Graph Neural Networks Pretraining Through Inherent Supervision for Molecular Property Prediction

Roy Benjamin
Technion - Israel Institute of Technology
Haifa, Israel
roe.b@cs.technion.ac.il

Uriel Singer
Meta AI Research
Tel-Aviv, Israel
urielsinger@fb.com

Kira Radinsky
Technion - Israel Institute of Technology
Haifa, Israel
kirar@cs.technion.ac.il

ABSTRACT

Recent global events have emphasized the importance of accelerating the drug discovery process. A way to deal with the issue is to use machine learning to increase the rate at which drugs are made available to the public. However, chemical labeled data for real-world applications is extremely scarce making traditional approaches less effective. A fruitful course of action for this challenge is to pretrain a model using related tasks with large enough datasets, with the next step being finetuning it for the desired task. This is challenging as creating these datasets requires labeled data or expert knowledge. To aid in solving this pressing issue, we introduce MISU - Molecular Inherent SUpervision, a unique method for pretraining graph neural networks for molecular property prediction. Our method leapfrogs past the need for labeled data or any expert knowledge by introducing three innovative components that utilize inherent properties of molecular graphs to induce information extraction at different scales, from the local neighborhood of an atom to substructures in the entire molecule. Our empirical results for six chemical-property-prediction tasks show that our method reaches state-of-the-art results compared to numerous baselines.

CCS CONCEPTS

• **Applied computing** → **Life and medical sciences**; *Chemistry*; • **Computing methodologies** → **Neural networks**; **Transfer learning**.

KEYWORDS

Drug Discovery, ML for Healthcare, Molecular Property Prediction

ACM Reference Format:
Roy Benjamin, Uriel Singer, and Kira Radinsky. 2022. Graph Neural Networks Pretraining Through Inherent Supervision for Molecular Property Prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557085

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557085

1 INTRODUCTION

The importance of rapid development of pharmaceutical solutions for humanity's ailments has become clear in recent years. However, the process requires an enormous investment of resources, and a single drug development process might take more than a decade and costs billions. A major challenge of drug design is discovering target molecules with desired properties. Today, potential leads are either identified through a manual process or High Throughput Screening (HTS), which involves screening large amounts of compounds to test varieties of potential candidates. However, since the number of realistic drug-like molecules is estimated to be between $10^{23} - 10^{60}$ [36], larger by many orders of magnitude than what we can test with current lab technology, it is apparent that some section of search space will never be explored. Hence, a more efficient method is needed to improve the way we perform drug discovery.

To accelerate the drug discovery process numerous machine learning algorithms were applied for various chemical tasks [4]. The current state-of-the-art results are achieved by graph neural networks [5, 12, 22, 23, 42, 43, 51] which are applied on a graph representation of a molecule. Each atom of the molecule is represented as a node and each bond is an edge, and the topology is encoded in the connectivity of nodes. Although initial results show the potential of machine learning approaches for chemical application, two fundamental issues remain open: First, labeled data is hard to procure as it requires experiments in wet-labs, which are time and resource constrained making this domain different than others where supervised machine learning has flourished. Second, applications often contain out-of-distribution samples, as training set molecules might be structurally dissimilar from molecules in the test set, e.g., when a chemist might want to assess a newly synthesized molecule that is different from the ones examined so far.

A commonly-used approach to these issues is to pretrain the model using sufficiently large datasets from the same domain, where through the use of proxies for the desired task the model is pretrained to encode information useful for the downstream task.

However, large supervised data are not always available in the chemical domain given the possible candidate space. In this work, we develop a form of a self-supervised learning approach for the chemical domain. We introduce Molecular Inherent SUpervision (MISU), a pretraining method for molecular property prediction that bypasses the need for labeled data by introducing three innovative sources for supervision. Those sources utilize inherent properties of molecular graphs to induce information extraction at different scales, from the local neighborhood of an atom to substructures in

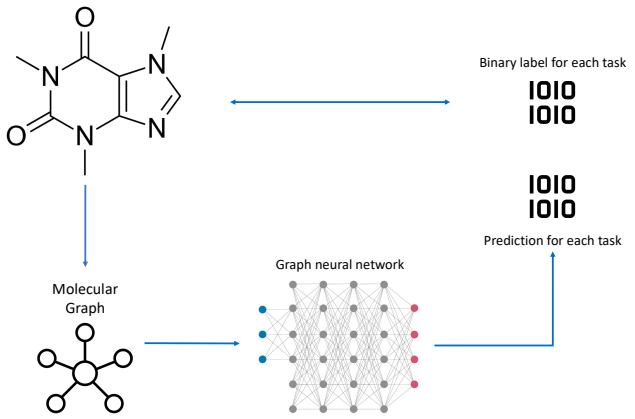

Figure 1: General scheme for molecular property prediction using GNNs. Molecule depicted - Caffeine.

the entire molecule. We propose an end-to-end framework combining the three chemical structure granularities: (1) We leverage molecular fingerprints [7, 32, 39], particular substructures of a molecule, as pseudo-labels during model pretraining. (2) We extract information from local scale through the use of an atom-level Graph VAE. For example, information such as the existence or absence of a bond between two atoms is propagated using this VAE. (3) We extract information from the chemical cluster scale through the use of self-learnt cluster-level Graph VAE. For example, information such as the existence or absence of an edge between two chemical clusters is propagated using this VAE.

We perform empirical evaluation on six molecular property prediction datasets containing hundreds of binary prediction tasks. We show that MISU improves performance of every GNN architecture and reaches state-of-the-art results on all tasks with the GIN [50] GNN architecture. Overall, the method boosts ROC-AUC scores on classification datasets by up to 9.45% and by 5.25% on average compared to non pretrained GNNs.
The contribution of this work is threefold:

(1) We introduce MISU, a self-supervised approach for molecular property prediction that attempts to leverage information from multiple chemical granularities, including molecular fingerprints, atom-level and molecular-cluster level thus providing self-supervision during pretraining.
(2) We introduce an end-to-end framework implementing MISU as a multi-task optimization.
(3) We perform empirical evaluation on 6 different well-known molecular prediction datasets reaching state-of-the-art results.

2 RELATED WORK

Machine learning for molecular property prediction has long been the subject of research [45], initially fixed representations were used, for molecular application the most common are called fingerprints. There are two types of fingerprints, hash based [32, 39] and dictionary based [7]. These were then used as input for traditional machine learning algorithms [41].

Molecular representation learning. Motivated by advances in representation learning, focus had shifted to learned molecular representations, mainly using graph and sequence representations as inputs. For graph-based models molecules are expressed as graphs where heavy atoms are nodes and covalent bonds are edges. These are then fed to a graph neural network to produce a fixed-length representation of the entire molecule. For sequence-based molecules, Simplified Molecular-Input Line-Entry System (SMILES)[1] [48] is common, it is used in conjunction with modules from natural language processing and had shown success for various tasks [2, 3, 16, 27, 47]. We focus on graph-based models in this work.

Graph Neural Networks for molecular property prediction: Molecular tasks had been the target of many works using graph neural networks [15, 29, 46, 50, 51]. Essentially, labeled data is used to train the model to predict the chemical or biological activity of a molecule. However, since labeled data is scarce for chemical applications these method do not solve real-world problems when trying to apply machine learning to molecular property prediction. A general scheme for molecular property prediction through graph neural networks can be seen in Figure 1.

Pretraining schemes for molecular property prediction using GNNs: To address the need for an alternative to traditional machine learning training schemes, pretraining for graph neural networks was the sought after solution. Hu* et al. examined pretraining strategies for GNNs and proposes various tasks for pretraining. However, graph-level tasks discussed require significant labeled data which is often scarce. To this end Rong et al., presented GROVER a pretraining framework that combines message passing networks with transformer-like architecture as well as self-supervised tasks. GROVER uses a motif prediction for the graph-level task, essentially treating functional groups as labels for a multi-label classification problem. However, the choice of which functional group to include in the task required expert knowledge and was manually selected per task. In our method, no such knowledge is required and we empirically show that the manual selection does not scale for other chemical tasks.

To the best of our knowledge, we are the first to present a novel scheme for graph neural network pretraining for molecular property prediction, that does not require any labeled data or expert knowledge bypassing all the caveats mentioned above. We empirically show that inherent supervision using molecular graph properties at different scales reaches state-of-the-art results across most chemical tasks.

3 MOLECULAR INHERENT SUPERVISION

Let $G = (V, E)$ be a molecular graph, where V is the set of atoms, E is the set of bonds connecting the atoms and Y the chemical property label for the molecule. Each node is represented using d features. We denote $X^{|V|\times d}$ as the feature matrix for the nodes in V. The task we consider is graph classification, where the goal is to learn a function $F(\cdot)$ with parameters θ_F, such that given a graph G with features X, $F(G, X) = Y$. Since chemical labeled data is scarce, our goal is to use inherent properties of molecular graphs without

[1]SMILES is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings.

any expert involvement to provide *inherent supervision* for information extraction as these allow for the use of large scale datasets (containing only unlabeled graphs of molecules). The effect of pretraining is an initial model $F(\cdot)$ that is better suited for finetuning in the settings of chemical applications of machine learning (i.e low-data regime). The result is a model that achieves better results than otherwise possible. Our approach relies on three different components and losses to promote rich information extraction at different scales, from the local neighborhood of nodes (i.e., whether an edge exists) to the existence of substructures in the molecular graph.

We introduce Molecular Inherent SUpervision (MISU) a pretraining framework for molecular property prediction. The architecture is illustrated in Figure 3. Given an input molecule, represented by a graph G, we first create an embedding for each atom in V using a GNN encoder (Section 3.1). These embeddings are then used in three different ways to promote information extraction on various scales: (1) Section 3.2: using variational graph autoencoder (VGAE) (depicted in purple in Figure 3). (2) Section 3.3: using an adaption of VGAE for molecular clusters (depicted in red in Figure 3). Jin et al. presented promising results of molecular graph generation by using information on a molecular cluster level rather than on an atom level. They presented the concept of junction trees as spanning tree of the chemical cluster graphs, which we leverage. (3) Section 3.4: using the molecular fingerprint of the graph as pseudo-labels for prediction (depicted in green in Figure 3). The end-to-end pretraining MISU architecture is presented in Section 3.5.

3.1 GNN Encoder

The first component in our framework is the GNN backbone, En, which we would like to pretrain. GNNs use the node and edge features and connectivity properties of the graph to produce a representation $\langle h_v \rangle$ for any given node $v \in V$, which are used to create a representation for the entire graph $\langle g \rangle$. Common GNN architectures use the neighborhood of a given node v, i.e., the set $N(v) = \{u : \exists e_{uv} \in E\}$, to aggregate the representation of each node. Pooling is then used to group together the embeddings of all the nodes in the graph into a single representation for G. For our framework we mainly use Graph Isomorphism Network (GIN) [50], a key advantage of GIN is the aggregation making it one of the most expressive GNN architectures [20]. The aggregation function for the $k-th$ layer of GIN is:

$$\langle h_v \rangle^{(k)} = MLP\left((1+\epsilon^{(k)}) \cdot \langle h_v \rangle^{(k-1)} + \sum_{u \in N(v)} \langle h_u \rangle^{(k-1)} \right), \quad (1)$$

where $\langle h \rangle^{(0)} = X$, MLP stands for multi-layer perceptron and ϵ is a learnable parameter responsible for weighting the current representation with regards to the neighborhood of v. In addition for our encoder we also use a virtual node [14], augmenting the graph by adding a node that is connected to all other nodes in the input graph. Adding it was shown to be effective for many molecular property prediction tasks in [19]. Following [18, 20] edge features are used in conjunction with node features for embedding the input graph. The encoder can be seen in Figure 3 in blue. Note that the arrow from the GNN encoder to the molecular fingerprint prediction symbolized the use of multilayer node embeddings, whereas the other arrow indicate the use of the embeddings from the last layer alone.

3.2 Graph Decoder

Denote A as the adjacency matrix for graph G and $|V| = N$. X is the $N \times d$ matrix of node features where each node embedding is of size d. The inference model is parameterized by our GNN encoder: $q(Z|X,A) = \prod_{i=1}^{N} q(z_i|X,A)$, with $q(z_i|X,A) = \mathcal{N}(z_i|\mu_i, diag(\sigma_i^2))$. Where z_i are the latent variables that are in Z. We use the output of the encoder for $\mu = En(X)$ and similarly for σ.

3.2.1 Inner Product Decoder. For decoding the embeddings of each node we use an inner product decoder, that is defined by an inner product between two latent variables z_i, z_j For decoding back to the adjacency matrix A given Z, we use the dot product of all pairs z_i and z_j to reconstruct the corresponding entry in the adjacency matrix A_{ij}:

$$p\left(A|Z\right) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij}|z_i, z_j\right),$$

where $p\left(A_{ij} = 1|z_i, z_j\right) = \sigma(z_i^T z_j)$, A_{ij} are the elements of the adjacency matrix A and $\sigma(\cdot)$ is the sigmoid function.

3.2.2 Optimization. For optimization we use the following loss:

$$\mathcal{L}_{\text{VGAE}} = \mathbb{E}\left[\log p(A|Z)\right] - \lambda_G KL\left[q(Z|X,A)||p(Z)\right], \quad (2)$$

where λ_G is a weighting hyper-parameter for the Kullback-Leibler (KL) divergence. This promotes valuable information extraction from the immediate local neighborhood of a node thus capturing interactions between atoms. The graph decoder can be seen in Figure 3 in purple.

3.3 Junction Tree Decoder

Jin et al. showed that representing molecules on an atom-level might generate chemically invalid intermediaries during a graph generation task. For example, an aromatic bond is chemically invalid on its own unless the entire aromatic ring is present. They suggest a method that mines the training data using a tree decomposition algorithm and generates tree structured objects, aka junction trees, which represent chemically-valid subgraph components and their coarse relative arrangements. In this work, we are the first to use this scale resolution for a graph classification task.

Formally, for a graph $G = (V, E)$, $\mathcal{T}_G = (V_\mathcal{T}, E_\mathcal{T})$ is the junction tree of G with node set $V_\mathcal{T} = \{C_i\}_{i=1}^{n}$ and C_i is a subgraph of G. Using the tree decomposition algorithm [22], our method creates a mapping from the graph to its tree s.t. the union of all clusters C_i results in the original graph G. Note that a node in G can contribute to multiple nodes in $\mathcal{T}$ as it may take part in more than one cluster.

3.3.1 Tree Decomposition. The tree decomposition mapping is denoted as $Tr(\cdot)$, it returns a junction tree t and the corresponding embedding $X_\mathcal{T} = \langle t \rangle$. The embedding is created using sum pooling on the embeddings from the last layer of En $\langle h \rangle_L$, meaning the embedding for each node in $\mathcal{T}$ is the sum of all the nodes in G mapped to it by Tr. Denote $C_i \in \mathcal{T}$, the embedding $\langle C_i \rangle$ is computed as such:

$$\langle C_i \rangle = \sum_{j \in C_i} \langle h_j \rangle_L.$$

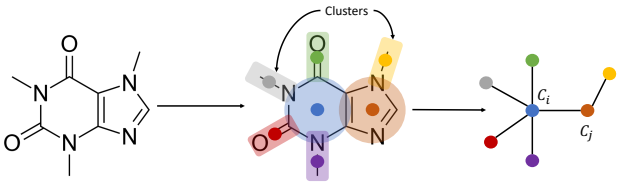

Figure 2: A visualization of the tree decomposition algorithm.

The embedding for all the nodes in $\mathcal{T}$ is denoted by $\langle t \rangle$. This embedding is used as an input for the component. See Figure 3 (red) for the the tree decomposition component. A visualization of the tree decomposition algorithm is depicted in figure 2.

3.3.2 Decoder. Following the generation of the junction tree and its corresponding node feature matrix $X_{\mathcal{T}}^{N_{\mathcal{T}} \times d}$, similarly to Section 3.2, we use the same inner product and loss. Formally, denote $A_{\mathcal{T}}$ as the adjacency matrix for junction tree $\mathcal{T}$ (produced by the tree decomposition algorithm), $|V_{\mathcal{T}}| = N_{\mathcal{T}}$ the number of nodes in $\mathcal{T}$ and $Z_{\mathcal{T}}$ the latent variable matrix for $\mathcal{T}$. The loss used is:

$$\mathcal{L}_{\text{JT}} = \mathbb{E}\left[\log p(A_{\mathcal{T}}|Z_{\mathcal{T}})\right] - \lambda_{JT} KL\left[q(Z_{\mathcal{T}}|X_{\mathcal{T}}, A_{\mathcal{T}})||p(Z_{\mathcal{T}})\right]. \quad (3)$$

Note that this is different from Jin et al., as our main purpose is to promote richer information extraction from a larger neighborhood of any node instead of generating valid molecular sequences. The resulting model is capable of better representing interactions on a cluster level. The junction tree decoder can be seen in Figure 3 in red.

3.4 Molecular Fingerprint Prediction

The previous section explored atom and cluster level information. In this section, we explore the use of molecular fingerprints. Fingerprints are binary vectors of varying sizes and are used regularly for chemical applications such as virtual screening. We focus on two types of molecular fingerprints, Morgan fingerprints (also known as extended-connectivity fingerprint ECFP4) [32, 39] and MACCS fingerprint [7]. Both are substructure fingerprints that are known to perform best for small molecules such as drugs. Morgan fingerprint indicates the existence of circular substructures around each atom, these in turn can be indicative of the biological activities of small organic molecules making this kind of fingerprint particularly useful for our case, we use Morgan fingerprint of dimension 2048 as is common among other works[2, 3, 8]. MACCS (Molecular ACCess System) are $166 - bit$ 2D structure fingerprints with each entry corresponding to a specific pattern.

As the Encoder is used after the pretraining phase, we want all layers of the encoder to encounter valuable information. In order to promote information extraction using the molecular fingerprints, we introduce a linear layer for each layer in En denoted as FC_{layer} and attempt to predict the fingerprints, which serve as pseudo-labels. Intuitively, this encourages the model to extract information in such a way that it is possible to recreate the fingerprint using a linear combination of the elements in the embeddings. Formally, given a molecule, for each layer $i = \{1, \ldots, L\}$ of the encoder we take its embedding $\langle h \rangle_i$, utilize pooling to create an embedding for the whole graph $\langle g \rangle_i$. Then, using a linear layer FC_i we treat the fingerprints as binary labels by minimizing the binary cross-entropy (BCE) loss. The latter ensures information about substructures is extracted by each layer, in turn contributing to the ability of the model to extract information on a graph scale. Overall for a given layer i the prediction of the model is:

$$\text{fp'}_i = FC_i(Pool(En(G)_i)),$$

where $En(G)_i$ denote the embeddings of the nodes at the $i - th$ layer. The loss term is:

$$\mathcal{L}_{\text{fp}} = \sum_{i=1}^{L} -\left(\text{fp} \cdot \log(\text{fp'}_i) + (1 - \text{fp}) \cdot \log(1 - \text{fp'}_i)\right).$$

The path for molecular fingerprint prediction can be seen in Figure 3 in green.

3.5 End-to-End Architecture

We introduce MISU, a pretraining framework for molecular property prediction. The framework is illustrated in Figure 3. It is two-staged, the first is pretraining and the second is finetuning.

3.5.1 Pretraining Framework. The pretraining architecture is made up of three distinct components, a graph VAE , an adaption to JTVAE and a molecular fingerprint predictor. We feed the input graph through each part in parallel and combine the losses while introducing three weighting hyperparameters $\{\lambda_i\}_{i=1}^{3}$. Note that for $\mathcal{L}_G, \mathcal{L}_{\text{JT}}$ the weighting hyperparameter multiplies only the reconstruction part of the loss and we write $\lambda_i \cdot L_i$ for simplicity. The detailed algorithm is presented in Algorithm 1.

Algorithm 1 Pretraining algorithm.

Input: G_t training set. G_v validation set. E_{max} epochs for training. L the number of layers in En. **Output**: En pretrained model.

for $epoch = 1, 2, \ldots, E_{max}$ **do**
  Sample mini-batches $g \in G_t$
  $\langle h \rangle = En(g)$
  $g' = De_G(\langle h \rangle_L)$
  $L_1 = \mathcal{L}_{\text{VGAE}}(g, g')$
  $t, \langle t \rangle = Tr(\langle h \rangle_L, g)$
  $t' = De_{JT}(\langle t \rangle)$
  $L_2 = \mathcal{L}_{\text{JT}}(t, t')$
  Calculate fp_g
  for $layer = 1, 2, \ldots, L$ **do**
    $\langle g \rangle_{layer} = Pool(\langle h \rangle_{layer})$
    $L_3 += BCE\left(\text{fp}_g, FC_{layer}(\langle g \rangle_{layer})\right)$
  end for
  $L = \lambda_1 \cdot L_1 + \lambda_2 \cdot L_2 + \lambda_3 \cdot L_3$
  Minimize L using Adam optimizer
  Evaluate the model every V_f epochs on G_v
  if evaluation criterion improves **then**
    save model
  end if
  if no improvement for P evaluations **then**
    stop training
  end if
end for

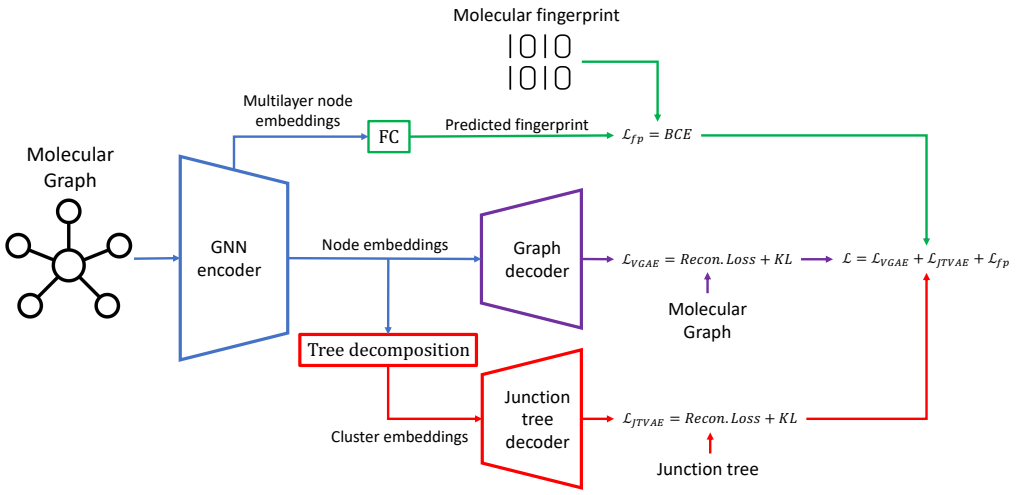

Figure 3: End-to-end pretraining framework composed of a GNN encoder and three components, a variational graph autoencoder (purple), tree decomposition and junction tree decoder (red) and molecular fingerprint as pseudo-labels for prediction (green).

3.5.2 Finetuning. Using the pretrained model En we produce embeddings for all nodes in G. We then pool the embeddings to a single representation, denoted as $\langle g \rangle$, using sum pooling and stack over it a new trainable MLP denoted as FF which produces a prediction. The detailed algorithm is presented in Algorithm 2.

Algorithm 2 Finetuning algorithm.

Pretrained backbone denoted as En, prediction MLP denoted as FF; **Input**: G_t training set with labels $y \in Y_t$. G_v validation set. E_{max} epochs for training. **Output**: $En + FF$ trained predictor.

for $epoch = 1, 2, \ldots, E_{max}$ **do**
  Sample mini-batches $g \in G_t$
  $\langle h \rangle = En(g)$
  $\langle g \rangle = Pool(\langle h \rangle)$
  $y' = FF(\langle g \rangle)$
  $L = BCE(y, y')$
  Minimize L using Adam optimizer
  Evaluate the model every V_f epochs on G_v
  if evaluation criterion improves **then**
    save model
  end if
  if no improvement for P evaluations **then**
    stop training
  end if
end for

4 EMPIRICAL EVALUATION

In this section, we review the implementation and hyperparameter details, the baselines compared and the datasets used and publish the code for reproducibility.

4.1 Datasets

For empirical evaluation, we focus on the molecular property prediction task (i.e., *graph-wise* classification). For pretraining we use the unlabeled version of PCQM4M which contains slightly less than 4 million molecules. The latter is a quantum chemistry dataset curated under the PubChemQC project [34]. Note, that our method does not require any labeled data therefore we use the raw data in PCQM4M ignoring the labels. For finetuning we use the classification datasets in MoleculeNet [49]. We focus on real-world use of chemical applications, that are usually limited to less than 10,000 labeled molecules:

(1) **BACE [44]**: Binary labels of binding results for a set of inhibitors of human β-secretase 1 (BACE-1).

(2) **BBBP [30]**: Binary labels of blood-brain barrier penetration.

(3) **ClinTox [13]**: Binary labels of drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.

(4) **SIDER [1]**: Database of marketed drugs and adverse drug reactions, grouped into 27 system organ classes.

(5) **Tox21 [21, 31]**: Binary labels of toxicity measurements on 12 biological targets, including nuclear receptors and stress response pathways.

(6) **ToxCast [38]**: Toxicology data for a large library of compounds based on in-vitro high-throughput screening.

Note that each dataset may contain more than one binary prediction task. We follow other works [19, 20, 40, 49], and present the metric as averaged scores over the tasks. Detailed description of the classification datasets is shown in Table 1. Positive rate refers to

the percentage of positively labeled molecules for each task. APR stands for Average Positive Rate and MPR stands for Median Positive Rate, both across tasks. Bond and atom density are the average number per molecule for the dataset. An important observation is that BACE contains the most dense molecules with the highest density for both atoms and bonds.

4.2 Data Splitting

To mirror real-world usage we follow [20] and use *scaffold splitting* [37], splitting the molecules according to their molecular structure (scaffold). This is necessary since real-world molecular applications tend to be composed of different structures for training and test data. Therefore we split the pretraining and downstream datasets using *scaffold splitting* to assess the models' performance in an out-of-distribution sense. It is important to note that *scaffold splitting* imitates the way a chemist might go about exploring the available molecules, starting from a lead molecule with a desired property and changing small substructures as the work progresses.

4.3 Baselines

For different applications a different GNN architecture was reported in the literature. We explore numerous such architectures and evaluate over all the datasets the performance of the GNN encoders and compare them to MISU with GIN architecture (i.e., MISU architecture leveraging Graph Isomorphism Network (GIN) architecture [50] as the GNN encoder). Specifically, we compare our method to the GIN vanilla architecture [50] that reached state-of-the-art results on numerous applications [6, 17, 33], vanilla VGAE [25] that showed state-of-the-art performance in link-prediction applications, and GROVER [40] which combines message passing networks with transformer-like architecture that reached state-of-the-art results at molecular property prediction applications [49]. For all baselines we use similarly sized backbones (roughly 6M parameters), the same dataset for pretraining (PCQM4M), same data split for finetuning, metric (ROC-AUC) and computing resources.

4.4 Implementation Details

Testing was done using the RDKit, PyTorch, PyTorch Geometric and PyTorch Lightning frameworks[10, 11, 28, 35], as well as the Adam optimization algorithm [24]. Software: Ubuntu 18.04.6, Pytorch 1.10.2, PyTorch Geometric 2.0.3, PyTorch Lightning 1.5.10, OGB 1.3.2, Python 3.9.10, RDKit 2021.09.4. Reproducibility: We set seed 42, different seeds yield similar results. For all tests we train the models using 10 consecutive seeds starting from 42, and calculate the results. All code can be found online[2].

4.4.1 Pretraining Implementation Details. Pretraining Hardware: 6 Nvidia RTX A6000 48GB GPU, 2 Intel Xeon Gold 6258R 2.70 GHz CPUs. Hyperparameters: Architecture - GIN with 5 layers, number of layers for the MLP used in GIN 5, hidden dimension 600, we use sum as graph pooling. Training - optimizer Adam [24], mini-batch size 5120, dropout 0.5, weight decay $1 \cdot 10^{-7}$, learning rate $3 \cdot 10^{-4}$, max epochs $E_{max} = 80$, evaluation frequency $V_f = 1$, early stopping with $P = 20$ evaluations, $\lambda_G = 0.1$, $\lambda_{JT} = 1 \cdot 10^{-5}$, $\lambda_1 = 1$, $\lambda_2 = 1000$, $\lambda_3 = 100$. Hyperparameters were chosen using a validation set. Data - scaffold split, we shuffle the data during training. Runtime - roughly 16 hours.

4.4.2 Finetuning Implementation Details. Finetuning Hardware: Nvidia GeForce RTX 2080 Ti 11GB GPU, 2 Intel Xeon Gold 6230 2.10GHZ CPUs. Hyperparameters: Architecture - Pretrained GIN as in 4.4.1, number of layers in FF 3, we use sum as graph pooling. Training - optimizer Adam, mini-batch size 256, dropout 0.5, weight decay $1 \cdot 10^{-7}$, learning rate $3 \cdot 10^{-4}$, max epochs $E_{max} = 150$, evaluation frequency $V_f = 1$, early stopping with $P = 30$ evaluations. Data - scaffold split, we shuffle the data during training. Runtime - between 3 to 15 minutes depending on the size of the dataset.

[2]https://github.com/RoyBenjamin/MISU

5 EXPERIMENTAL RESULTS

We conduct numerous empirical experiments. First, we evaluate MISU GIN performance compared to state-of-the-art GNN architectures (Section 5.1). Then, we conduct thorough ablation experiments to show the contribution of each component in our model (Section 5.2.1). All experiments were done on out-of-distribution prediction (scaffold split).

5.1 Main Results

Table 2 shows the results of all models on all downstream datasets on the test-set using scaffold splitting. We observe that MISU surpasses all baselines on all tasks except the BACE task and reaches competitive performance on BBBP. This result highlights the ability of MISU to promote propagation of information at various scales using the different components. As observed from Table 1, BACE is the dataset with the most dense molecules. This translates to the most dense graphs, as grover is made up of a transformer-like architecture it is able to deal with the size of the graphs better leading to a better result for BACE. In addition, we conjecture that the functional groups used for pretraining correlated well with the substructures in the molecular graph leading to better performance on the task at hand. Note that these does not scale to the rest of the datasets, with a less complicated model reaching better results.

The biggest improvement of the model is observed for the ClinTox task, where MISU improves over the non-pretrained model by 9.45%. ClinTox includes two tasks, (1) clinical trial toxicity (or absence of toxicity) and (2) FDA approval status. When analyzing changes in relative performance from the baseline to MISU on this dataset for the molecules with improved performance the average bond and atom densities are $(27.68, 25.36)$ respectively. Whereas for molecules with lower performance it is $(43.33, 41.66)$. Since the densities for the entire dataset are $(27.88, 26.16)$, one can conclude that for this dataset MISU facilitates the propagation of useful information for relatively small molecules prediction which are the majority, whereas molecules of larger proportions (which are anomalies for this datasets) incur a decrease in useful information extraction.

Additionally, to analyze MISU qualitatively we run the model on BACE dataset, then we collect molecules with changes in prediction for a decision rule with threshold 0.5. We group these into three groups. The first includes positively labeled molecules with better predicition when using MISU, denoted as A. The second includes negatively labeled molecules with better prediction when using MISU, denoted as B. The third includes mixed labeled molecules

Table 1: Finetuning datasets characteristics, MPR - Median Positive Rate, APR - Average Positive Rate.

Dataset	# Tasks	# Compounds	Train Size	Validation Size	Test Size	MPR	APR	Bond density	Atom density	Category
BACE	1	1,513	1,210	151	152	45.67%	45.67%	36.86	34.09	Physiology
BBBP	1	2,039	1,631	204	204	76.51%	76.51%	25.95	24.06	Biophysics
ClinTox	2	1,477	1,181	148	148	7.58%	50.61%	27.88	26.16	Physiology
SIDER	27	1,427	1,141	143	143	66.29%	56.76%	35.36	33.64	Physiology
Tox21	617	7,831	6,264	783	784	4.47%	6.24%	19.29	18.57	Physiology
ToxCast	12	8,576	6,860	858	858	1.28%	2.36%	19.26	18.78	Physiology

Table 2: Evaluation results of our method and other baselines for ROC-AUC. In bold: best metrics.

Dataset	Size	# Tasks	MISU	GROVER	VGAE	GIN
BACE	1,513	1	0.7052 ± 0.0379	**0.8160 ± 0.0122**	0.6597 ± 0.0529	0.6525 ± 0.0524
BBBP	2,039	1	**0.6671 ± 0.0175**	**0.6671 ± 0.0092**	0.6650 ± 0.0175	0.6414 ± 0.0248
ClinTox	1,477	2	**0.7800 ± 0.0434**	0.7040 ± 0.0380	0.7450 ± 0.0283	0.7126 ± 0.0514
SIDER	1,427	27	**0.5973 ± 0.0081**	0.5918 ± 0.0132	0.5851 ± 0.0120	0.5582 ± 0.0189
Tox21	7,831	617	**0.7630 ± 0.0071**	0.7500 ± 0.0035	0.7567 ± 0.0100	0.7524 ± 0.0048
ToxCast	8,576	12	**0.6279 ± 0.0049**	0.6123 ± 0.0056	0.6216 ± 0.0049	0.6185 ± 0.0054

with worse predicition when using MISU, denoted as C. This is depicted in Figure 4. First, we analyze bond and atom densities. For group A we get (50.75, 44.75), bond and atom average densities respectively. For group B we get (52.75, 50.0) and for group C (36.5, 33.25) while the overall is (36.86, 34.09). This goes to show that for the given hyperparameters MISU improves performance for larger molecules in the BACE dataset, as expressed in group A. We conjecture this is due to the richness of information (substructures and functional groups) found in larger molecules. Additionally, from Figure 4 we observe that molecules with shared molecular substructures tend to experience roughly the same change in prediction. This is due to the fact that MISU promotes certain behavior for different substructures in the input graph, this results in various changes in prediction for subsets in the dataset with common substructures.

We conclude that the general approach of MISU boosts performance across all tasks without any expert knowledge. By leveraging external unlabeled datasets with chemical supervision, it is not limited to labeling constraints nor requires expert involvement.

5.2 Ablation Tests

We conduct multiple ablation experiments on MISU models for all the downstream datasets.

5.2.1 Component Analysis. Results of experiments conducted for component analysis are shown in Table 3.

- **No Pretraining.** We evaluate the contribution of the entire pretraining scheme for generalization on downstream tasks. One can notice that for every task MISU improves the performance by a significant margin. We conjecture this is due to the effectiveness of each of the pretraining tasks introduced.

- **No JTVAE.** We appraise the value of our adaption to JTVAE. The architecture of MISU without JTVAE is made up of the VGAE for local relation extraction and the molecular fingerprint prediction for graph-level. This is similar to setting $\lambda_2 = \lambda_{JT} = 0$ in algorithm 1. We then train the encoder using the two remaining components with the relevant hyperparameters as in Section 4.4. We observe that JTVAE had a positive effect contributing to an increase of 2% to the average ROC-AUC.

- **No fingerprints (FP).** We evaluate the importance of molecular fingerprint prediction (FP). The architecture of MISU without FP is made up of the VGAE for local relation extraction and the JTVAE for cluster-level. This is similar to setting $\lambda_3 = 0$ in algorithm 1. We then train the encoder using the two remaining components with the relevant hyperparameters as in Section 4.4. We observe that molecular fingerprint prediction had a positive effect contributing to an increase of 1.2% to the average ROC-AUC.

- **No VGAE.** We test the contribution of VGAE. The architecture of MISU without VGAE is made up of the JTVAE for cluster-level relation extraction and the molecular fingerprint prediction for graph-level. This is similar to setting $\lambda_1 = \lambda_G = 0$ in algorithm 1. We then train the encoder using the two remaining components with the relevant hyperparameters as in Section 4.4. We observe that VGAEa had a

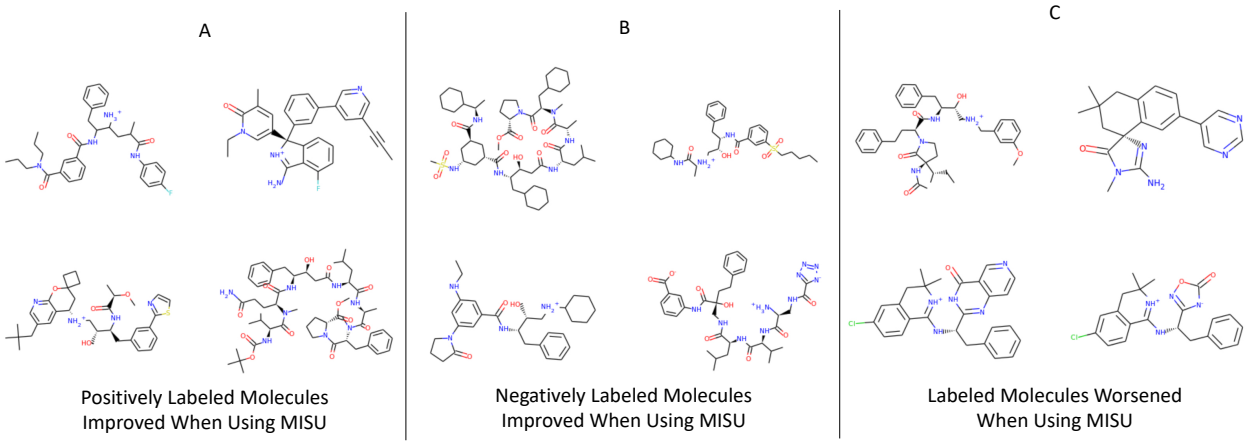

Figure 4: Qualitative Examples

Table 3: Component analysis results of our method. In bold: best metrics.

Dataset	Size	# Tasks	MISU	No VGAE	No FP	No JTVAE	No Pretraining
BACE	1,513	1	**0.7052 ± 0.0379**	0.7034 ± 0.0318	0.7025 ± 0.0403	0.6714 ± 0.0395	0.6525 ± 0.0524
BBBP	2,039	1	**0.6671 ± 0.0175**	0.6650 ± 0.0246	0.6567 ± 0.0205	0.6590 ± 0.0232	0.6414 ± 0.0248
ClinTox	1,477	2	**0.7800 ± 0.0434**	0.7590 ± 0.0452	0.7677 ± 0.0435	0.7607 ± 0.0354	0.7126 ± 0.0514
SIDER	1,427	27	**0.5973 ± 0.0081**	0.5613 ± 0.0134	0.5779 ± 0.0146	0.5837 ± 0.0114	0.5582 ± 0.0189
Tox21	7,831	617	**0.7630 ± 0.0071**	0.7610 ± 0.0074	0.7623 ± 0.0067	0.7616 ± 0.0075	0.7524 ± 0.0048
ToxCast	8,576	12	**0.6279 ± 0.0049**	0.6206 ± 0.0045	0.6219 ± 0.0074	0.6230 ± 0.0065	0.6185 ± 0.0054

positive effect contributing to an increase of 1.74% to the average ROC-AUC.

5.2.2 Different GNN Architectures. Results of experiments conducted to assess the behavior of MISU for different GNN encoderd. For testing we use GIN [50] as specified in Section 4.4, GCN [26] with 5 layers and similar hyperparameters to GIN and DeeperGCN [29] which uses stacking of multiple layers, it is the common model used for leading methods in ogbg-molhiv leaderboard and is used with 14 layers. Results are shown in Table 4. We observe the following:

- **Observation (1).** It is apparent that the most expressive model tested (GIN) achieves the best results when trained using MISU, it gains the most with 5.21% improvement on average performance.

- **Observation (2).** We can see that pretraining GIN outperforms other architectures. This result is consistent with Erhan et al. and follows a similar trend from earlier observations by Hu* et al., which hypothesized that this is due to the expressiveness of the architecture.

- **Observation (3).** From the results we can conclude that the depth of the model contributes less than other factors, in our experiment DeeperGCN uses 14 layers and both GCN and GIN have 5. However, for the non-pretrained result GCN performs slightly better than DeeperGCN on average. We conjecture that 5 layers provide sufficient information about the neighborhood of an atom for the model to extract useful information.

6 CONCLUSIONS

The COVID-19 pandemic had emphasized the importance of rapid drug development. Scientists rely on the drug discovery process to reliably search through a massive search space of chemically viable drug-like molecules. A step in drug discovery is identifying lead molecules with desired properties. Currently targets are singled out through a manual process or high throughput screening. To optimize this process, machine learning is being applied to various steps in drug discovery, potentially allowing scientists to accelerate the process. Research interest started from traditional machine learning models later shifting to SMILES based computational models and graph neural networks. The current state-of-the-art results are achieved by graph neural networks which are applied on a graph representation of a molecule.

Despite initial results for machine learning approaches for molecular application, two concerns prevent machine learning from further integration into the drug discovery process. Firstly, labeled

Table 4: Average ROC-AUC of GNN architectures with and without MISU. In bold: best results, before and after pretraining.

Architecture	Non MISU pretrained	MISU Pretrained	Gain (%)
GIN	0.6559	**0.6901**	**+5.21%**
GCN	**0.675**	0.681	+0.8%
DeeperGCN	0.669	0.6827	+2.05%

data is relatively scarce as creating it requires wet-lab experimentation. Secondly, application frequently include out-of-distribution samples as training and test molecules are often structurally different. The prevailing approach to these challenges is to pretrain the model with large datasets from the same domain. The use of intermediaries for the task enables creating a model pretrained to propagate information better. Since large supervised data is rare in the chemical domain there is a need to develop methods that do not require supervision. To aid in this important issue we create a self-supervision framework for chemical applications.

We present Molecular Inherent SUpervision (MISU), a novel pretraining scheme for graph-based molecular property prediction. Unlike previous work, it is designed in such way that there is no need for labeled data or expert knowledge and uses three sources for self-supervision. We call this type *inherent supervision* as it depends only on chemical and graph based characteristics. Each source contributes to information extraction at different scale from the immediate surroundings of an atom to substructures in the molecule.

We compare our method with other frameworks on six different well-known molecular prediction datasets reaching state-of-the-art results. We conduct a thorough ablation experiment and emphasize the contribution of each component in the method. In addition, we explore the effect of MISU on various GNN architectures and find our method is consistent with work done on supervised pretraining. Altogether, our framework improves the ROC-AUC metric on classification datasets by up to 9.45% and by 5.25% on average compared to the non pretrained GNNs.

We leave it as future work to explore the effect of MISU on regression datasets as well as other GNN architectures. We trust that our method lays a strong foundations for the integration of machine learning as a standard tool in the drug discovery process possibly shortening the time it takes to find a cure for various diseases.

REFERENCES

[1] Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay Pande. 2016. Low Data Drug Discovery with One-shot Learning. https://doi.org/10.48550/ARXIV.1611.03199

[2] Guy Barshatski, Galia Nordon, and Kira Radinsky. 2021. *Multi-Property Molecular Optimization Using an Integrated Poly-Cycle Architecture.* Association for Computing Machinery, New York, NY, USA, 3727–3736. https://doi.org/10.1145/3459637.3481938

[3] Guy Barshatski and Kira Radinsky. 2021. Unpaired Generative Molecule-to-Molecule Translation for Lead Optimization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21).* Association for Computing Machinery, New York, NY, USA.

[4] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. 2018. The rise of deep learning in drug discovery. *Drug discovery today* 23, 6 (2018), 1241–1250.

[5] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018).

[6] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34, 2 (1991), 786–797.

[7] Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. 2002. Reoptimization of MDL keys for use in drug discovery. *Journal of chemical information and computer sciences* 42, 6 (2002), 1273–1280.

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

[9] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning?. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 201–208.

[10] William Falcon et al. 2019. PyTorch Lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

[11] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds* (New Orleans, USA). https://arxiv.org/abs/1903.02428

[12] Tianfan Fu, Cao Xiao, and Jimeng Sun. 2020. CORE: Automatic Molecule Optimization Using Copy & Refine Strategy. In *34th AAAI Conference on Artificial Intelligence (AAAI-20).*

[13] Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. 2016. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology* 23, 10 (2016), 1294–1301.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning.* PMLR, 1263–1272.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[16] Shahar Harel and Kira Radinsky. 2018. Accelerating Prototype-Based Drug Discovery Using Conditional Diversity Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18).* 331–339.

[17] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. 2001. The predictive toxicology challenge 2000–2001. *Bioinformatics* 17, 1 (2001), 107–108.

[18] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *arXiv preprint arXiv:2103.09430* (2021).

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 22118–22133. https://proceedings.neurips.cc/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf

[20] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations.*

[21] Ruili Huang, Menghang Xia, Dac-Trung Nguyen, Tongan Zhao, Srilatha Sakamuru, Jinghua Zhao, Sampada A Shahane, Anna Rossoshek, and Anton Simeonov. 2016. Tox21Challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science* 3 (2016), 85.

[22] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *International Conference on Machine Learning.* 2323–2332.

[23] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. 2019. Learning Multimodal Graph-to-Graph Translation for Molecule Optimization. In *International Conference on Learning Representations.*

[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015,*

San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

[25] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Neural Information Processing Systems Workshop on Bayesian Deep Learning*. http://arxiv.org/abs/1611.07308.

[26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[27] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1945–1954. https://proceedings.mlr.press/v70/kusner17a.html

[28] G Landrum. 2016. Rdkit: Open-source cheminformatics software.

[29] Guohao Li, Chenxin Xiong, Ali K. Thabet, and Bernard Ghanem. 2020. DeeperGCN: All You Need to Train Deeper GCNs. *ArXiv* abs/2006.07739 (2020).

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

[31] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. 2016. DeepTox: toxicity prediction using deep learning. *Frontiers in Environmental Science* 3 (2016), 80.

[32] H. L. Morgan. 1965. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation* 5, 2 (1965), 107–113.

[33] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. www.graphlearning.io

[34] Maho Nakata and Tomomi Shimazaki. 2017. PubChemQC Project: A Large-Scale First-Principles Electronic Structure Database for Data-Driven Chemistry. *Journal of Chemical Information and Modeling* 57, 6 (2017), 1300–1308. https://doi.org/10.1021/acs.jcim.7b00083

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8026–8037.

[36] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. 2013. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of computer-aided molecular design* 27, 8 (2013), 675–679.

[37] Bharath Ramsundar, Peter Eastman, Patrick Walters, and Vijay Pande. 2019. *Deep learning for the life sciences: applying deep learning to genomics, microscopy, drug discovery, and more*. O'Reilly Media.

[38] 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, et al. 2016. ToxCast chemical landscape: paving the road to 21st century toxicology. *Chemical research in toxicology* 29, 8 (2016), 1225–1251.

[39] D. Rogers and Mathew Hahn. 2010. Extended-Connectivity Fingerprints. *Journal of chemical information and modeling* 50 5 (2010), 742–54.

[40] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12559–12571. https://proceedings.neurips.cc/paper/2020/file/94aef38441efa3380a3bed3faf1f9d5d-Paper.pdf

[41] Jie Shen, Feixiong Cheng, You Xu, Weihua Li, and Yun Tang. 2010. Estimation of ADME properties with substructure pattern recognition. *Journal of chemical information and modeling* 50, 6 (2010), 1034–1041.

[42] Uriel Singer, Kira Radinsky, and Eric Horvitz. 2020. On biases of attention in scientific discovery. *Bioinformatics* 36, 22-23 (12 2020), 5269–5274. https://doi.org/10.1093/bioinformatics/btaa1036

[43] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. 2020. A deep learning approach to antibiotic discovery. *Cell* 180, 4 (2020), 688–702.

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

[45] Alexandre Varnek and Igor Baskin. 2012. Machine learning methods for property prediction in chemoinformatics: quo vadis? *Journal of chemical information and modeling* 52, 6 (2012), 1413–1437.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[47] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. 2019. Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*. 429–436.

[48] David Weininger. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* 28, 1 (1988), 31–36.

[49] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* 9 (2018), 513–530. Issue 2. https://doi.org/10.1039/C7SC02664A

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

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