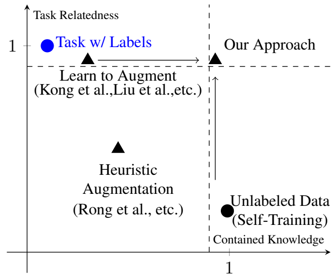
A Additional Related Work on Data-Centric Approach

Data Augmentation Data augmentation creates new examples with preserved labels but uses no unlabeled data (Shorten and Khoshgoftaar, 2019; Kashefi and Hwa, 2020; Balestriero et al., 2022). Examples of heuristic data augmentation techniques include flipping, distorting, and rotating images (Shorten and Khoshgoftaar, 2019), using lexical substitution, inserting words, and shuffling sentences in texts (Kashefi and Hwa, 2020), and deleting nodes and dropping edges in graphs (Zhao et al., 2022). While human knowledge can be used to improve data diversity and reduce over-fitting in heuristic methods, it is difficult to use a single heuristic method to preserve the different labels for different tasks (Balestriero et al., 2022; Cubuk et al., 2019). So, automated augmentation (Cubuk et al., 2019) learned from data to search for the best policy to combine a bunch of predefined heuristic augmentations. Generation models (Antoniou et al., 2017; Bowles et al., 2018; Han et al., 2022) create in-class examples. Other learning ideas such as FATTEN (Liu et al., 2018) and GREA (Liu et al., 2022) learned to split the latent space for data augmentation. However, learning and augmenting from insufficient labels at the same time may limit the diversity of new examples and cause over-fitting. DCT leverages unlabeled data to avoid them.

Relationship between Data-Centric Approaches As presented in Figure 7, perturb edges, delete nodes and mask attributes (Rong et al., 2019; Trivedi et al., 2022) for graphs are some heuristic ways for data augmentation. The augmented knowledge from them is mainly controlled by human prior knowledge on the perturbations and it often fails to be close to the task, *i.e.,* , random perturbations hardly preserve labels for the augmented graphs. The learning to augment approaches learn from labeled graphs to perturb graph structures (Luo et al., 2022), to estimate graphons for different classes (Han et al., 2022), or to split the latent space for augmentation (Liu et al., 2022). Although these approaches could preserve labels for the augmented graphs, they introduce less extra knowledge to improve the model prediction. In summary, graph data augmentation is effective in expanding knowledge for limited labels, but it makes no use of unlabeled graphs. Besides, the diversity and richness of the domain knowledge from augmented graphs are far from that contained in a large number of unlabeled graphs. To learn from unlabeled graphs, data-centric approaches like the self-training is assumed to be useful when the unlabeled and labeled data are from the same source. It is less studied when we have a single unified unlabeled source for different tasks.

Figure 7: Qualitative relationship of graphs from different data-centric approach on the task relatedness and contained knowledge.



B Additional Method Details

B.1 Upper bounding the mutual information

In Eq. (6), we use a leave-one-out variant of InfoNCE ($\mathcal{I}_{\text{bound}}$) to derive the upper bound of mutual information. We summarize the derivation (Poole et al., 2019) here.

$$\begin{aligned} \mathcal{I}_1(G';G) &= \mathbb{E}_{p(G,G')}\left[\log\frac{p(G'|G)}{p(G')}\right] \\ &= \mathbb{E}_{p(G,G')}\left[\log\frac{p(G'|G)q(G')}{q(G')p(G')}\right] \\ &= \mathbb{E}_{p(G,G')}\left[\log\frac{p(G'|G)}{q(G')}\right] - \mathrm{KL}(p(G')||q(G')) \\ &\leq \mathbb{E}_{p(G,G')}\left[\log\frac{p(G'|G)}{q(G')}\right] \end{aligned} \tag{9}$$

The intractable upper bound is minimized when the variational approximation $q(G')$ matches the true marginal $p(G')$ (Poole et al., 2019). For each G_i, its augmented output G'_i, and $M-1$ negative examples with different labels, we could approximate $q(G'_i) = \frac{1}{M-1}\sum_{j\neq i} p(G'_i|G_j)$. So, we have

$$\begin{aligned}\mathcal{I}_1(G'_i, G_i) &\leq \log \frac{p(G'_i|G_i)}{\frac{1}{K-1}\sum_{j=1,j\neq i}^{M} p(G'_i|G_j)} \\ &= \log \frac{p(G'_i|G_i)}{\sum_{j=1,j\neq i}^{M} p(G'_i|G_j)} + \log(M-1) \\ &= \mathcal{I}_{\text{bound}}(G'_i; G_i) + \text{constant}\end{aligned} \tag{10}$$

B.2 Extraction of Statistical Features on Graphs

For each molecule and polymer graph, we concatenate the following vectors or values for statistical feature extraction.

- the sum of the degree in the graph;
- the vector indicating the distribution of atom types;
- the vector containing the maximum, minimum and mean values of atoms weights in a molecule or polymer;
- the vector containing the maximum, minimum, and mean values of bond valence.

For each protein-protein interaction ego-graph in the biology field, we use the sorted vector of node degree distribution in the graph as the statistical features.

B.3 Technical Details for Graph Data Augmentation with Diffusion Model

The Lookup Table from Atom Type to Node Embedding Space Given a graph G, we assume the node feature matrix on the graph is $\mathbf{X} \in \mathbb{R}^{n\times F_{\text{n}}}$, where n is the number of nodes. The edge feature matrix is $\mathbf{E} \in \mathbb{R}^{m\times F_{\text{e}}}$, where m is the number of edges. There are two ways for G to represent the graph structure in practice. We can use either the dense adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ or sparse edge index $\mathbf{I}_e \in \mathbb{R}^{2\times m}$. The diffusion model (Jo et al., 2022) on graphs prefers the former, which is more straightforward for graph generations. The prediction model prefers the latter because of its flexibility, and less computational cost and time. The transformation between two types of graph structure representation takes additional time. Particularly for molecular graphs, the node features used for generation (one-hot encoding of the atom type) and for prediction (see the official package of OGBG [1] for details) are different, which introduces extra time to process the graph data. For details, we (1) first need to extract discrete node attributes given the atom type and its neighborhoods; (2) we then need to use an embedding table to embed node attributes in a continuous embedding space; (3) the embedding features of nodes with their graph structure are inputted into the graph neural networks to get the latent representation for nodes. The reverse process for data augmentation in DCT may need to repeatedly process graph data with steps (1) and (2). It introduces additional time. So, we build up a lookup table to directly map the atom type to the node embedding. To achieve it, we average the node attributes for the same type of node within the batch. We then use the continuous node attributes as weights to average the corresponding node embedding according to the embedding table.

Instantiations of SDE on Graphs According to Song et al. (2021), we use the Variance Exploding (VE) SDE for the diffusion process. Given the minimal noise $\sigma_{\min}$ and the maximal noise $\sigma_{\max}$, the VE SDE is:

$$\mathrm{d}G = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^t \sqrt{2\log\frac{\sigma_{\max}}{\sigma_{\min}}}\mathrm{d}\mathbf{w}, \quad t\in(0,1] \tag{11}$$

The perturbation kernel is derived (Song et al., 2021) as:

$$p_{0t}(G^{(t)} \mid G^{(0)}) = \mathcal{N}\left(G^{(t)}; G^{(0)}, \sigma^2_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{2t}\mathbf{I}\right), \quad t\in(0,1] \tag{12}$$

On graphs, we follow Jo et al. (2022) to separate the perturbation of adjacency matrix and node features:

$$p_{0t}(G^{(t)} \mid G^{(0)}) = p_{0t}(\mathbf{A}^{(t)} \mid \mathbf{A}^{(0)})p_{0t}(\mathbf{X}^{(t)} \mid \mathbf{X}^{(0)}). \tag{13}$$

---

[1] `https://github.com/snap-stanford/ogb/blob/master/ogb/utils/features.py`

The Sampling Algorithm in the Reverse Process for Graph Data Augmentation We adapt the Predictor-Corrector (PC) samplers for the graph data augmentation in the reverse process. The algorithm is shown in Algorithm 1.

Algorithm 1 Diffusion-Based Graph Augmentation with PC Sampling

```
Input: Graph $G$ with node feature $\mathbf{X}$ and adjacency matrix $\mathbf{A}$, the denoising function for node feature $\mathbf{s}_\mathbf{X}$ and adjacency matrix $\mathbf{s}_\mathbf{A}$, the fine-tune loss $\mathcal{L}_{\textbf{aug}}$, Lagevin MCMC step size $\beta$, scaling coefficient $\epsilon_1$
$\mathbf{A}^{(D)} \leftarrow \mathbf{A} + \mathbf{z}_A; \quad \mathbf{z}_A \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\mathbf{X}^{(D)} \leftarrow \mathbf{X} + \mathbf{z}_X; \quad \mathbf{z}_X \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = D - 1$ to $0$ do
    $\hat{G}_{(t+1)} \sim p_{0t+1}(\hat{G}_{(t+1)}|G^{(t+1)})$ {inner-loop sampling with another PC sampler}
    $\mathbf{S}_A = \frac{1}{2}\mathbf{s}_\mathbf{A}(G^{(t+1)}, t+1) - \frac{1}{2}\alpha\nabla_{\mathbf{A}^{(t)}}\mathcal{L}_{\textbf{aug}}(\hat{G}_{(t+1)})$
    $\mathbf{S}_X = \frac{1}{2}\mathbf{s}_\mathbf{X}(G^{(t+1)}, t+1) - \frac{1}{2}\alpha\nabla_{\mathbf{X}^{(t)}}\mathcal{L}_{\textbf{aug}}(\hat{G}_{(t+1)})$
    $\tilde{\mathbf{A}}^{(t)} \leftarrow \mathbf{A}^{(t+1)} + g(t)^2\mathbf{S}_A + g(t)\mathbf{z}_A; \quad \mathbf{z}_A \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Predictor for adjacency matrix}
    $\tilde{\mathbf{X}}^{(t)} \leftarrow \mathbf{X}^{(t+1)} + g(t)^2\mathbf{S}_X + g(t)\mathbf{z}_X; \quad \mathbf{z}_X \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Predictor for node features}
    $\mathbf{A}^{(t)} \leftarrow \tilde{\mathbf{A}}^{(t)} + \frac{\beta}{2}\mathbf{S}_A + \epsilon_1\sqrt{\beta}\mathbf{z}_A; \quad \mathbf{z}_A \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Corrector for adjacency matrix}
    $\mathbf{X}^{(t)} \leftarrow \tilde{\mathbf{X}}^{(t)} + \frac{\beta}{2}\mathbf{S}_X + \epsilon_1\sqrt{\beta}\mathbf{z}_X; \quad \mathbf{z}_X \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Corrector for node features}
end for
return $G' = (\mathbf{A}^{(0)}, \mathbf{X}^{(0)})$
```

The Algorithm of the Framework The algorithm of the proposed data-centric knowledge transfer framework is presented in Algorithm 2 and Algorithm 3. In detail, Algorithm 2 corresponds to Section 4.2 and Algorithm 3 corresponds to Section 4.3.

Algorithm 2 The Data-Centric Knowledge Transfer Framework: Learning from Unlabeled Graphs

```
Input: Given unlabeled graphs from the space $\mathcal{G}^{[U]}$, randomly initialized score models $\mathbf{s}_\mathbf{X}$ and $\mathbf{s}_\mathbf{A}$ for node feature and graph adjacency matrix, respectively, the total diffusion time step $T$.
while $\mathbf{s}_\mathbf{X}$ and $\mathbf{s}_\mathbf{A}$ not converged do
    Sample $G = (\mathbf{X}, \mathbf{A}) \in \mathcal{G}^{[U]}$
    Sample $t \in \mathrm{Uniform}(1, 2, \ldots, T)$
    Sample $\mathbf{z}_A \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Sample $\mathbf{z}_X \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Sample $\hat{G}$ with $t$, $\mathbf{z}_A$, $\mathbf{z}_X$ and Eq. (13)
    Optimize $\mathbf{s}_\mathbf{A}$ with the gradient:
        $\nabla\|\mathbf{z}_A - \mathbf{s}_\mathbf{A}(\hat{G}, t)\|^2$
    Optimize $\mathbf{s}_\mathbf{X}$ with the gradient:
        $\nabla\|\mathbf{z}_X - \mathbf{s}_\mathbf{X}(\hat{G}, t)\|^2$
end while
```

Algorithm 3 The Data-Centric Knowledge Transfer Framework: Generating Task-specific Labeled Graphs

```
Input: Given task $k$ with the graph-label space $(\mathcal{G}, \mathcal{Y})$, a randomly initialized prediction model $f_\theta$, the well-trained score model $\mathbf{s} = (\mathbf{s}_\mathbf{X}, \mathbf{s}_\mathbf{A})$, the training data set $\{G_i, y_i\}_i^{N_t}$, total training epoch $e$, the hyper-paramtere $n$
for current epoch $e_i$ from 1 to $e$ do
    Train $f_\theta$ on current training data $\{G_i, y_i\}_i^{N_t}$
    if $e_i$ is divisible by the augmentation interval then
        Select $n$ graph-label pairs with the lowest training loss from $\{G_i, y_i\}_i^{N_t}$
        Get the augmented examples $\{G'_i, y'_i\}_i^n$ by Algorithm 1 with the selected examples
        Update $\{G_i, y_i\}_i^{N_t}$ with $\{G'_i, y'_i\}_i^n$, e.g., add $\{G'_i, y'_i\}_i^n$ to $\{G_i, y_i\}_i^{N_t}$.
    end if
end for
```

C Additional Experiments Set-ups

We perform experiments on 15 datasets, including eight classification and seven regression tasks from chemistry, material science, and biology. We use Area under the ROC curve (AUC) for classification performance and mean absolute error (MAE) for regression.

C.1 Molecule Classification and Regression Tasks

Seven molecule classification and three molecule regression tasks are from open graph benchmark (Hu et al., 2020). They were originally collected by MoleculeNet (Wu et al., 2018) and used to predict molecule properties. They include (1) inhibition to HIV virus replication in ogbg-HIV, (2) toxicological properties of 617 types in ogbg-ToxCast, (3) toxicity measurements such as nuclear receptors and stress response in ogbg-Tox21, (4) blood–brain barrier permeability in ogbg-BBBP, (5) inhibition to human β-secretase 1 in ogbg-BACE, (6) FDA approval status or failed clinical trial in ogbg-ClinTox,

Table 3: Statistics of datasets for graph property prediction in different domains.

Data Type	Dataset	# Graphs	Prediction Task	# Task	Avg./Max # Nodes	Avg./Max # Edges
Molecules	ogbg-HIV	41,127	Classification	1	25.5 / 222	54.9 / 502
	ogbg-ToxCast	8,576	Classification	617	18.8 / 124	38.5 / 268
	ogbg-Tox21	7,831	Classification	12	18.6 / 132	38.6 / 290
	ogbg-BBBP	2,039	Classification	1	24.1 / 132	51.9 / 290
	ogbg-BACE	1,513	Classification	1	34.1 / 97	73.7 / 202
	ogbg-ClinTox	1,477	Classification	2	26.2 / 136	55.8 / 286
	ogbg-SIDER	1,427	Classification	27	33.6 / 492	70.7 / 1010
	ogbg-Lipo	4200	Regression	1	27 / 115	59 / 236
	ogbg-ESOL	1128	Regression	1	13.3 / 55	27.4 / 124
	ogbg-FreeSolv	642	Regression	1	8.7 / 24	16.8 / 50
Polymers	GlassTemp	7,174	Regression	1	36.7 / 166	79.3 / 362
	MeltingTemp	3,651	Regression	1	26.9 / 102	55.4 / 212
	ThermCond	759	Regression	1	21.3 / 71	42.3 / 162
	O_2Perm	595	Regression	1	37.3 / 103	82.1 / 234
Proteins	PPI	88000	Classification	40	49.4 / 111	890.8 / 11556

(7) having drug side effects of 27 system organ classes in ogbg-SIDER, (8) predicting the property of lipophilicity in ogbg-Lipo, (9) predicting the water solubility ($\log$ solubility in mols per litre) from chemical structures in ogbg-ESOL, (10) predicting the hydration free energy of molecules in water in ogbg-FreeSolv. For all molecule datasets, we use the scaffold splitting procedure as the open graph benchmark adopted (Hu et al., 2020). It attempts to separate structurally different molecules into different subsets, which provides a more realistic estimate of model performance in experiments (Wu et al., 2018).

C.2 Polymer Regression Tasks

Four polymer regression tasks include GlassTemp, MeltingTemp, ThermCond, and O_2Perm. They are used to predict different polymer properties such as *glass transition temperature* ($^\circ$C), *melting temperature* ($^\circ$C), *thermal conductivity* (W/mK) and *oxygen permeability* (Barrer). GlassTemp and MeltingTemp are collected from PolyInfo, which is the largest web-based polymer database (Otsuka et al., 2011). The ThermCond dataset is from molecular dynamics simulation and is an extension from the dataset used in (Ma et al., 2022). The O_2Perm dataset is created from the Membrane Society of Australasia portal, consisting of a variety of gas permeability data (Thornton et al., 2012). Since a polymer is built from repeated units, researchers often use a single unit graph with polymerization points as polymer graphs to predict properties. Different from molecular graphs, two polymerization points are two special nodes (see "$*$" in Figure 2), indicating the polymerization of monomers (Cormack and Elorza, 2004). For all the polymer tasks, we randomly split by 60%/10%/30% for training, validation, and test.

C.3 Protein Classification Task

An additional task is protein function prediction using protein-protein interaction graphs (Hu et al., 2019). A node is a protein without attributes, an edge is a relation type between two proteins such as co-expression and co-occurrence. In our DCT, we treat all the relations as the undirected edge without attributes.

C.4 Baselines and Implementation

When implementing GIN (Xu et al., 2019), we tune its hyper-parameters for different tasks with an early stop on the validation set. We generally implement pre-training baselines following their own setting. For molecule and polymer property prediction and protein function prediction, the pre-trained GIN models with self-supervised tasks such as EdgePred, AttrMask, ContextPred in (Hu et al., 2019), InfoMax (Velickovic et al., 2019) are available. So we directly use them. For other self-supervised methods, we implement their codes with default hyper-parameters. Following their settings, we use 2M ZINC15 (Sterling and Irwin, 2015) to pre-train GIN models for molecule and polymer property prediction. We use 306K unlabeled protein-protein interaction ego-networks (Hu et al., 2019) to pre-train the GIN for the downstream protein function property prediction. For self-training with real unlabeled graphs and INFOGRAPH (Sun et al., 2020), we use 113K QM9 (Ramakrishnan et al., 2014). For self-training with generated unlabeled graphs, we train the diffusion model (Jo et al., 2022) on the real QM9 dataset and then produce the same number of generated unlabeled graphs. To train the diffusion model in our DCT, we also use QM9 (Ramakrishnan et al., 2014).

D Additional Experiment Analysis

D.1 The Power of Diffusion Model to Learn from Unlabeled Graphs

In Table 2, when we replace the 133K QM9 with the 249K ZINC (Jo et al., 2022) to train the diffusion model, which nearly doubles the size of the unlabeled graphs and includes more atom types, we do not observe any additional improvement, and in some cases, even worse performance. It is possible because of the constraint of the current diffusion model's capacity to model the data distribution for a much larger number of more complex graphs. It encourages powerful generative models in the future, which could be directly used to benefit predictive models under the proposed framework.