulting graph embeddings (denoted by their classes, $+$ and $-$) that are created by pooling embeddings of individual nodes are not separable. **(b.ii)** With graph-level pre-training only, graph embeddings are well separated, however the embeddings of individual nodes do not necessarily capture their domain-specific semantics. **(b.iii)** High-quality node embeddings are such that nodes of different types are well separated, while at the same time, the embedding space is also composable. This allows for accurate and robust representations of entire graphs, which allows robust transfer of pre-trained models to a variety of downstream tasks.

scientific applications naturally involve out-of-distribution prediction. For example, one may want to predict chemical properties of newly-synthesized molecules which are often structurally different from the training molecules, or one may want to predict functionality of proteins from a new species that has different PPI network structure than previously studied species. Unfortunately, deep learning models are known to be extremely poor at out-of-distribution prediction [20, 16].

One promising approach to address the above two challenges is to pre-train GNNs using large amounts of easily-accessible unlabeled data as well as relatively easily-accessible labeled data that comes from related auxiliary tasks. For example, to perform a variety of downstream molecular property prediction tasks (*e.g.*, predicting toxicity or enzyme binding), one could use large amounts of easily-accessible molecule data to pre-train a model to capture chemistry domain knowledge, such as valency and chemical properties of different functional groups. Afterwards, very little hard-to-obtain labeled data would be needed to specialize the pre-trained model to the given downstream prediction task. Beyond its benefit of increasing data efficiency, pre-training could also improve predictive performance in out-of-distribution samples [16]. Therefore, pre-training could provide an attractive solution to the above two challenges. However, currently there exists no systematic investigation of potential strategies for pre-training GNNs and their effectiveness. In fact, as we see in our experiments, naïve pre-training of GNNs can often only give marginal increase in generalization performance on downstream tasks, and sometimes even worsen the performance compared to non-pre-trained models.

In this paper, we examine effective pre-training approaches to graph representation learning using GNNs. Our key observation is that GNNs obtain a representation of an entire graph by combining the following two steps [13]: (1) recursively aggregating neighboring information to obtain low-dimensional node embeddings that capture neighborhood structure, and (2) pooling/composing node embeddings to obtain a representation of the entire graph. Based on this observation, our goals for pre-training GNNs are to produce node embeddings that:

1. capture *structural similarity* of nodes' network neighborhoods.
2. are *composable* so that node embeddings can be pooled into an accurate graph-level representation.
3. capture *domain-knowledge* at the level of individual nodes and entire graphs.

Our approach to achieve these goals, which we briefly summarize below, is categorized in Figure 1 (a). Importantly, we aim to pre-train GNNs *both* at the level of *individual nodes* as well as *entire graphs*, which provides composability of embeddings as it builds a bridge between local node embeddings and the global graph embeddings, as illustrated in Figure 1 (b.iii). This is in contrast to naïve approaches to pre-train GNNs, *i.e.*, either only apply an (off-the-shelf) unsupervised node representation learning

technique, as illustrated in Figure 1 (b.i), or only perform supervised pre-training to predict auxillary properties of entire graphs, as illustrated in Figure 1 (b.ii).

**Context Prediction.** Most of the existing off-the-shelf unsupervised node representation learning methods are designed for node classification [14, 37, 49, 15, 25, 51] and enforce nearby nodes to have similar embeddings. This is not suited for representation learning of an entire graph, where capturing the *structural* similarity of local neighborhoods is more important [61, 42, 57]. To learn node embeddings that capture local graph structure, we introduce *Context Prediction*, which is a novel self-supervised node-level pre-training method that applies the distributional hypothesis [44, 31] to the graph domain. In particular, we use node embeddings to predict *surrounding graph structure*, so nodes that have similar surrounding graph structure will be mapped into similar representations.

**Masking.** To learn node embeddings that capture domain knowledge, we propose a novel self-supervised node-level pre-training method called *Masking*. In Masking, we randomly mask input node/edge attributes and let GNNs predict the masked attributes from the surrounding structure. For example, in the chemistry application, we can use node embeddings to predict atom types of masked atoms, as illustrated in Figure 2 (b). This forces the model to capture chemistry domain knowledge, such as valency and the electronic or steric properties of functional groups [30].

**Graph-level Prediction.** To learn composable node embeddings that are useful for downstream tasks, we can either perform (1) supervised graph-level pre-training on *domain-specific* auxiliary tasks, or (2) self-supervised pre-training to predict structural properties of the graphs. Here, to directly encode domain knowledge into graph embeddings, we take the first approach and combine our novel Context Prediction and Masking methods with *graph-level supervised pre-training.* This ensures that individual node embeddings are easily composed to obtain domain-specific representations of an entire graph, as illustrated in Figure 1 (b.iii).

We extensively evaluate the above pre-training methods and their combinations (one from node-level and another from graph-level) on two scientific applications of graph classification: molecular property prediction in chemistry, and protein function prediction in biology. First, on many downstream tasks, performing only graph-level supervised pre-training gives marginal performance gain or sometimes even *worsen* the generalization performance compared to a non-pre-trained model. This phenomenon is referred to as *negative transfer* [35, 43], which poses a significant problem when deploying pre-trained models to real world applications, and has been previously observed for multi-task learning of molecular property prediction tasks [39, 40, 54, 22]. When our *node-level* self-supervised pre-training is combined with the graph-level supervised pre-training, negative transfer is completely avoided across all the 8 downstream datasets of molecular prediction and all the 40 downstream tasks of protein function prediction; thus, robustly transferable pre-trained models are achieved. Furthermore, on these downstream tasks, GNNs pre-trained with such combined strategies achieve significantly better out-of-distribution generalization performance than GNNs pretrained with a single type of (or no) pre-training method. Specifically, on molecular property (resp. protein function) prediction tasks, our combined pre-training methods give 7.2% (resp. 11.7%) higher average ROC-AUC scores compared to the non-pre-trained GNNs, 4.1% (resp. 6.1%) higher average ROC-AUC scores compared to GNNs pre-trained only with graph-level supervised auxillary tasks, and 3.1% (resp. 9.8%) higher average ROC-AUC scores compared to GNNs pre-trained only with node-level self-supervised tasks.

## 2 Preliminaries on Graph Neural Networks

We begin by formalizing the task of supervised learning of graphs, and review the basic components of GNNs [13]. Then, we review existing methods for unsupervised representation learning on graphs.

**Supervised learning of graphs.** Let $G = (V, E)$ denote a graph with node feature vectors $X_v$ for $v \in V$ and edge feature vectors $e_{uv}$ for $(u, v) \in E$. Given a set of graphs $\{ G_1, \ldots, G_N \}$ and their labels $\{y_1, \ldots, y_N\}$, the task of graph supervised learning is to learn a representation vector $h_G$ that helps predict the label of an entire graph, $y_G = g(h_G)$.

**Graph Neural Networks (GNNs)**. GNNs use the graph structure as well as node features and edge features to learn a representation vector of a node, $h_v$, and of the entire graph $h_G$. Modern GNNs follow a neighborhood aggregation strategy, where we iteratively update the representation of a node by aggregating representations of its neighboring nodes and edges [13]. After $k$ iterations of aggregation, a node's representation captures the structural information within its $k$-hop network

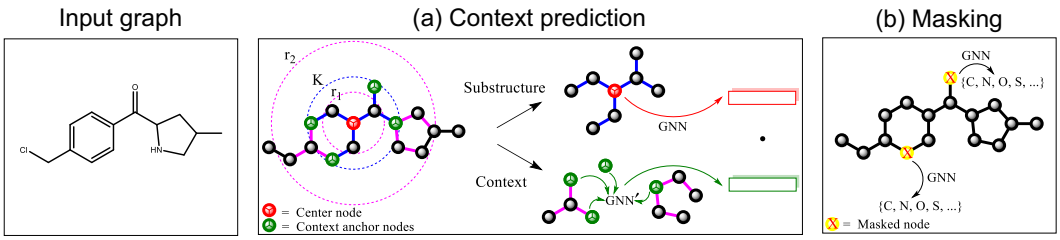



Figure 2: Illustration of Context Prediction and Masking for pre-training GNNs. **(a)** In Context Prediction, the substructure is defined as a $K$-hop subgraph around a selected center node, where $K$ is the number of GNN layers and is set to 2 in the figure. The context is defined as the surrounding subgraph that is between $r_1$- and $r_2$-hop from the center node, where we use $r_1 = 1$ and $r_2 = 4$ in the figure. **(b)** In Masking, the input node/edge attributes are randomly masked, and the GNN is asked to predict them.

neighborhood. Formally, the $k$-th layer of a GNN is

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, \text{AGGREGATE}^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right) : u \in \mathcal{N}(v)\right\}\right)\right), \tag{2.1}$$

where $h_v^{(k)}$ is the feature vector of node $v$ at the $k$-th iteration/layer, $e_{uv}$ is the feature vector of the edge between $u$ and $v$, and $\mathcal{N}(v)$ is a set of nodes adjacent to $v$. We initialize $h_v^{(0)} = X_v$.

To obtain the entire graph's representation $h_G$, the READOUT function pools node features from the final iteration $K$,

$$h_G = \text{READOUT}\left(\left\{h_v^{(K)} \mid v \in G\right\}\right). \tag{2.2}$$

READOUT can be a simple permutation invariant function such as averaging or a more sophisticated graph-level pooling function [59, 62].

**Unsupervised node representation learning.** There is abundant literature in unsupervised representation learning of *nodes* within graphs, which can be broadly classified into two frameworks. The first framework is based on random walk-based objectives [14, 37, 49] or is further simplified to reconstruct adjacency information [15, 25], *i.e.*, predicting existence of edges. The second framework is Deep Graph Infomax [51], which aims to train a node encoder that maximizes mutual information between node representations and the pooled global graph representation. These frameworks are originally proposed and evaluated for node classification and link prediction tasks, and encourage nearby nodes to have similar embeddings. This, however, can be sub-optimal for graph-level prediction tasks, where capturing *structural* similarity of local neighborhood is often more important [61, 42, 57].

## 3 Strategies for pre-training Graph Neural Networks

Our aim of pre-training GNNs is to generate node embeddings that (1) capture structural similarities of neighborhood structure, (2) are composable to give accurate representations of the entire graph, and (3) capture domain knowledge. We achieve these goals by our methods categorized in Figure 1 (a), which we detail in the following.

### 3.1 Context Prediction: Learning node embeddings to capture local graph structure

One successful hypothesis in NLP is the distributional hypothesis [44], which states that words with similar meaning occur in similar contexts. Word2vec is a successful application of this hypothesis [31], which obtains word embeddings by using them to predict context words. Here the context words are simply defined as surrounding words in the same sentence. To pre-train GNNs, we apply this hypothesis to the complex graph domain. Here two challenges arise: (1) how to define "words" and "contexts" in the graph domain, and (2) how to represent the contexts in the prediction problem. In the following, we address these challenges.

**Words and contexts in graph domain.** To define the analogy of words in the graph domain, we note that the final node embedding $h_v^{(K)}$ obtained by GNNs encode the $K$-hop subgraph structure

centered at node $v$. Therefore, if we use the node embeddings to predict context, the analogy of words in the graph domain would be the *$K$-hop subgraph* centered at each node, which is depicted as "substructure" in Figure 2 (a). Analogously, the context can be defined as the graph structure *surrounding* the $K$-hop subgraph, *i.e.*, the subgraph that is between $r_1$- and $r_2$-hop from the center node, depicted as "context" in Figure 2 (a). We require $r_1 < K$ so that some nodes are shared between substructure and context, and we call these shared nodes *context anchor nodes*.

**Encoding context into a fixed vector using another GNN.** Unlike NLP, context in the graph domain is a structured object, which is non-trivial as a prediction target. Here to enable such prediction, we encode the structured object as a *fixed-length vector* using another GNN with learnable parameters. This allows us to apply the framework of negative sampling in word2vec [31] to predict context represented as a vector. To obtain the vector for context, as depicted in Fig. 2 (a), we first apply the context GNN (denoted as GNN$'$ in Fig. 2 (a)) to obtain node embeddings in the context graph. We then average the embeddings across the *context anchor nodes* to obtain the fixed-length context embedding. For node $v$ in graph $G$, we denote the corresponding context embedding as $c_v^G$.

**Learning via negative sampling.** We apply negative sampling [31, 58] to learn the two GNNs jointly: one for encoding the substructure to obtain node embedding $h_v^{(K)}$, and another for encoding the context to obtain context embedding $c_{v'}^{G'}$ for node $v'$ from graph $G'$. Specifically, our learning objective is binary classification of whether a given substructure context pair is true (positive) or false (negative):

$$\sigma\left(h_v^{(K)\top} c_{v'}^{G'}\right) \approx \mathbf{1}\{v \text{ and } v' \text{ are the same node in } G\}, \tag{3.1}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\mathbf{1}(\cdot)$ is the indicator function. We either let $v' = v$ and $G' = G$ (for the positive substructure context pair), or randomly sample $v'$ from a randomly chosen graph $G'$ (for the negative substructure context pair). We use the negative sampling ratio of 1 (one negative pair per positive pair), and use the negative log likelihood as the loss function. After training, we expect the first GNN to map substructures with similar surrounding contexts to similar vectors in the embedding space.

It should be noted that similar intuition has been explored in the literature [32, 19, 63]. However, crucially, all of these methods use distinct embeddings for different substructures/contexts without any parameter sharing; thus, they are inherently not inductive, cannot be fine-tuned in an end-to-end manner, and cannot capture large and diverse substructures/contexts due to data sparsity. All of these issues can be elegantly handled by our framework using GNNs. It should be also noted that our way of applying distributional hypothesis to the graph domain is fundamentally different from existing unsupervised node representation learning methods based on random walks [14, 37], which define context as surrounding *nodes*, not surrounding *structure.* Consequently, the existing methods capture positional information of nodes within a graph, rather than their neighboring *structure* information.

### 3.2 Masking: Learning node embeddings to capture domain knowledge

A different approach to pre-train GNNs would be to design a node-level self-supervised task so that by performing the task, the pre-trained GNNs can uncover domain-specific principles hidden in data.

In NLP, BERT pre-trained models have achieved impressive performance on a wide range of downstream tasks [8]. At the core of BERT is the masked language model, which is a self-supervised prediction task to encourage the model to capture useful domain knowledge about language.

Here, for graph structured data, we propose to mask domain-specific attributes associated with nodes or edges, and let GNNs predict those attributes based on neighboring structure. Figure 2 (b) illustrates our proposed method for the node masking case applied to a molecular graph. We mask input node/edge attributes, for example atom type, by replacing them with special masked indicators. We then apply GNNs to obtain the corresponding node/edge embeddings (we get edge embeddings by simply summing the embeddings of its two end-nodes). Finally, we use a linear model on top of the embeddings to predict the masked node/edge attributes.

We use Masking to learn meaningful neighboring structure that reflects domain-specific knowledge. For example, we can apply Masking on molecular graphs, as shown in Figure 2 (b), so that GNNs can learn simple chemistry rules such as valency, as well as potentially more complex chemistry phenomenon such as the electronic or steric properties of functional groups. Another interesting application of our Masking is PPI networks with multiple edge types, each of which indicates a

different kind of interaction between a pair of proteins. Here, predicting masked edge types based on neighboring structure enables GNNs to learn how different edge types relate and interact with each other.

### 3.3 Graph-level Prediction: Learning composable node embeddings

Having obtained node embeddings that capture the graph structure and/or domain knowledge, we then pre-train GNNs to generate node embeddings that can be effectively *composed* to obtain meaningful representations of an entire graph. As illustrated in Figure 1 (b.iii), this ensures both node and graph embeddings are of high-quality; thus, the graph embeddings are more robust and transferable to diverse downstream tasks.

As shown in Figure 1 (a), there are two options possible for graph-level pre-training: learn composable embeddings either (1) by making predictions on *domain-specific* auxiliary supervised tasks, or, (2) by performing self-supervised prediction on graph-level structure, *e.g.*, graph edit distance [3] or graph structure similarity [33]. As the graph-level representations will be directly used for fine-tuning on downstream graph-level prediction tasks, it is desirable to directly encode *domain-specific* information into the graph embeddings. Therefore, we focus on the first approach and introduce domain-specific tasks for *graph-level supervised pre-training* of GNNs, which we detail in Section 4.

## 4 Experiments

In this section, we evaluate GNN pre-training methods in Section 3 on two important scientific applications of graph classification, namely, molecular property prediction in chemistry, and protein function prediction in biology. We find that (1) naïvely performing only node-level self-supervised pre-training or only graph-level supervised pre-training is *not* enough to achieve robust performance gain over non-pre-trained models across diverse downstream tasks, but that (2) the combination of both node-level and graph-level pre-training methods can significantly and consistently improve out-of-distribution generalization performance on diverse downstream tasks.

**Chemical/Biological graphs.** In chemistry, molecules can be represented by chemical graphs composed of atoms (nodes) and chemical bonds (edges). We use 8 binary classification datasets from MoleculeNet [52] as our downstream tasks (some datasets contain multiple prediction tasks), whose statistics are summarized in Table 1 and details are provided in Appendix B. Most of our chemical graphs contain around 10 to 30 nodes corresponding to heavy (non-hydrogen) atoms. In biology, PPI networks can be represented by biological graphs composed of proteins (unlabeled nodes) and PPI relationships (edges with 7 categorical attributes indicating different types of interactions between proteins). PPI network subgraphs centered at a protein of interest (*i.e.*, ego-networks) are used to predict their biological functions. We define *fine-grained protein functions* as Gene Ontology (GO) annotations that are leaves in the GO hierarchy, and define *coarse-grained protein functions* as GO annotations that are the immediate parents of leaves [2, 7]. For example, a fine-grained protein function is "Factor XII activation", while a coarse-grained function is "positive regulation of protein". The former is a specific type of the latter, and is much harder to derive experimentally. As downstream tasks, we perform binary classification on 40 *fine-grained* protein functions; therefore, we have 40 prediction tasks. The number of graphs is 88K derived from PPI networks of 8 labelled species, and the average graph size is 49 nodes with 365 edges. Additional details about the datasets, and the features of the molecule and protein graphs can be found in Appendix B and C.

**Pre-training datasets.** For pre-training with molecular graphs, we use 2 million unlabeled molecules sampled from the ZINC15 database [47] for the self-supervised pre-training, and use a preprocessed ChEMBL dataset [29, 12], containing 456K molecules with 1310 kinds of binary biochemical assays, for the graph-level supervised pre-training. For pre-training with protein graphs, we use 395K unlabeled protein ego-networks derived from PPI networks of 50 species for the self-supervised pre-training, and use the 88K protein ego-networks to jointly predict their 5000 *coarse-grained* protein function labels for the graph-level supervised pre-training.

**Dataset splitting.** In many scientific applications of machine learning, conventional random split is overly optimistic and does not simulate the real-world use case, where test graphs can be structurally different from training graphs [52, 65]. To relect the actual use case, we split the downstream data in the following ways to evaluate the models' *out-of-distribution generalization* performance.

For molecular prediction tasks, following [38], we cluster molecules by graph substructure [4], and recombine the clusters by placing the most common scaffolds (molecular substructures) in the training

set, producing validation and test sets that contain structurally different molecules. Prior work has shown that this *scaffold split* provides a more realistic estimate of model performance in prospective evaluation compared to random split [6, 45]. The split for train/validation/test sets is 80%:10%:10%.

In the PPI network, *species split* simulates a scenario where we have only high-level *coarse-grained* knowledge on a subset of proteins (prior set) in a species of interest (human in our experiments), and want to predict *fine-grained* biological functions for the rest of the proteins in that species (test set). For species split, we use 50% of the protein subgraphs from human as test set, and 50% as a prior set containing only coarse-grained protein annotations. The protein subgraphs from 7 other labelled species (arabidopsis, celegans, ecoli, fly, mouse, yeast, zebrafish) are used as train and validation sets, which are split 85% : 15%. The effective split ratio for the train/validation/prior/test sets is 69% : 12% : 9.5% : 9.5%.

All the chemical/biological test graphs that are used for downstream test evaluation are removed from the graph-level supervised pre-training datasets; thus, during the supervised pre-training in our PPI experiments, we only use train/validation/prior graphs to predict their coarse-grained protein labels.

**GNN architectures.** Throughout the experiments, we use the Graph Isomorphism Network (GIN) [53]. We select the following hyper-parameters that performed well across all downstream tasks in the validation sets: 300 dimensional hidden units, 5 GNN layers ($K = 5$), Batch Normalization [18] and dropout [46] at each layer, and average pooling for the READOUT function. All models are trained with Adam optimizer [24] with a learning rate of 0.001. We use Pytorch [36] and Pytorch Geometric [10] for all of our implementation. Additional details can be found in Appendix A.

**Pre-training procedure.** For Context Prediction illustrated in Figure 2 (a), on molecular graphs, we define context graphs by setting inner radius $r_1 = 4$, and on PPI networks, we use $r_1 = 1$. For both molecular and PPI graphs, we let outer radius $r_2 = r_1 + 3$, and use a 3-layer GNN to encode the context structure. For Masking shown in Figure 2 (b), we randomly mask 15% of node (for molecular graphs) or edge attributes (for PPI networks) for prediction. We run all pre-training methods for 100 epochs. As baselines for node-level self-supervised pre-training, we adopt the original Edge Prediction (denoted by EdgePred) [15] and Deep Graph Infomax (denoted by Infomax) [51] implementations. For self-supervised pre-training, we use a batch size of 256, while for supervised pre-training, we use batch size of 32 with dropout rate of 20%. We combine node-level self-supervised pre-training and graph-level supervised pre-training by first performing the node-level pre-training, and subsequently performing the graph-level pre-training.

**Fine-tuning procedure.** After pre-training, models are fine-tuned using the training sets of the downstream task datasets. We apply randomly-initialized linear classifiers on top of the graph-level representations to predict labels in the downstream tasks. We fine-tune the entire models with a batch size of 32 and dropout rate of 50%. Datasets with multiple prediction tasks are fit jointly. On the molecular property prediction datasets, we train models for 100 epochs, while on the protein function prediction dataset (with the 40 binary prediction tasks), we train models for 50 epochs.

**Evaluation.** We evaluate test performance on downstream tasks using ROC-AUC [5] with the validation early stopping protocol, *i.e.*, test ROC-AUC at the best validation epoch is reported. For datasets with multiple prediction tasks, we take the average ROC-AUC across all their tasks. The downstream experiments are run with 10 random seeds, and we report mean ROC-AUC and standard deviation.

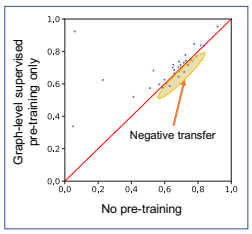
**Results and discussion.** We report the test results for molecular property prediction and protein function prediction in Table 1 and Figure 3, respectively. We observe the following trends across datasets/tasks and domains:

1. When only graph-level supervised pre-training is performed, we observe negative transfer in many downstream datasets/tasks (2 out of 8 datasets in molecular prediction, and 12 out of 40 tasks in protein function prediction. See the shaded cells of Table 1, and the highlighted region in the middle panel of Figure 3). Such negative transfer is completely avoided when *node-level* self-supervised pre-training is combined with graph-level supervised pre-training. This indicates that appropriate *node-level* pre-training can regularize GNNs to learn robust and transferable *graph-level* representations during graph-level supervised pre-training, as illustrated in Figure 1 (b.iii).
2. Combining the node- and graph-level pre-training methods gives significantly better generalization performance than only performing a single type of (or no) pre-training method.

| Dataset | | BBBP | Tox21 | ToxCast | SIDER | ClinTox | MUV | HIV | BACE | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| # Molecules | | 2039 | 7831 | 8575 | 1427 | 1478 | 93087 | 41127 | 1513 | / |
| # Prediction tasks | | 1 | 12 | 617 | 27 | 2 | 17 | 1 | 1 | / |
| Pre-training strategy | | Out-of-distribution prediction (scaffold split) | | | | | | | | |
| Graph-level | Node-level | | | | | | | | | |
| None | None | 65.8 ±4.5 | 74.0 ±0.8 | 63.4 ±0.6 | 57.3 ±1.6 | 58.0 ±4.4 | 71.8 ±2.5 | 75.3 ±1.9 | 70.1 ±5.4 | 67.0 |
| None | Infomax | **68.8 ±0.8** | 75.3 ±0.5 | 62.7 ±0.4 | 58.4 ±0.8 | 69.9 ±3.0 | 75.3 ±2.5 | 76.0 ±0.7 | 75.9 ±1.6 | 70.3 |
| None | EdgePred | 67.3 ±2.4 | 76.0 ±0.6 | 64.1 ±0.6 | 60.4 ±0.7 | 64.1 ±3.7 | 74.1 ±2.1 | 76.3 ±1.0 | 79.9 ±0.9 | 70.3 |
| None | Masking | 64.3 ±2.8 | 76.7 ±0.4 | 64.2 ±0.5 | 61.0 ±0.7 | 71.8 ±4.1 | 74.7 ±1.4 | 77.2 ±1.1 | 79.3 ±1.6 | 71.1 |
| None | ContextPred | 68.0 ±2.0 | 75.7 ±0.7 | 63.9 ±0.6 | 60.9 ±0.6 | 65.9 ±3.8 | 75.8 ±1.7 | 77.3 ±1.0 | 79.6 ±1.2 | 70.9 |
| Supervised | None | 68.3 ±0.7 | 77.0 ±0.3 | 64.4 ±0.4 | 62.1 ±0.5 | 57.2 ±2.5 | 79.4 ±1.3 | 74.4 ±1.2 | 76.9 ±1.0 | 70.0 |
| Supervised | Infomax | 68.0 ±1.8 | **77.8 ±0.3** | 64.9 ±0.7 | 60.9 ±0.6 | 71.2 ±2.8 | **81.3 ±1.4** | 77.8 ±0.9 | 80.1 ±0.9 | 72.8 |
| Supervised | EdgePred | 66.6 ±2.2 | **78.3 ±0.3** | **66.5 ±0.3** | **63.3 ±0.9** | 70.9 ±4.6 | 78.5 ±2.4 | 77.5 ±0.8 | 79.1 ±3.7 | 72.6 |
| Supervised | Masking | 66.5 ±2.5 | **77.9 ±0.4** | 65.1 ±0.3 | **63.9 ±0.9** | **73.7 ±2.8** | **81.2 ±1.9** | 77.1 ±1.2 | 80.3 ±0.9 | 73.2 |
| Supervised | ContextPred | **68.7 ±1.3** | **78.1 ±0.6** | 65.7 ±0.6 | 62.7 ±0.8 | **72.6 ±1.5** | **81.3 ±2.1** | **79.9 ±0.7** | **84.5 ±0.7** | **74.2** |

Table 1: **Test ROC-AUC (%) performance on molecular prediction benchmarks with different pre-training strategies.** The rightmost column averages the mean of test performance across the 8 datasets. The best result for each dataset and the comparable results that are within one standard deviation from the best ones are bolded. The shaded cells indicate negative transfer, *i.e.*, ROC-AUC of pre-trained models is worse than that of the non-pre-trained models.

| Pre-training strategy | | Out-of-dist. |
|---|---|---|
| Graph-level | Node-level | (species split) |
| None | None | 64.2 ±1.4 |
| None | Infomax | 63.6 ±1.2 |
| None | EdgePred | 66.1 ±1.5 |
| None | ContextPred | 64.8 ±1.3 |
| None | Masking | 64.3 ±1.4 |
| Supervised | None | 69.8 ±2.2 |
| Supervised | Infomax | 71.7 ±1.8 |
| Supervised | EdgePred | 72.9 ±1.2 |
| Supervised | ContextPred | 74.3 ±0.9 |
| Supervised | Masking | **75.9 ±0.8** |



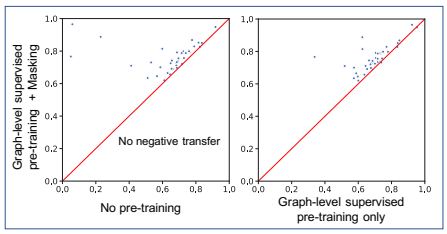




Figure 3: **Test ROC-AUC of protein function prediction. (Left)** Test ROC-AUC scores (%) obtained by different pre-training strategies, where the scores are averaged over the 40 fine-grained prediction tasks. **(Middle and right):** Each figure represents a scatter plot comparison of ROC-AUC scores for a pair of pre-training strategies on the 40 *individual* downstream tasks. Each point in a figure represents a particular individual downstream task. **(Middle):** We see that there are many *individual* downstream tasks where the graph-level supervised pre-trained model performs worse than the non-pre-trained model, which indicates negative transfer. **(Right):** When the graph-level supervised pre-training and our Masking are combined, the negative transfer is completely avoided across all the 40 individual downstream tasks.

3. Our proposed node-level self-supervised pre-training methods (Masking and Context Prediction) give particularly promising results compared with existing methods (Edge Prediction and Infomax), especially when combined with graph-level supervised pre-training.
4. In the molecular property prediction, as shown in Table 1, our Context Prediction + graph-level supervised pre-training strategy gives the most promising performance, leading to an increase in average ROC-AUC of 7.2% over the non-pre-trained baseline and 4.2% over the graph-level supervised pre-trained baseline.
5. In the protein function prediction, as seen in Figure 3, our Masking + graph-level supervised pre-training strategy gives superior generalization performance compared to the other baselines *across almost all* the 40 individual prediction tasks, improving average ROC-AUC by 11.7% over the non-pre-trained baseline and 6.1% over the graph-level supervised pre-trained baseline.
6. Finally, at the fine-tuning stage, our pre-trained models achieve significantly faster training convergence than the non-pre-trained model across all downstream datasets (see Figures 4 and 5 in Appendix D). One possible explanation is that the pre-trained GNNs learn discriminative graph features in the process of performing pre-training tasks, which makes training on downstream tasks easier than starting from scratch.

## 5 Conclusion

In this paper, on a variety of downstream tasks in the scientific domains, we demonstrate that pre-training can significantly and consistently increase the out-of-distribution generalization capabilities

of GNNs. Crucially, to achieve this, the node-level and graph-level pre-training methods need to be effectively combined, which ensures that the node embeddings capture neighborhood semantics and can be pooled to obtain a representation of an entire graph useful for a variety of downstream tasks. Otherwise, we observe negative transfer in many downstream tasks, which significantly limits the applicability and reliability of pre-trained GNNs. Overall, we believe pre-training is a promising approach to close the gap of applying GNNs to real-world scientific problems.

## Acknowledgments

We thank Camilo Ruiz, Rex Ying, Zhenqin Wu, Shantao Li, Srijan Kumar, Hongwei Wang, and Robin Jia for their helpful discussion. W.H is supported by Funai Overseas Scholarship. J.L is a Chan Zuckerberg Biohub investigator. This research has been supported in part by NSF OAC-1835598, DARPA MCS, DARPA ASED, ARO MURI, Boeing, Docomo, Hitachi, Huawei, JD, Siemens, and Stanford Data Science Initiative. The Pande Group acknowledges the generous support of Dr. Anders G. Frøseth and Mr. Christian Sundt for our work on machine learning. The Pande Group is broadly supported by grants from the NIH (R01 GM062868 and U19 AI109662) as well as gift funds and contributions from Folding@home donors.

V.S.P is a consultant & SAB member of Schrodinger, LLC and Globavir, sits on the Board of Directors of Apeel Inc, Asimov Inc, BioAge Labs, Ciitizen, Devoted Health, Freenome Inc, Omada Health, Patient Ping, Rigetti Computing, and is a General Partner at Andreessen Horowitz.

## References

[1] Aact database, Jan 2017.

[2] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

[3] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive whole-graph embedding by preserving graph proximity. *arXiv preprint arXiv:1904.01098*, 2019.

[4] Guy W. Bemis and Mark A. Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996. PMID: 8709122.

[5] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[6] Bin Chen, Robert P. Sheridan, Viktor Hornak, and Johannes H. Voigt. Comparison of random forest and pipeline pilot naïve bayes in prospective qsar predictions. *Journal of Chemical Information and Modeling*, 52(3):792–803, 2012. PMID: 22360769.

[7] Gene Ontology Consortium. The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1):D330–D338, 2018.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] 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. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2224–2232, 2015.

[10] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[11] Eleanor J Gardiner, John D Holliday, Caroline O'Dowd, and Peter Willett. Effectiveness of 2d fingerprints for scaffold hopping. *Future medicinal chemistry*, 3(4):405–414, 2011.

[12] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2011.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pages 1273–1272, 2017.

[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 855–864. ACM, 2016.

[15] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1025–1035, 2017.

[16] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

[17] Aids antiviral screen data - nci dtp data - national cancer institute, 2004.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[19] Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1):27–35, 2018.

[20] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[21] Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2607–2616, 2017.

[22] Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling industrial admet data with multitask networks. *arXiv preprint arXiv:1606.08793*, 2016.

[23] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[25] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[26] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[27] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2015.

[28] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[29] 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.

[30] John E McMurry. *Fundamentals of organic chemistry*. Cengage Learning, 2010.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[32] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*, 2016.

[33] Nicolò Navarin, Dinh V Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels. *arXiv preprint arXiv:1811.06930*, 2018.

[34] Paul A. Novick, Oscar F. Ortiz, Jared Poelman, Amir Y. Abdulhay, and Vijay S. Pande. Sweetlead: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery. *PLOS ONE*, 8(11), 11 2013.

[35] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710. ACM, 2014.

[38] Bharath Ramsundar, Peter Eastman, Patrick Walters, and Vijay Pande. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. `https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837`.

[39] 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.

[40] 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.

[41] Ann M. Richard, Richard S. Judson, Keith A. Houck, Christopher M. Grulke, Patra Volarath, Inthirany Thillainadarajah, Chihae Yang, James Rathman, Matthew T. Martin, John F. Wambaugh, Thomas B. Knudsen, Jayaram Kancherla, Kamel Mansouri, Grace Patlewicz, Antony J. Williams, Stephen B. Little, Kevin M. Crofton, and Russell S. Thomas. Toxcast chemical landscape: Paving the road to 21st century toxicology. *Chemical Research in Toxicology*, 29(8):1225–1251, 2016.

[42] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[43] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[44] Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[45] Robert P. Sheridan. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of Chemical Information and Modeling*, 53(4):783–790, 2013. PMID: 23521722.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[47] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. PMID: 26479676.

[48] Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of $\beta$-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of chemical information and modeling*, 56(10):1936–1949, 2016.

[49] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 1067–1077, 2015.

[50] Tox21 data challenge 2014, 2014.

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

[52] 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.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[54] Yuting Xu, Junshui Ma, Andy Liaw, Robert P Sheridan, and Vladimir Svetnik. Demystifying multitask deep neural networks for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 57(10):2490–2504, 2017.

[55] Yan Yan, Shangzhao Qiu, Zhuxuan Jin, Sihong Gong, Yun Bai, Jianwei Lu, and Tianwei Yu. Detecting subnetwork-level dynamic correlations. *Bioinformatics*, 33(2):256–265, 2017.

[56] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Are learned molecular representations ready for prime time? *arXiv preprint arXiv:1904.01561*, 2019.

[57] Rendong Yang, Yun Bai, Zhaohui Qin, and Tianwei Yu. EgoNet: identification of human disease ego-network modules. *BMC Genomics*, 15(1):314, 2014.

[58] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 974–983, 2018.

[59] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[60] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421, 2018.

[61] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[62] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, pages 4438–4445, 2018.

[63] Quan Zhou, Peizhe Tang, Shenxiu Liu, Jinbo Pan, Qimin Yan, and Shou-Cheng Zhang. Learning atoms for materials discovery. *Proceedings of the National Academy of Sciences*, 115(28):E6411–E6417, 2018.

[64] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[65] Marinka Zitnik, Rok Sosič, Marcus W. Feldman, and Jure Leskovec. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*, 116(10):4426–4433, 2019.

# A Details of GNN architectures

Here we describe GNN architectures used in our in molecular property and protein function prediction experiments. Both follow the architecture of GIN [53] with some minor modifications to include edge features as well as center node information in the protein ego-networks.

**Molecular property prediction.** In molecular property prediction, the raw node features and edge features are both 2-dimensional categorical vectors (see Appendix B for the detail), denoted as $(i_{v,1}, i_{v,2})$ and $(j_{e,1}, j_{e,2})$ for node $v$ and edge $e$, respectively. Note that we also introduce unique categories to indicate masked node/edges as well as self-loop edges. As input features to GNNs, we first embed the categorical vectors by

$$h_v^{(0)} = \mathrm{EmbNode}_1(i_{v,1}) + \mathrm{EmbNode}_2(i_{v,2})$$
$$h_e^{(k)} = \mathrm{EmbEdge}_1^{(k)}(j_{e,1}) + \mathrm{EmbEdge}_2^{(k)}(j_{e,2}) \quad \text{for } k = 0, 1, \ldots, K-1,$$

where $\mathrm{EmbNode}_1(\cdot)$, $\mathrm{EmbNode}_2(\cdot)$, $\mathrm{EmbEdge}_1^{(k)}(\cdot)$, and $\mathrm{EmbNode}_1^{(k)}(\cdot)$ represent embedding operations that map integer indices to $d$-dimensional real vectors, and $k$ represents the index of GNN layers. At the $k$-th layer, GNNs update node representations as

$$h_v^{(k)} = \mathrm{ReLU}\left(\mathrm{MLP}^{(k)}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(k-1)} + \sum_{e=(v,u): u \in \mathcal{N}(v) \cup \{v\}} h_e^{(k-1)}\right)\right), \tag{A.1}$$

where $\mathcal{N}(v)$ is a set of nodes adjacent to $v$, and $e = (v, v)$ represents the self-loop edge. Note that for the final layer, i.e., $k = K$, we removed the ReLU from Eq. (A.1) so that $h_v^{(k)}$ can take negative values. This is crucial for pre-training methods based on the dot product, *e.g.*, Context Prediction and Edge Prediction, as otherwise, the dot product between two vectors would be always positive.

The graph-level representation $h_G$ is obtained by averaging the node embeddings at the final layer, i.e.,

$$h_G = \mathrm{MEAN}(\{h_v^{(K)} \mid v \in G\}). \tag{A.2}$$

The label prediction is made by a linear model on top of $h_G$.

In our experiments, we set all the embedding dimension $d$ to 300. For MLPs in Eq. (A.1), we use the ReLU activation with 600 hidden units. We apply batch normalization [18] right before the ReLU in Eq. (A.1) and apply dropout [46] to $h_v^{(k)}$ at all the layers except the input layer.

**Protein function prediction.** The GNN architecture used for protein function prediction is similar to the one used for molecular property prediction except for a few differences. First, the raw input node features are uniform (denoted as $X$ here) and second, the raw input edge features are binary vectors (see Appendix C for the detail), which we denote as $c_e \in \{0, 1\}^{d_0}$. As input features to GNNs, we first embed the raw features by

$$h_v^{(0)} = X$$
$$h_e^{(k)} = W c_e + b \quad \text{for } k = 0, 1, \ldots, K-1,$$

where $W \in \mathbb{R}^{d \times d_0}$ and $b \in \mathbb{R}^d$ are learnable parameters, and $h_v^{(0)}, h_e^{(k)} \in \mathbb{R}^d$. At each layer, GNNs update node representations as

$$h_v^{(k)} = \mathrm{ReLU}\left(\mathrm{MLP}^{(k)}\left(\mathrm{CONCAT}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(k-1)}, \sum_{e=(v,u): u \in \mathcal{N}(v) \cup \{v\}} h_e^{(k-1)}\right)\right)\right), \tag{A.3}$$

where $\mathrm{CONCAT}(\cdot, \cdot)$ takes two vectors as input and concatenates them. Since the downstream task is ego-network classification, we use the embedding of the center node $v_{\mathrm{center}}$ together with the embedding of the entire ego-network. More specifically, we obtain graph-level representation $h_G$ by

$$h_G = \mathrm{CONCAT}\left(\mathrm{MEAN}(\{h_v^{(K)} \mid v \in G\}), h_{v_{\mathrm{center}}}^{(K)}\right). \tag{A.4}$$

## B Details of molecular datasets

**Input graph representation.** For simplicity, we use a minimal set of node and bond features that unambiguously describe the two-dimensional structure of molecules.

- Node features:
  - Atom number: [1, 118]
  - Chirality tag: {unspecified, tetrahedral cw, tetrahedral ccw, other}
- Edge features:
  - Bond type: {single, double, triple, aromatic}
  - Bond direction: {none, endupright, enddownright}

**Downstream task datasets.** 8 binary graph classification datasets from Moleculenet [52] are used to evaluate model performance.

- **BBBP.** Blood-brain barrier penetration (membrane permeability) [28].
- **Tox21.** Toxicity data on 12 biological targets, including nuclear receptors and stress response pathways [50].
- **ToxCast.** Toxicology measurements based on over 600 in vitro high-throughput screenings [41].
- **SIDER.** Database of marketed drugs and adverse drug reactions (ADR), grouped into 27 system organ classes [27].
- **ClinTox.** Qualitative data classifying drugs approved by the FDA and those that have failed clinical trials for toxicity reasons [34, 1].
- **MUV.** Subset of PubChem BioAssay by applying a refined nearest neighbor analysis, designed for validation of virtual screening techniques [11].
- **HIV.** Experimentally measured abilities to inhibit HIV replication [17].
- **BACE.** Qualitative binding results for a set of inhibitors of human $\beta$-secretase 1 [48].

## C Details of protein datasets

**Input graph representation.** The protein subgraphs only have edge features.

- Edge features:
  - Neighbourhood: {True, False}
  - Fusion: {True, False}
  - Co-occurrence: {True, False}
  - Co-expression: {True, False}
  - Experiment: {True, False}
  - Database: {True, False}
  - Text: {True, False}

These edge features indicate whether a particular type of relationship exists between a pair of proteins:

- Neighbourhood: if a pair of genes are consistently observed in each other's genome neighbourhood
- Fusion: if a pair of proteins have their respective orthologs fused into a single protein-coding gene in another organism
- Co-occurrence: if a pair of proteins tend to be observed either as present or absent in the same subset of organisms
- Co-expression: if a pair of proteins share similar expression patterns
- Experiment: if a pair of proteins are experimentally observed to physically interact with each other

- Database: if a pair of proteins belong to the same pathway, based on assessments by a human curator
- Text mining: if a pair of proteins are mentioned together in PubMed abstracts

**Datasets.** A dataset containing protein subgraphs from 50 species is used [65]. The original PPI networks contain edge attributes that correspond to the degree of confidence for 7 different types of protein-protein relationships. The edge weights range from 0, which indicates no evidence for the specific relationship, to 1000, which indicates the highest confidence. The weighted edges of the PPI networks are thresholded such that the distribution of edge types across the 50 PPI networks are uniform. Then, for every node in the PPI networks, subgraphs centered on each node were generated by: (1) performing a breadth first search to select the subgraph nodes, with a search depth limit of 2 and a maximum number of 10 neighbors randomly expanded per node, (2) including the selected subgraph nodes and all the edges between those nodes to form the resulting subgraph.

The entire dataset contains 394,925 protein subgraphs derived from 50 species. Out of these 50 species, 8 species (arabidopsis, celegans, ecoli, fly, human, mouse, yeast, zebrafish) have proteins with GO protein annotations. The dataset contains 88,000 protein subgraphs from these 8 species, of which 57,448 proteins have at least one positive coarse-grained GO protein annotation and 22,876 proteins have at least one positive fine-grained GO protein annotation. For the self-supervised pre-training dataset, we use all 394,925 protein subgraphs. The supervised pre-training dataset and the downstream evaluation dataset are derived from the 8 labeled species, as described in Section 4. The 40-th most common *fine-grained* protein label only has 121 positively annotated proteins, while the 40-th most common *coarse-grained* protein label has 9386 positively annotated proteins. This illustrates the extreme label scarcity of our downstream tasks.

For supervised pre-training, we combine the train, validation, and prior sets described previously, with the 5,000 most common coarse-grained protein function annotations as binary labels. For our downstream task, we predict the 40 most common fine-grained protein function annotations, to ensure that each protein function has at least 10 positive labels in our test set.

## D Additional experimental results

**Training and validation curves.** In Figures 4 and 5, we plot training and validation curves for all the datasets we used in the experiments.

**Additional scatter plot comparisons of ROC-AUCs.** In Figure 6, we compare our supervised + Context Prediction pre-training with a non-pretrained model and a supervised pre-trained model. We see from the left figure that the combined strategy again completely avoids negative transfer across all the 40 downstream tasks. Furthermore, we see from the right figure that adding our node-level Context Prediction pre-training almost always improves ROC-AUC scores of supervised pre-trained models *across the 40 downstream tasks*.

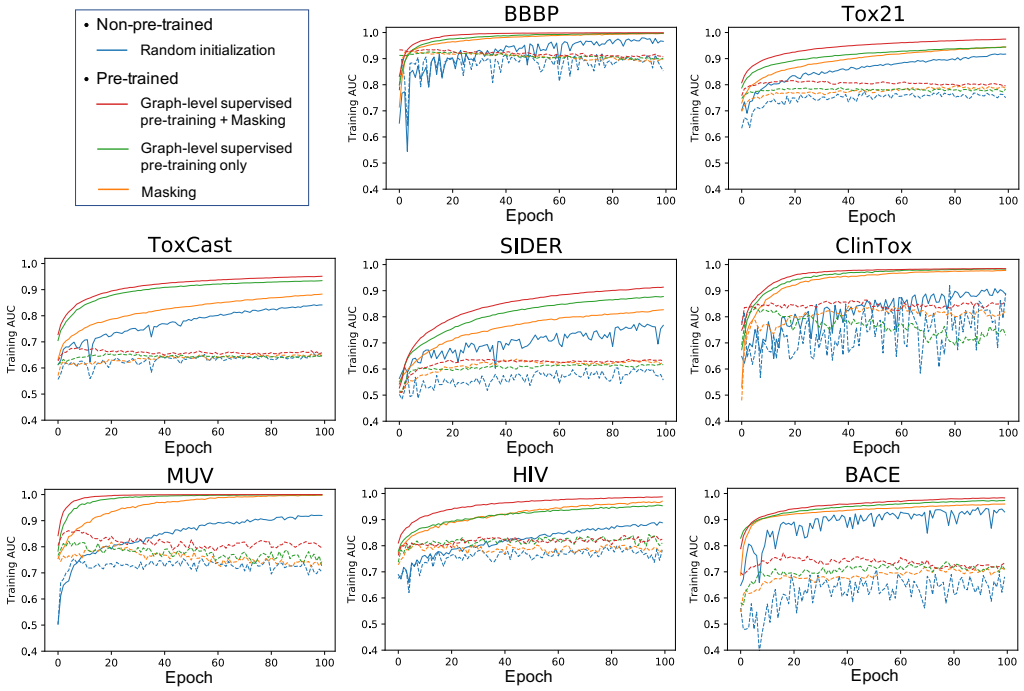



Figure 4: Training curves of different pre-traning strategies. The dashed lines indicate the validation curves.

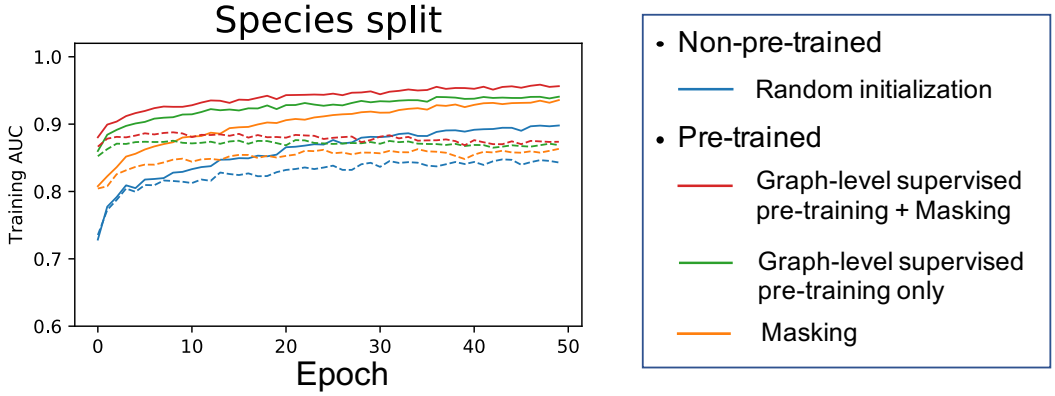



Figure 5: Training curves of different pre-traning strategies on the protein function prediction task. The dashed lines indicate the validation curves.

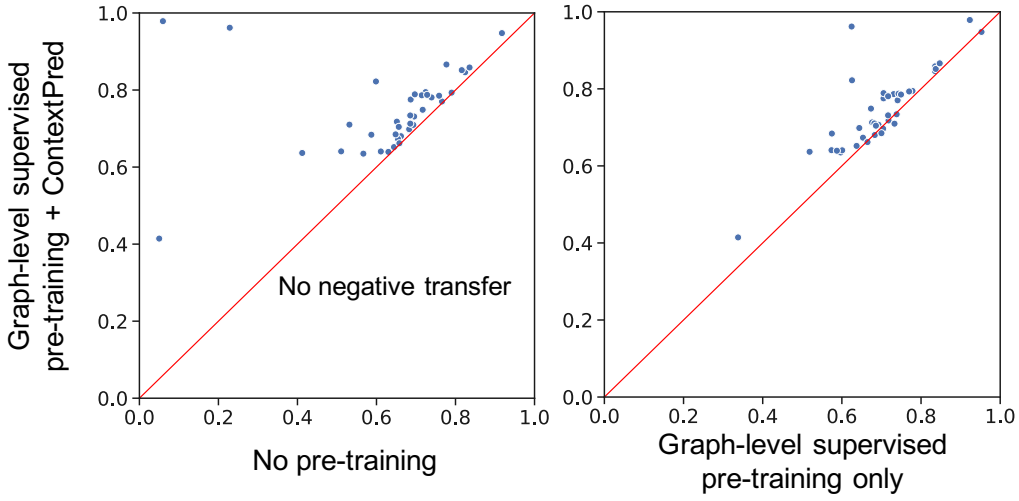



Figure 6: Scatter plot comparisons of ROC-AUC scores of our graph-level supervised + Context Prediction pre-training strategy versus the two baseline strategies (non-pre-trained and graph-level supervised pre-trained) on the 40 individual downstream tasks of predicting different fine-grained protein function labels.