

# Supplementary Materials: UniGM: Unifying Multiple Pre-trained Graph Models via Adaptive Knowledge Aggregation

Anonymous Authors

## 1 DATASET

### 1.1 Details of Molecular Datasets

In this section, we provide the detailed information of the molecular datasets used for downstream tasks.

**Molecular Property: Pharmacology** The Blood-Brain Barrier Penetration (BBBP) [23] dataset measures whether a molecule will penetrate the central nervous system. All three datasets, Tox21 [1], ToxCast [35], and ClinTox [9] are related to the toxicity of molecular compounds. The Side Effect Resource (SIDER) [19] dataset stores the adverse drug reactions on a marketed drug database.

**Molecular Property: Physical Chemistry** Dataset proposed in [6] measures aqueous solubility of the molecular compounds. Lipophilicity (Lipo) dataset is a subset of ChEMBL [8] measuring the molecule octanol/water distribution coefficient. CEP dataset is a subset of the Havard Clean Energy Project (CEP) [11], which estimates the organic photovoltaic efficiency.

**Molecular Property: Biophysics** Maximum Unbiased Validation (MUV) [29] is another sub-database from PCBA, and is obtained by applying a refined nearest neighbor analysis. HIV is from the Drug Therapeutics Program (DTP) AIDS Antiviral Screen [38], and it aims at predicting inhibit HIV replication. BACE measures the binding results for a set of inhibitors of  $\beta$ -secretase 1 (BACE-1) and is gathered in MoleculeNet [35]. Malaria [7] measures the drug efficacy against the parasite that causes malaria.

**Drug-Target Affinity** Davis [5] measures the binding affinities between kinase inhibitors and kinases, scored by the  $K_d$  value (kinase dissociation constant). KIBA [31] contains binding affinities for kinase inhibitors from different sources, including  $K_i$ ,  $K_d$  and  $IC_{50}$ . KIBA scores [26] are constructed to optimize the consistency among these values.

**Input graph representation.** For simplicity, we use a minimal set of node and bond features that unambiguously describe the two-dimensional structure of molecules following previous works [14]. We use RDKit [20] to obtain these features.

- Node features:
  - Atom number: 1 ~ 118
  - Chirality tag:  
{unspecified, tetrahedral cw, tetrahedral ccw, other}
- Edge features:
  - Bond type: {single, double, triple, aromatic}
  - Bond direction: {-, endupright, enddownright}

### 1.2 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 [39]. The original PPI networks do not have node attributes, but 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.

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, 4]. 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

**Table 1: Summary for the molecule datasets for downstream tasks.**

Dataset	Task	# Tasks	# Molecules	# Proteins	# Molecule-Protein
BBBP	Classification	1	2,039	–	–
Tox21	Classification	12	7,831	–	–
ToxCast	Classification	617	8,576	–	–
Sider	Classification	27	1,427	–	–
ClinTox	Classification	2	1,478	–	–
MUV	Classification	17	93,087	–	–
HIV	Classification	1	41,127	–	–
Bace	Classification	1	1,513	–	–
Delaney	Regression	1	1,128	–	–
Lipo	Regression	1	4,200	–	–
Malaria	Regression	1	9,999	–	–
CEP	Regression	1	29,978	–	–
Davis	Regression	1	68	379	30,056
KIBA	Regression	1	2,068	229	118,254

latter, and is much harder to derive experimentally. The GO hierarchy information is obtained using GOATOOLS [18]. 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.

### 1.3 Details of Bibliography Datasets

We also adopt PreDBLP, a new compilation of bibliographic graphs. We derive the new PreDBLP data from AMiner and DBLP. Specifically, PreDBLP contains 1,054,309 paper subgraphs in 31 fields (e.g., artificial intelligence, data mining). Each subgraph is centered at a paper and contains the associated information of the paper. The original Aminer/DBLP contains both the records of each paper and the implicit relations between papers, authors, venues and keywords. For each paper record in the Aminer/DBLP data, we generate a subgraph centered on the paper as follows: (1) according to the citation relationship, we perform 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) we include the selected paper nodes and all the edges between those paper nodes into the subgraph; (3) we convert the authors attached to each paper’s record to nodes as well, and link them to the paper; (4) we utilize the same procedure as in (3) to incorporate the information of venues and keyword terms. As a result, each subgraph compiled contains four types of nodes (i.e., paper, author, venue and keywords) and edges (i.e., paper-paper, paper-author, paper-venue, paper-keywords).

We further utilize a set of node and edge features for the subgraph. For each subgraph, we set the node/edge features as their corresponding types. For instance, for nodes  $u$  and  $v$  connected via edge  $(u, v)$ , the feature of  $u$  and  $v$  are their respective type and that of edge  $(u, v)$  is the type of  $(u, v)$ . During the pre-training process, we utilize 794,862 subgraphs that belong to 25 research fields to pre-train a GNN model. On average, each subgraph contains 262.43 nodes and 900.07 edges. In fine-tuning, we predict the research field of 299,447 labeled subgraphs from the remaining 6 research fields, including: Artificial intelligence (86,956 subgraphs), Computational

linguistics (20,024 subgraphs), Computer Vision (95,729 subgraphs), Data mining (14,934 subgraphs), Databases (68,287 subgraphs) and Fuzzy systems (13,517 subgraphs).

## 2 DETAILS OF GNN ARCHITECTURES

### 2.1 Molecular Property Prediction

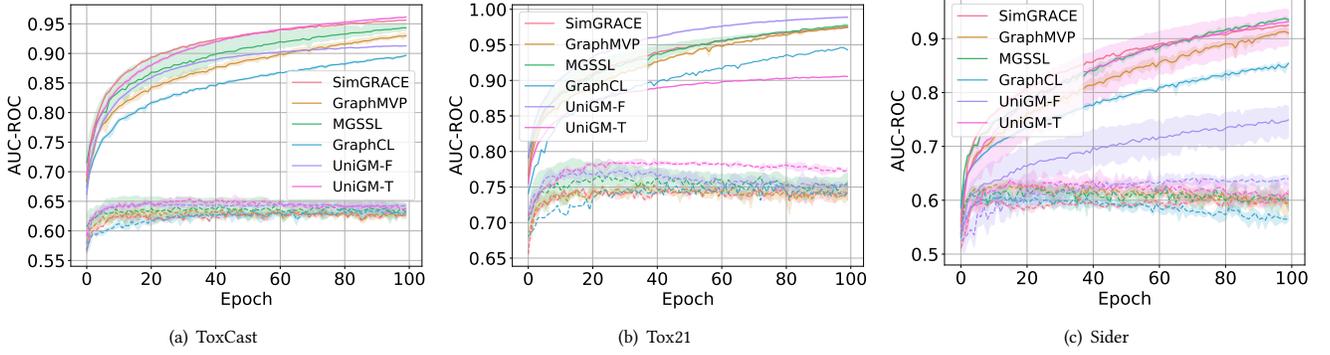
As we describe in the main text, we use the GIN architecture as the main encoder. To incorporate edge features, following previous works [15], we make some minor modifications to include bond features. Specifically, the raw node features and edge features are both 2-dimensional categorical vectors, denoted 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 nodes/edges as well as self-loop edges. As input features to GNNs, we first embed the categorical vectors by

$$\begin{aligned} h_v^{(0)} &= \text{EmbNode}_1(i_{v,1}) + \text{EmbNode}_2(i_{v,2}) \\ h_e^{(k)} &= \text{EmbEdge}_1^{(k)}(j_{e,1}) + \text{EmbEdge}_2^{(k)}(j_{e,2}) \text{ for} \\ &k = 0, 1, \dots, K - 1, \end{aligned}$$

where  $\text{EmbNode}_1(\cdot)$ ,  $\text{EmbNode}_2(\cdot)$ ,  $\text{EmbEdge}_1^{(k)}(\cdot)$ , and  $\text{EmbNode}_1^{(k)}(\cdot)$  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 by

$$\begin{aligned} h_v^{(k)} &= \text{ReLU}(\text{MLP}^{(k)}(\sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(k-1)} \\ &+ \sum_{e=(v,u):u \in \mathcal{N}(v) \cup \{v\}} h_e^{(k-1)})), \end{aligned} \quad (1)$$

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. (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.



**Figure 1: Training (solid lines) and testing (dashed lines) curves of various GMs, UniGM-F, and UniGM-T on ToxCast, Tox21 and Sider datasets.**

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

$$h_G = \text{Mean} \left( \{h_v^{(K)} \mid v \in G\} \right). \quad (2)$$

The label prediction is made by a linear model on top of  $h_G$ . In our experiments, we set the embedding dimension  $d$  to 300. For MLPs in Eq. (1), we use the ReLU activation with 600 hidden units. We apply batch normalization right before the ReLU in Eq. (1) and apply dropout to  $h_v^{(k)}$  at all the layers except the input layer.

**Other GNN architectures.** For GCN, GraphSAGE, and GAT, we adopt the implementation in the Pytorch Geometric library, where we set the number of GAT attention heads to be 2. The dimensionality of node embeddings as well as the number of GNN layers are kept the same as GIN. These models do not originally handle edge features. We incorporate edge features into these models similarly to how we do it for the GIN; we add edge embeddings into node embeddings, and perform the GNN message-passing on the obtained node embeddings.

## 2.2 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 Section 1.2 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

$$\begin{aligned} h_v^{(0)} &= X \\ h_e^{(k)} &= Wc_e + b \quad \text{for } k = 0, 1, \dots, K-1, \end{aligned}$$

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 by

$$h_v^{(k)} = \text{ReLU}(\text{MLP}^{(k)}(\text{CONCAT}(\sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(k-1)}, \sum_{e=(v,u):u \in \mathcal{N}(v) \cup \{v\}} h_e^{(k-1)}))), \quad (3)$$

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

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

## 2.3 Research Field Prediction

In research field prediction, the raw node features are 4-dimensional one-hot vectors, denoted as  $x_v \in \mathbb{R}^4$  for node  $v$ . The raw edge features are 1-dimensional type vector indicating the type of edge, denoted as  $z_{uv} \in \mathbb{R}^1$  (see Appendix B for details). As input features to GNNs, we first embed the feature vectors by

$$\begin{aligned} h_v^0 &= \mathbf{W}^{\text{node}} x_v + \mathbf{b}^{\text{node}} \\ h_{uv}^l &= \mathbf{W}^{\text{edge}} z_{uv} + \mathbf{b}^{\text{edge}} \quad \text{for } l = 0, 1, \dots, L-1, \end{aligned}$$

where  $\mathbf{W}^{\text{node}}, \mathbf{b}^{\text{node}}, \mathbf{W}^{\text{edge}}$  and  $\mathbf{b}^{\text{edge}}$  are learnable parameters. At each layer, GNNs update node representations by

$$h_v^l = \text{RELU}(\text{MLP}^l(\text{CONCAT}(\sum_{u \in \mathcal{N}_u \cup \{v\}} h_u^{l-1}, \sum_{e_{uv}:u \in \mathcal{N}_u \cup \{v\}} h_e^{l-1}))), \quad (5)$$

where  $\text{CoNCAT}(\cdot)$  takes two vectors as input and concatenates them, and  $\mathcal{N}_u$  is a set of nodes adjacent to node  $v$ . Note that we remove the ReLU activation in the final layer so as to output negative values in  $h_v^l$ .

With the aggregation and update of node/edge features, we generate node embeddings at final layer  $l$  to obtain the graph-level representation  $\mathbf{h}_G$ :

$$\mathbf{h}_G = \text{MLP} \left( \text{MEAN} \left( \{h_v^l \mid v \in \mathcal{G}\} \right) \right),$$

where  $\text{MEAN}$  is the mean pooling operation and  $\Omega(\cdot) = \text{MLP}(\text{MEAN}(\cdot))$  is the graph-level pooling calculation.

## 3 VARIOUS UNIONS OF GMS

In addition to the Unions (GraphCL, GraphMVP, SimGRACE, MGSSL) of GMs as we used in the experiments, we also try various unions of

**Table 2: Various Unions of GMs in UniGM-T.**

Unions	ToxCast	Tox21
GraphCL + SimGRACE + JOAO + AD-GCL	76.2(1.0)	63.8(0.9)
Attrmask + GraphLoG + InfoGraph + GPT-GNN	77.8(0.3)	65.1(0.6)
GraphCL + SimGRACE + MGSSL + GraphMVP	<b>78.0(0.8)</b>	<b>65.3(0.3)</b>

GMs in Table 2. The results are reported with UniGM-T. As can be observed, the union of GMs (GraphCL, SimGRACE, JOAO, AD-GCL) with similar pre-training tasks (contrastive tasks based on instance discrimination) is inferior to the union of GMs with diverse tasks. The union of GMs including MGSSL and GraphMVP that introduce extra domain knowledge is conducive to the UniGM.

#### 4 BROADER RANGE OF DOWNSTREAM TASKS

We report the performance of UniGM in regressive property prediction and Drug-target affinity (DTA) tasks in Table 3. DTA is a crucial task in drug discovery, where we aim to predict the affinity scores between the molecular drugs and protein targets. We follow the settings of a recent work [25] on DTA which models the molecular graphs with GNN and target protein (as an amino-acid sequence) with convolution neural network (CNN). We substitute the GNN in their approach with pre-trained GNNs. The superior performance indicates that UniGM can work well in a broader range of downstream tasks.

#### 5 CAN HE-UNIGM UNIFY HOMOGENEOUS MODELS?

In this section, we verify that He-UniGM can also unify homogeneous models. As can be seen in Table 5, He-UniGM achieves better performance than the ensemble. However, He-UniGM underperforms in homogeneous settings than UniGM because He-UniGM is unable to generate layer-dependent aggregation policies.

#### 6 TRAINING AND TESTING CURVES

We plot the training and testing curves of UniGM and single GM in Figure 1. For small-scale datasets such as Sider, we can observe that single GM and UniGM-T are prone to over-fit the training data. UniGM-F can achieve better performance on small-scale datasets due to the less learnable parameters while UniGM-T performs better on larger-scale datasets such as ToxCast and Tox21.

#### 7 IMPLEMENTATION OF BASELINES

For baselines, we optimize their parameters empirically under the guidance of literature. Specifically, we also train the baselines with Adam optimizer with a learning rate of 0.001 and the shared GINs architectures. Other baseline parameters either adopt the original optimal settings or are optimized by the validation set. During the fine-tuning stage, considering that previous works adopt different evaluation protocols, we reproduce all the results with the same protocol as the pioneering work [15] rigorously for fairness. Specifically, we fine-tune the respective publicly available pre-trained models with 10 random seeds (0-9) following the pioneering work.

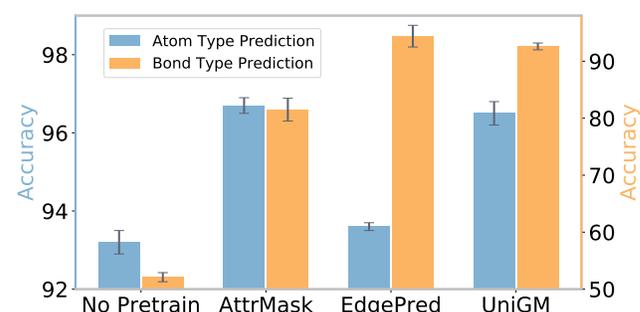
## 8 CLARIFICATIONS ON THE FINE-TUNING SETTINGS

In this section, we clarify the different fine-tuning settings for evaluating the pre-trained GNNs. Specifically, the original paper of GraphLog [37], D-SLA [17], and GraphMAE [13] reported the performance of the last epoch on chemical datasets with an advanced learning rate schedule, which results in unfair comparisons to previous pre-training strategies. For fairness, we reproduce their performance following the standard fine-tuning settings of the pioneering work [15] rigorously.

## 9 GNN ARCHITECTURES OF HETEROGENEOUS GMS

We show the GNN architectures of heterogeneous GMs used in our experiments in Table 5.

## 10 MORE CASE STUDIES



**Figure 2: Comparisons of AttrMask, EdgePred and UniGM in Bond Type Prediction and Atom Type Prediction.**

In addition to *3D Diameter Prediction* and *Atom Type Prediction*, we introduce another task: *Bond Type Prediction*, which means we predict the bonds' type based on their neighboring structure. This task coincides with the pre-training task of EdgePred. We unify the pre-trained GMs including EdgePred and AttrMask here. As can be observed in Figure 2, EdgePred is better at *Bond Type Prediction* while AttrMask is better at *Atom Type Prediction*. The unified model achieves comparable or better performance in the above two tasks simultaneously, which indicates that UniGM can obtain the specialized skills or advantages of each GM.

## 11 COMPARE UNIGM WITH MORE MODEL FUSION METHODS

We also compare UniGM with more model fusion based methods in Table 6. The results illustrate that UniGM is superior to a broad spectrum of methods of this category. Note that there are also some model fusion based methods that are not reported here due to the different applicable scenarios or unavailable codes [3, 33].

**Table 3: Results for molecular property prediction and DTA (regression). We report the mean (and standard variance) RMSE of 3 seeds with scaffold splitting for molecular property prediction, and mean (and standard variance) MSE for 3 seeds with random splitting on DTA tasks. Both indicators are the less the better. The best performance is highlighted in bold.**

Datasets	Molecular Property Prediction				Drug-Target Affinity	
	ESOL	Lipo	Malaria	CEP	Davis	KIBA
No Pre-train	1.178 (0.044)	0.744 (0.007)	1.127 (0.003)	1.254 (0.030)	0.286 (0.006)	0.206 (0.004)
AttMask	1.112 (0.048)	0.730 (0.004)	1.119 (0.014)	1.256 (0.000)	0.291 (0.007)	0.203 (0.003)
ContextPred	1.196 (0.037)	0.702 (0.020)	1.101 (0.015)	1.243 (0.025)	0.279 (0.002)	0.198 (0.004)
JOAO	1.120 (0.019)	0.708 (0.007)	1.145 (0.010)	1.293 (0.003)	0.281 (0.004)	0.196 (0.005)
GraphMVP	1.064 (0.045)	0.691 (0.013)	1.106 (0.013)	1.228 (0.001)	0.274 (0.002)	0.175 (0.001)
Ensemble	1.115 (0.020)	0.685 (0.009)	1.101 (0.013)	1.212 (0.005)	<b>0.265</b> (0.004)	0.171 (0.006)
<b>UniGM-F</b>	<b>0.997</b> (0.025)	<b>0.662</b> (0.015)	1.082 (0.011)	1.198 (0.010)	0.272 (0.001)	0.162 (0.003)
<b>UniGM-T</b>	1.018 (0.032)	0.676 (0.021)	<b>1.075</b> (0.008)	<b>1.187</b> (0.008)	0.282 (0.005)	<b>0.151</b> (0.001)

**Table 4: Comparisons in the homogeneous setting.**

Methods	Tox21	Toxcast
Best single GM	75.9(0.5)	63.4(0.5)
Ensemble	76.6(0.1)	64.1(0.4)
UniGM-F	77.2(0.4)	64.9(0.5)
UniGM-T	<b>78.0</b> (0.5)	<b>65.3</b> (0.3)
He-UniGM	76.8(0.1)	64.5(0.2)

**Table 5: GNN architectures of heterogeneous GMs.**

GNN Type	Number of Layers	Dimension of Hidden Units
GraphCL GCN	6	128
GraphMVP GIN	3	256
SimGRACE GIN	5	300
MGSSL GraphSAGE	4	300

**Table 6: Comparisons with more model fusion methods.**

Methods	Tox21	Toxcast
Elastic Weight Consolidation [21]	75.9(0.8)	62.5(0.6)
OT-fusion [30]	75.7(0.3)	63.0(0.1)
UniGM-F	77.2(0.4)	64.9(0.5)
UniGM-T	<b>78.0</b> (0.5)	<b>65.3</b> (0.3)

## 12 THE NUMBER OF PARAMETERS IN MOLECULAR TASKS (MEMORY CONSUMPTION)

In this section, we compare the number of parameters of various methods during the training and inference stages. As can be observed in Table 7, for the homogeneous setting, the parameters of UniGM are less than ensemble because (1) the number of atom or bond embedding layers is the same as the single model in UniGM (2) the number of parameters in RAPNet can be negligible compared to the scale of GMs. Additionally, as for the heterogeneous setting in Table 8, although He-UniGM trains with more parameters than ensemble, it only uses the unified model (one model) for inference and thus is more memory-efficient.

**Table 7: Comparisons of the number of parameters in the homogeneous setting.**

Methods	Training Parameters	Inference Parameters
Single GM	1.87 M	1.87 M
Ensemble	7.46 M	7.46 M
UniGM	7.28 M	7.28 M

**Table 8: Comparisons of the number of parameters in the heterogeneous setting.**

Methods	Training Parameters	Inference Parameters
GraphCL (6-layer GCN)	0.13 M	0.13 M
GraphMVP (3-layer GIN)	0.84 M	0.84 M
SimGRACE (5-layer GIN)	1.87 M	1.87 M
MGSSL (4-layer GraphSAGE)	0.42 M	0.42 M
Ensemble	3.26 M	3.26 M
He-UniGM	5.33 M	1.87 M

## 13 MORE RELATED WORKS

Transfer learning [27] is a common and effective way to transfer knowledge learned from related tasks to a target task to improve generalization, which have achieved overwhelming success in CV [22], NLP [28] and graph [36] domains. Multiple strategies are developed to fully exploit the knowledge in pre-trained models. For example, in computer vision, SpotTune [10] develops a policy network to make routing decisions on whether to pass the image through the fine-tuned layers or the pre-trained layers. In this way, they address the question of where to fine-tune its parameters with examples of the target task. For pre-trained language models, SMART [16] proposes a smoothness-inducing regularization to prevent over-fitting and trust region-based methods to prevent knowledge forgetting. To alleviate the issue of catastrophic forgetting in graph transfer learning, Han et al. [12] utilize meta-learning to adaptively select and combine different auxiliary tasks with the target task in the fine-tuning stage. However, above works focus on how to transfer knowledge from single pre-trained model. As remedies, recent works aim to evaluate the transferability of pre-trained models to select the best one before tuning [24, 32]. However, these strategies still cannot exploit all the pre-trained models. Another line of works distills knowledge from multiple pre-trained models

to a new one [34]. Despite the effectiveness, these methods need to pass the input data through all models during training, which may cause high computation and memory costs. On the other hand, the distillation process is not conditioned on the input, which may hurt the performance because all the pre-trained models contribute unequally to the downstream tasks. To address these issues, we propose UniGM which can adaptively aggregate various pre-trained models with less computational budget.

## REFERENCES

- [1] 2017. Tox21 challenge. <https://tripod.nih.gov/tox21/challenge/> (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. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics* 25, 1 (2000), 25.
- [3] Stephen Ashmore and Michael Gashler. 2015. A method for finding similarity between multi-layer perceptrons by Forward Bipartite Alignment. In *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.
- [4] Gene Ontology Consortium. 2018. The Gene Ontology resource: 20 years and still GOing strong. *Nucleic Acids Research* 47, D1 (2018), D330–D338.
- [5] Mindy I Davis, Jeremy P Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M Wodicka, Gabriel Pallares, Michael Hocker, Daniel K Treiber, and Patrick P Zarrinkar. 2011. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology* 29, 11 (2011), 1046–1051.
- [6] John S Delaney. 2004. ESOL: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences* (2004).
- [7] Francisco-Javier Gamo, Laura M Sanz, Jaume Vidal, Cristina De Cozar, Emilio Alvarez, Jose-Luis Lavandera, Dana E Vanderwall, Darren VS Green, Vinod Kumar, Samiul Hasan, et al. 2010. Thousands of chemical starting points for antimalarial lead identification. *Nature* 465, 7296 (2010), 305.
- [8] 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. 2012. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research* 40, D1 (2012), D1100–D1107.
- [9] 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* (2016).
- [10] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. 2019. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4805–4814.
- [11] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M Brockway, and Alán Aspuru-Guzik. 2011. The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters* 2, 17 (2011), 2241–2251.
- [12] Xueting Han and others. 2021. Adaptive Transfer Learning on Graph Neural Networks. In *KDD*.
- [13] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. *arXiv e-prints* (2022), arXiv–2205.
- [14] 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, ICLR*.
- [15] Weihua Hu, Bowen Liu, and others. 2020. Strategies for Pre-training Graph Neural Networks. *ICLR* (2020).
- [16] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437* (2019).
- [17] Dongki Kim, Jinheon Baek, and Sung Ju Hwang. 2022. Graph Self-supervised Learning with Accurate Discrepancy Learning. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). <https://openreview.net/forum?id=JY6fLgR8Yq>
- [18] DV Klopfenstein, Liangsheng Zhang, Brent S Pedersen, Fidel Ramirez, Alex Warwick Vesztrocy, Aurélien Naldi, Christopher J Mungall, Jeffrey M Yunes, Olga Botvinnik, Mark Weigel, et al. 2018. GOATOOLS: A Python library for Gene Ontology analyses. *Scientific Reports* 8, 1 (2018), 10872.
- [19] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. 2015. The SIDER database of drugs and side effects. *Nucleic acids research* 44, D1 (2015), D1075–D1079.
- [20] Greg Landrum et al. 2013. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling.
- [21] Mikhail Iu Leontev, Viktoriia Islenteva, and Sergey V Sukhov. 2020. Non-iterative knowledge fusion in deep convolutional neural networks. *Neural Processing Letters* 51, 1 (2020), 1–22.
- [22] Ying Lu, Lingkun Luo, Di Huang, Yunhong Wang, and Liming Chen. 2020. Knowledge transfer in vision recognition: A survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–35.
- [23] 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.
- [24] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. 2020. Leap: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*. PMLR, 7294–7305.
- [25] Thin Nguyen, Hang Le, Thomas P Quinn, Tri Nguyen, Thuc Duy Le, and Svetha Venkatesh. 2021. GraphDTA: Predicting drug–target binding affinity with graph neural networks. *Bioinformatics* 37, 8 (2021), 1140–1147.
- [26] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. 2018. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics* 34, 17 (2018), i821–i829.
- [27] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* (2009).
- [28] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.
- [29] Sebastian G Rohrer and Knut Baumann. 2009. Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data. *Journal of chemical information and modeling* 49, 2 (2009), 169–184.
- [30] Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. *Advances in Neural Information Processing Systems* 33 (2020), 22045–22055.
- [31] Jing Tang, Agnieszka Szwajda, Sushil Shakyawar, Tao Xu, Petteri Hintsanen, Krister Wennerberg, and Tero Aittokallio. 2014. Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. *Journal of Chemical Information and Modeling* 54, 3 (2014), 735–743.
- [32] Anh T Tran, Cuong V Nguyen, and Tal Hassner. 2019. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1395–1405.
- [33] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papaliopoulos, and Yasaman Khazaeni. 2020. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=BklulqSFDS>
- [34] Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2021. One teacher is enough? pre-trained language model distillation from multiple teachers. *arXiv preprint arXiv:2106.01023* (2021).
- [35] 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. *Chemical science* 9, 2 (2018), 513–530.
- [36] Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. 2022. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893* (2022).
- [37] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. 2021. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*. PMLR, 11548–11558.
- [38] Daniel Zaharevitz. 2015. Aids antiviral screen data.
- [39] Marinka Zitnik, Rok Sosič, Marcus W. Feldman, and Jure Leskovec. 2019. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences* 116, 10 (2019), 4426–4433. <https://doi.org/10.1073/pnas.1818013116> arXiv:<https://www.pnas.org/content/116/10/4426.full.pdf>

581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638

639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696