Yale

Strategy for Improving Pre-trained Model on Graph

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

Strategy for Improving Pre-trained Model on Graph

- **Improved pretrained model with negative sampling**
  - **Contrastive learning**
  - **KDD2023-BatchSampler: Sampling Mini-Batches for Contrastive Learning in Vision, Language, and Graphs**
- **Improved pretrained model with dataset grouping**
  - **Auxiliary molecule dataset**
  - **NeurIPS2023-Learning to group auxiliary dataset for molecule**

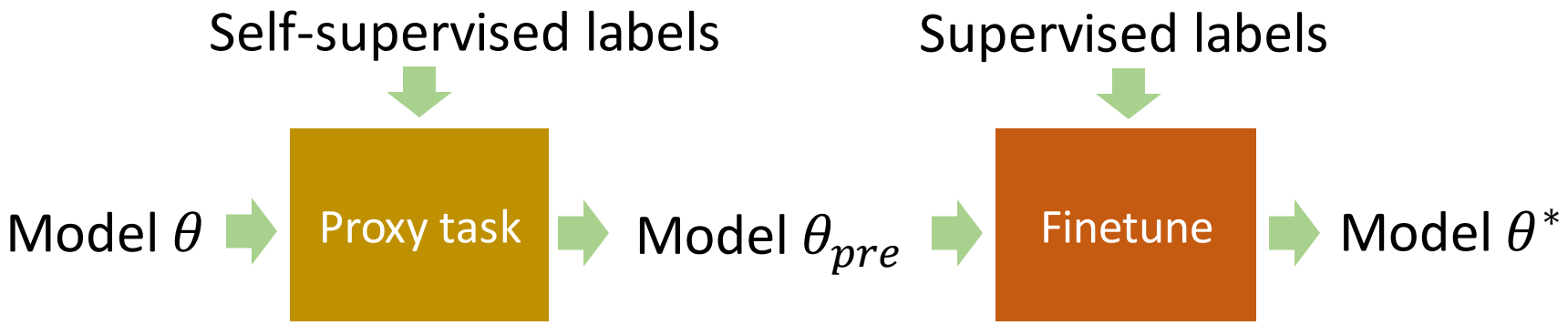
Strategy for Improving Pre-trained Model on Graph

- **Improved pretrained model with negative sampling**
  - **Contrastive learning**
  - **KDD2023-BatchSampler: Sampling Mini-Batches for Contrastive Learning in Vision, Language, and Graphs**

- **Improved pretrained model with dataset grouping**
  - **Auxiliary molecule dataset**
  - **NeurIPS2023-Learning to group auxiliary dataset for molecule**

Self-Supervised Learning (1)

- How to train a model **without** supervised signals?
  - Design a proxy task!
  - Use the supervision signals from the data itself (**self-supervised** learning)

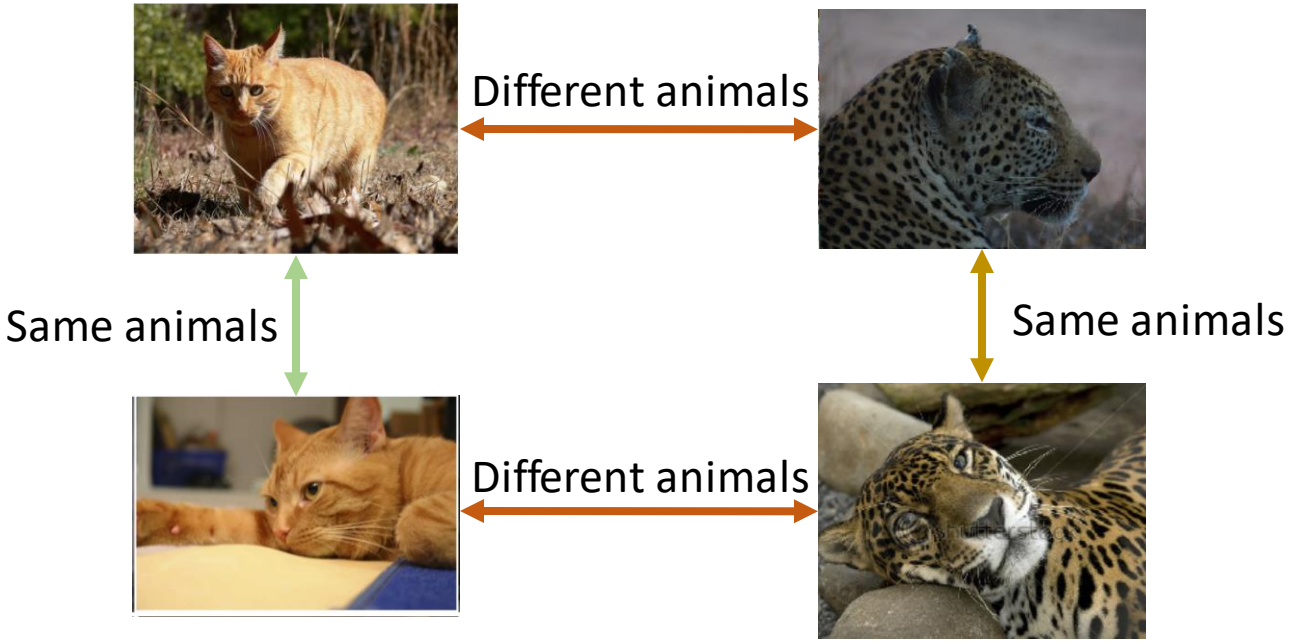


Self-Supervised Learning (2)

- How to design the pretrained proxy task?
- **Generative task** (Lecture 12)
  - Train the model by predicting some foundational properties of the data
  - E.g., molecule atom/bond types, graph structures, …
- **Contrastive task**
  - Train the model by discriminating similar and dissimilar data instances
  - Main idea: instances with similar semantic features typically fall within the same category

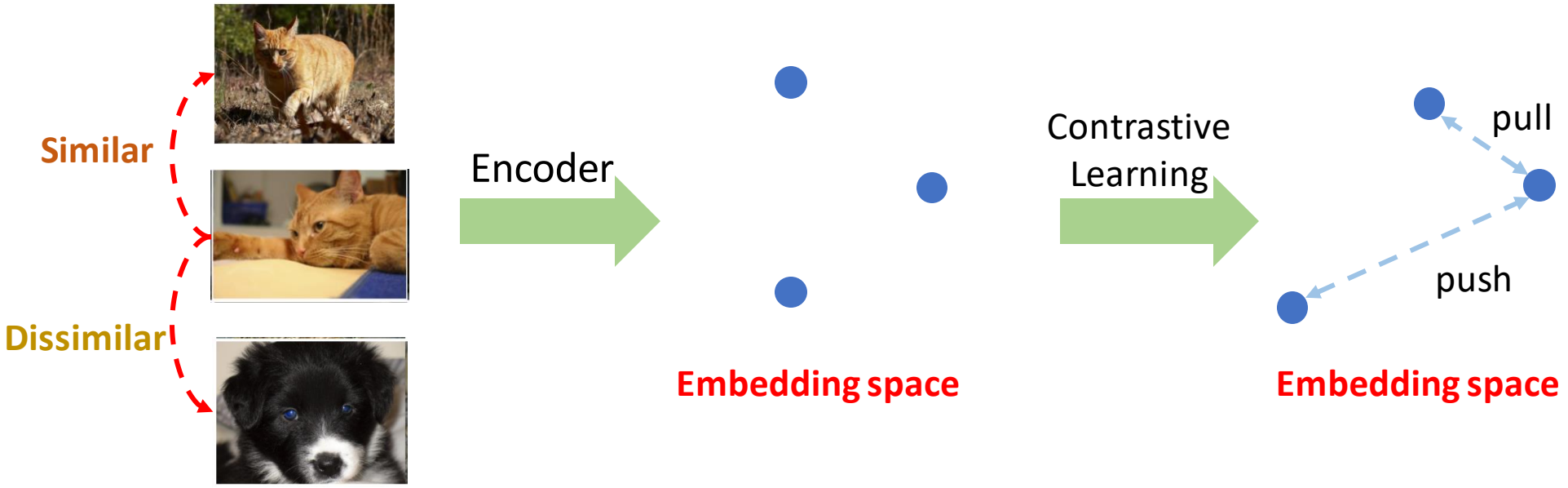
Contrastive Learning (1)

- How do humans differentiate between identical animals?
  - By contrasting fur color, body features, and overall appearance
  - Even without knowing the specific animal species
- For example,



Contrastive Learning (2)

- A pretrained model should be able to exploit the **semantic features**
  - Produce similar representations for similar instances
- Contrastive learning
  - Bringing **semantically similar** instances closer while pushing **dissimilar** instances



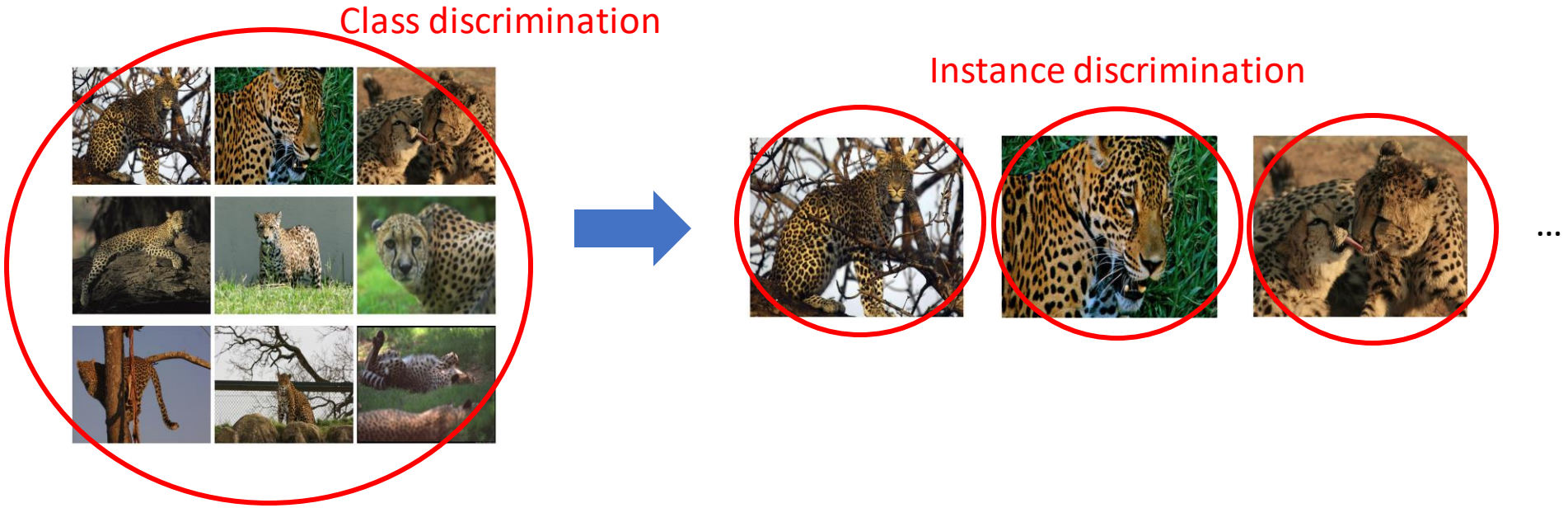
Contrastive Learning (3)

- Basic learning paradigm
  - Sample an instance (**anchor**), along with a semantically similar data point (**positive sample**) and a dissimilar data point (**negative sample**)
  - Minimize/maximize distance between **anchor** and **positive samples**/**negative samples** in latent space

Challenge in the foundation model setting: without labels, how could we know which instance is the positive one with respect to the anchor?

Contrastive Learning (4)

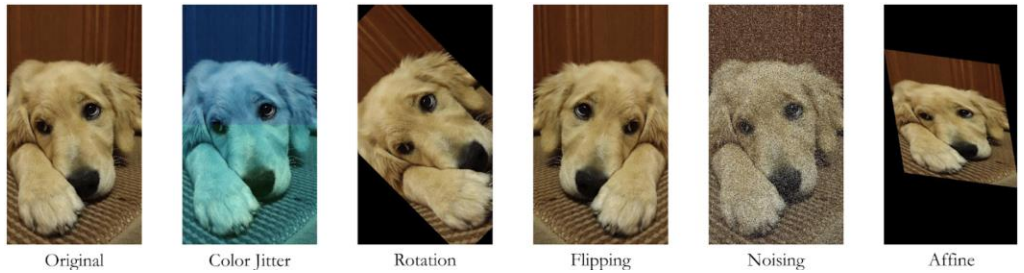
- Instance discrimination method
  - Instances that belong to same class contain similar semantic information
  - By differentiating between every instance, the model implicitly captures this underlying semantic structure



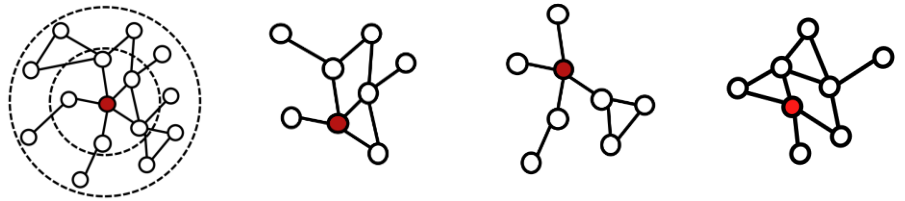
Contrastive Learning (5): Data augmentation

- Apply data augmentation on the **anchor** to generate the **positive sample**
  - The anchor is made to undergo transformations and used as positive sample to the anchor
  - Create diverse yet semantically consistent instances
  - E.g.,

Vision(credit)



Graph(credit)



Contrastive Learning (6): InfoNCE Loss

- Optimized model with InfoNCE loss
  - Sample a mini-batch of instances $\{x_i\}_B$ (**anchor**) and apply data augmentation to generate **positives** $\{x_i^+\}_B$
  - InfoNCE loss

$$\min -\sum_{i=1}^{B} \log \frac{e^{f(x_i)^T f(x_i^+)}}{\sum_j e^{f(x_i)^T f(x_j)}}$$

$$= \min -\sum_{i=1}^{B} \left(\underbrace{f(x_i)^T f(x_i^+)}_{\text{minimize distance}} - \underbrace{\log \sum_j e^{f(x_i)^T f(x_j)}}_{\text{maximize distance}} \right)$$

$f(\cdot)$: encoder
x_i: anchor
x_i^+: positive sample

  - One **anchor** is paired with a **positive** sample, while the remaining instances in the mini-batch serve as **negative** samples

Negative Sampling

- In InfoNCE loss, mini-batch sampling is equivalent to **negative sampling**
  - Every instance serves as negative to the other instances within the mini-batch
  - Different kinds of negative

Similarity



Easy negative



Hard negative



False negative:
Negative with same label

Hard Negative Sampling (1)

- What negatives contribute the most in the mini-batch?
  - **Hard negative pair!**
  - Hard negative is the sample with different label but is semantically similar to anchor
  - Hard-to-distinguish negative pairs provide meaningful gradient to the model

Hard Negative Sampling (2)

- Model learns to create more robust representations that can differentiate between closely related instances
  - E.g.,

Similar background and body shape

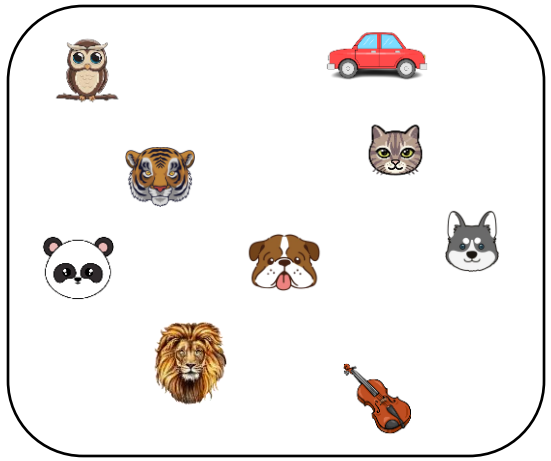
Similar patterns

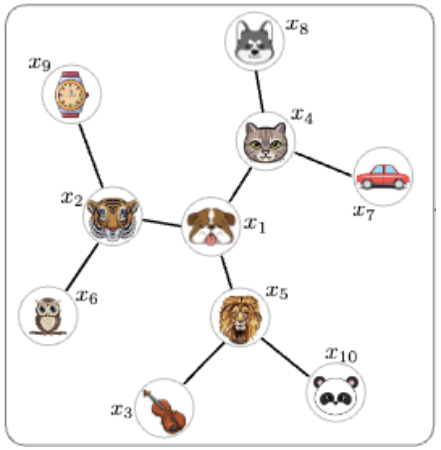
- How to sample hard negatives for InfoNCE loss?
  - Sample a mini-batch of instances where any instance pair are **hard to distinguish**

Proximity Graph (1)

- We use **proximity graph** to capture similarity relationships among the instances
  - Connect nodes based on similarity, ensuring that similar instances form local communities within graph

Representation space:
Instances are dispersed in space and pair with higher similarity will be closer in space





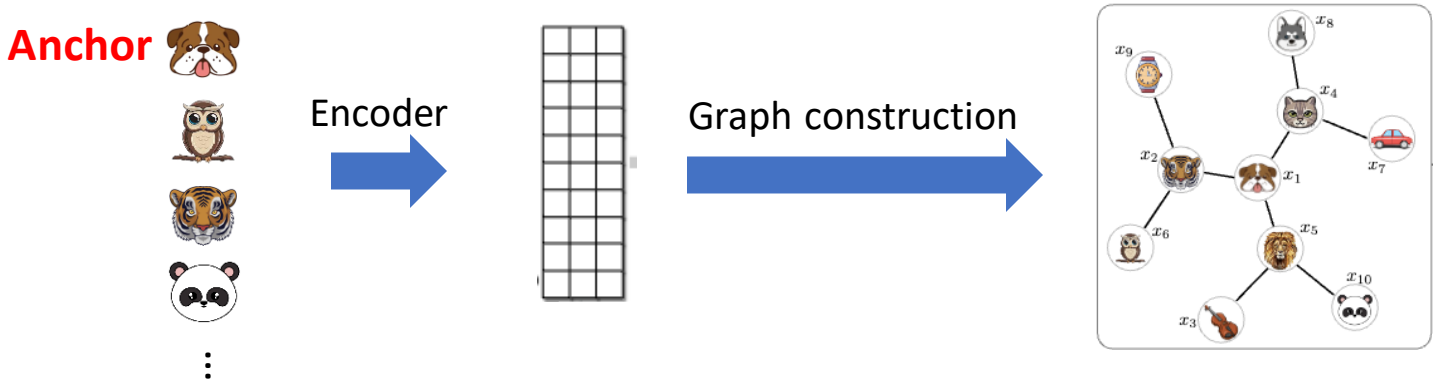
Constructed graph

Proximity Graph (2)

- Graph construction
  - For each v_i, randomly pick M **neighbor candidates** to form a candidate set $\{v_m\}$
    - 1) Accelerate the construction progress
    - 2) It can be proved that the introduction of M can be used to modulate neighbor's similarity
  - Select the K **nearest** ones from the candidate set to form the neighbor set

$$\mathcal{N}_i = \mathrm{TopK}_{\{v_m\}}(\mathbf{e}_i \cdot \mathbf{e}_m)$$ $\mathbf{e}$: representation

  - The graph will be updated after several training steps



Proximity Graph Construction (2)

- Theoretical guaranteed property: candidate set size M **controls similarity** between center node and its neighbors
  - When $M = N$, proximity graph degenerates to kNN graph
    - i.e., each node directly connects to top-k similar nodes
  - When $M = 1$, each node will randomly connect with the other node
- Theoretical proof

Given an instance v_i with the corresponding representation e_i, assume that there are at least S instances whose inner product similarity with v_i is larger than s, i.e.,

$$\left|\{v_j \in \mathcal{V} | e_i \cdot e_j > s\}\right| \geq S.$$

Then in the proximity graph G, the similarity between v_i and its neighbors is larger than s with proximate probability at least:

$$\mathbb{P}\{e_i \cdot e_k > s, \forall v_k \in \mathcal{N}_i\} \gtrapprox (1 - p^M)^K$$

where $p = \frac{N-S}{N}$, and K is the number of neighbors.

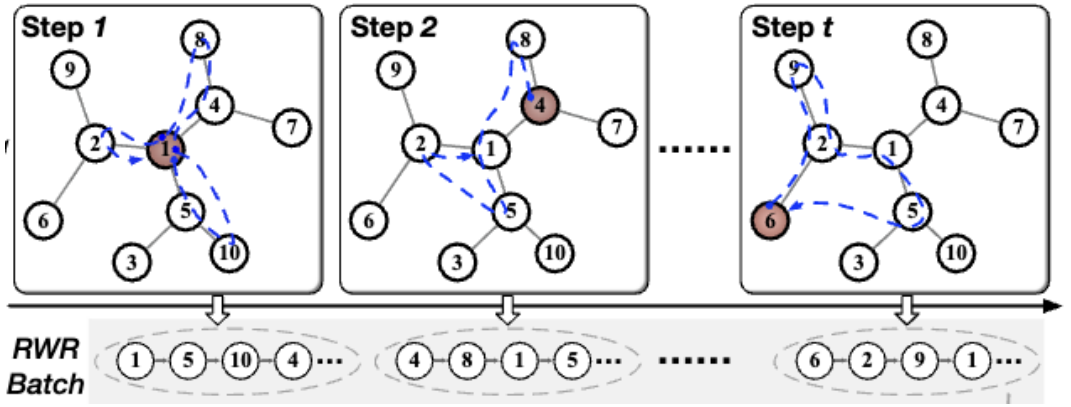
Higher M indicates a greater probability that two adjacent nodes are similar

Proximity Graph Sampling (1)

- With proximity graph, we can easily collect a batch of similar examples
  - Perform negative sampling as a walking in the proximity graph
- We apply **Random Walk with Restart (RWR)** to flexibly explore the negatives
  - It exhibits a mixture of BFS and DFS
    - **Breadth-first Sampling(BFS)** collects all the immediate neighbors
    - **Depth-first Sampling(DFS)** explores the node branch as far as possible
  - Beginning at a node, sampler iteratively teleports back to the start point with **probability α** or travels to a neighbor of the current position

Starting node

Walk



RWR started on brown node in the proximity graph at different time step (1~t)

Proximity Graph Sampling (2)

- Theoretical guaranteed property: Restart probability α can **modulate the probability** of sampling within a neighborhood
- Theoretical proof:

  For all $0 < \alpha \leq 1$ and $\mathcal{S} \subset \mathcal{V}$, the probability that a Lazy Random Walk with Restart starting from a node $u \in \mathcal{S}$ escapes $\mathcal{S}$ satisfies $\mathcal{S} \sum_{v \in (\mathcal{V} - \mathcal{S})} \mathbf{p}_u(v) \leq \frac{1-\alpha}{2\alpha} \Phi(\mathcal{S})$, where $\mathbf{p}_u$ is the stationary distribution, and $\Phi(\mathcal{S})$ is the graph conductance of $\mathcal{S}$.

  - The probability of RWR escaping from a local cluster can be bound by α
    - Higher α indicates that the walker will approximate BFS behavior and sample within a small locality
    - Lower α encourages the walker to visit the nodes which are further away from the center node.

BatchSampler (1)

- BatchSampler: Proximity Graph-based Sampler
  - Capture similarity relationships among instances by **proximity graph**
  - Perform negative sampling as a **walking** in the proximity graph
- The number of candidates M and the restart probability α are the key to **flexibly control the hardness** of a sampled batch
  - kNN Sampler: Retrieve a set of nearest neighbors to construct a batch
  - Uniform Sampler: Randomly sample a batch of instances

Proximity graph==kNN graph;
Sampler only collects immediate neighbors

RWR degenerates into DFS and chooses neighbors that are linked at random

BatchSampler (2): Empirical Criterion of M, α

500 or 1000 exhibits the best performance

Modality	Dataset	Neighbor Candidate Size M					Restart Probability α				
		500	1000	2000	4000	6000	0.1	0.3	0.5	0.7	0.2~0.05
Image	CIFAR10	**92.54**	92.49	91.83	91.72	91.43	92.41	92.26	92.12	92.06	**92.54**
	CIFAR100	67.92	**68.68**	67.05	66.19	65.55	68.31	67.98	68.20	68.00	**68.68**
	STL10	84.16	**84.38**	82.80	81.91	80.92	83.01	80.69	83.93	82.56	**84.38**
	ImageNet-100	59.6	**60.8**	60.1	59.1	58.4	60.8	59.6	58.1	57.7	**60.8**
Text	Wikipedia	71.36	**76.69**	76.09	75.76	75.11	71.74	72.13	72.41	**76.69**	–
Graph	IMDB-B	**71.90±.46**	71.28±.51	71.13±.48	70.86±.56	70.68±.59	71.26±.29	71.00±.46	71.06±.21	70.78±.58	**71.90±.46**
	IMDB-M	**48.93±.28**	48.68±.35	48.88±.94	48.71±.93	48.12±.75	48.48±1.07	48.27±.67	48.72±.41	48.78±.60	**48.93±.28**
	COLLAB	70.47±.33	**71.48±.28**	70.93±.50	70.46±.28	70.24±.56	70.36±.28	70.63±.53	70.63±.54	70.31±.37	**71.48±.28**
	REDDIT-B	**90.88±.16**	89.45±.99	90.64±.38	89.92±.75	90.37±.89	90.22±.38	89.51±.61	90.44±.48	90.28±.89	**90.88±.16**

linearly decay α from 0.2 to 0.05 as the training epoch increases

α should be high for pretrained language model

- The suggested M would be 500 for the small-scale dataset, and 1000 for larger dataset
- The suggested α should be relatively high (0.7) for the pre-trained language model.
- Besides, dynamic decay α, e.g., 0.2 to 0.05, is the best strategy for the other algorithms.

BatchSampler (3)

- BatchSampler is a modality-independent method which can generally improve the pretrained model

ImageNet (Vision modality)

Method	100 ep	400 ep	800 ep
SimCLR	64.0	68.1	68.7
w/ BatchSampler	**64.7**	**68.6**	**69.2**
MoCo v3	68.9	73.3	73.8
w/ BatchSampler	**69.5**	**73.7**	**74.2**

* Only conduct experiments on BatchSampler.

7 textual similarity datasets (Language modality)

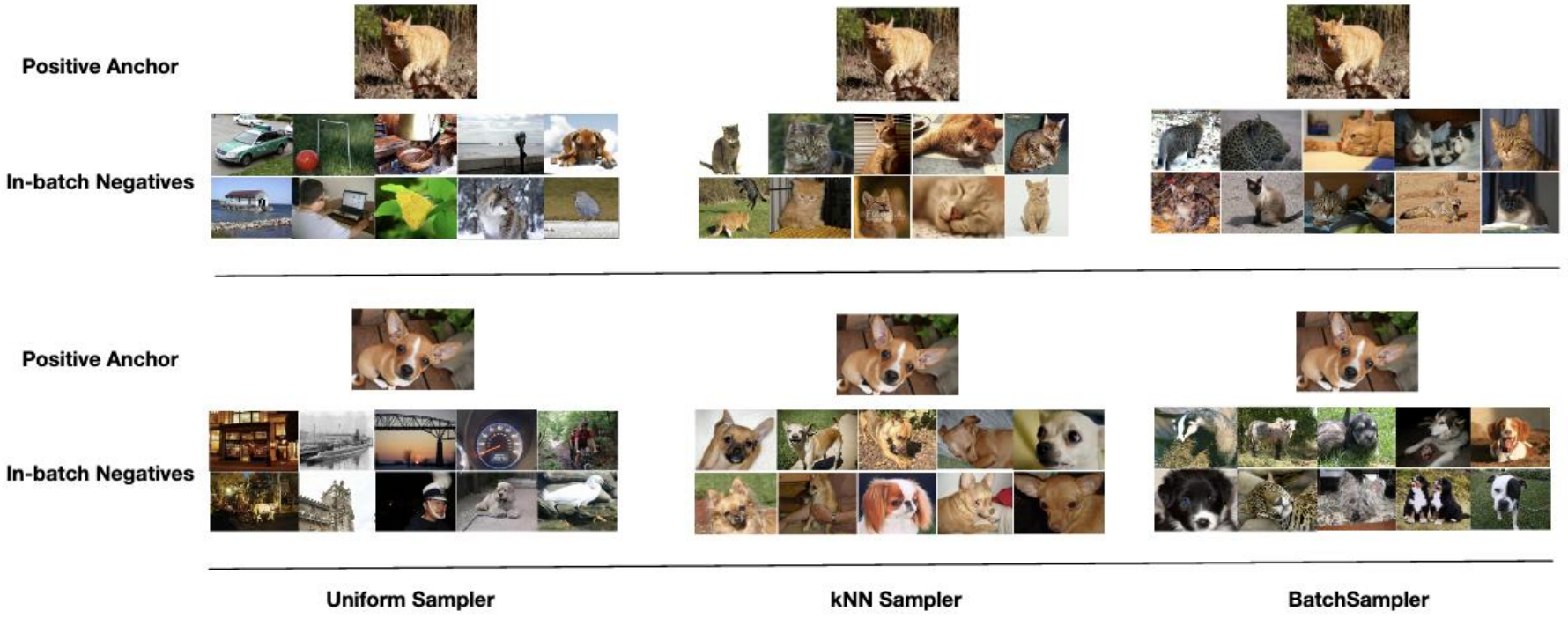
Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
SimCSE-BERT$_{base}$	68.62	80.89	73.74	80.88	77.66	**77.79**	**69.64**	75.60
w/ kNN Sampler	63.62	74.86	69.79	79.17	76.24	74.73	67.74	72.31
w/ BatchSampler	**72.37**	**82.08**	**75.24**	**83.10**	**78.43**	77.54	68.05	**76.69**

Molecular classification datasets (Graph modality)

Method	IMDB-B	IMDB-M	COLLAB	REDDIT-B	PROTEINS	MUTAG	NCI1
GraphCL	70.90±0.53	48.48±0.38	70.62±0.23	90.54±0.25	74.39±0.45	86.80±1.34	77.87±0.41
w/ kNN Sampler	70.72±0.35	47.97±0.97	70.59±0.14	90.21±.74	74.17±0.41	86.46±0.82	77.27±0.37
w/ BatchSampler	**71.90±0.46**	**48.93±0.28**	**71.48±0.28**	**90.88±0.16**	**75.04±0.67**	**87.78±0.93**	**78.93±0.38**
MVGRL	74.20±0.70	51.20±0.50	-	84.50±0.60	-	89.70±1.10	-
w/ kNN Sampler	73.30±0.34	50.70±0.36	-	82.70±0.67	-	85.08±0.66	-
w/ BatchSampler	**76.70±0.35**	**52.40±0.39**	-	**87.47±0.79**	-	**91.13±0.81**	-

BatchSampler (4): Case Study

- Case study of the negatives sampled on ImageNet



Fails to sample hard negatives

Sample lots of negatives with the same label (false negatives)

Sample hard and true negatives

Summary

- Contrastive learning
  - Bringing **semantically similar** instances closer while pushing **dissimilar** instances
  - Apply data augmentation to generate positive samples
  - Mini-batch sampling is equivalent to the negative sampling
- Improved pretrained model with negative sampling
  - Benefit of hard negative sampling
  - BatchSampler: Proximity Graph-based Sampler
    - Capture similarity relationships among instances by **proximity graph**
    - Perform negative sampling as a **walking** in the proximity graph

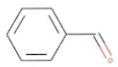
Strategy for Improving Pre-trained Model on Graph

- **Improved pretrained model with negative sampling**
  - **Contrastive learning**
  - **KDD2023-BatchSampler: Sampling Mini-Batches for Contrastive Learning in Vision, Language, and Graphs**

- **Improved pretrained model with dataset grouping**
  - **Auxiliary molecule dataset**
  - **NeurIPS2023-Learning to group auxiliary dataset for molecule**

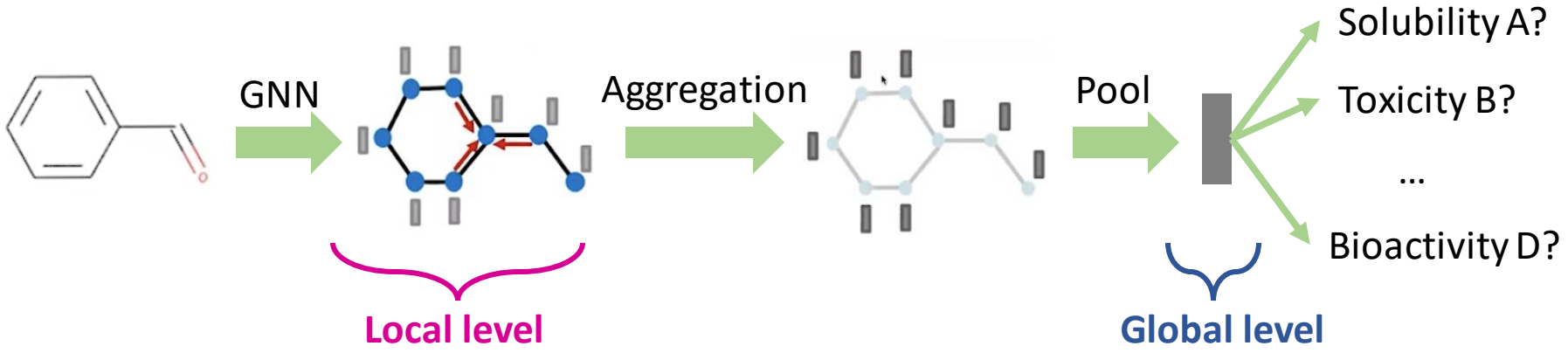
Recall: GNN for Molecule Prediction (1)

- Molecule property prediction is a fundamental task in biomedication
  - Evaluate the toxicity of new clinical drugs
  - Characterize the binding results for the inhibitors of human β-secretase
  - Predict the thermodynamic property of organic molecules

$$\underset{\text{model}}{f}\left(\text{  }\right) = \begin{cases} \text{Solubility A?} \\ \text{Toxicity B?} \\ \vdots \\ \text{Bioactivity D?} \end{cases}$$

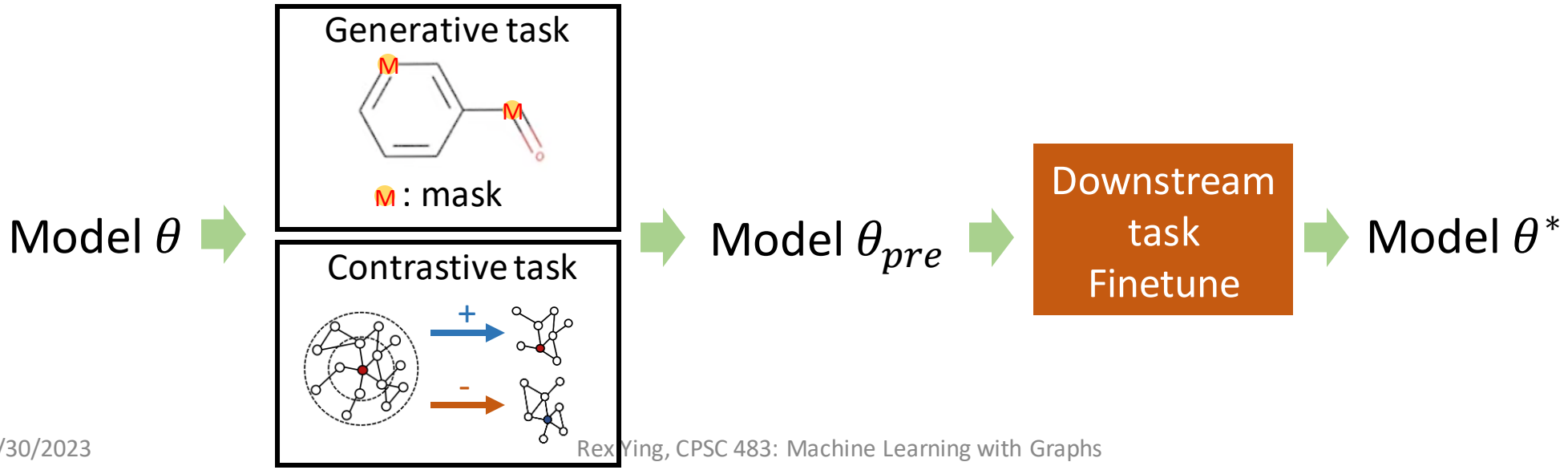
Recall: GNN for Molecule Prediction (2)

- We can apply GNN to learn the molecular graph representation
  - Molecule can naturally be modeled as graph
    - Node: atom
    - Edge: interaction between two atoms
  - **Local level**: Iteratively perform neighbor aggregation to obtain **node** representation
  - **Global level**: Use pooling operation to obtain **graph** representation



Recall: GNN for Molecule Prediction (3)

- Pretraining a GNN on molecular graph can bring better performance
  - Generative task
    - Node-level pretraining: attribute masking and context prediction
    - Graph-level pretraining: attribute prediction
  - Contrastive task
    - Apply data augmentation and InfoNCE loss



Challenge (1)

- However, labeling molecules requires expensive real-world clinical trials and expert knowledge
  - ClinTox: predict clinical toxicity of new drugs
    - 1,000 labeled instances
  - FreeSolv: estimate the hydration free energy of molecule in water
    - 400 labeled instances
  - BACE: Inhibitors of human β-secretase
    - 1,513 labeled instances
  - BBBP: Predict the permeability properties on the membrane
    - 2,039 labeled instances

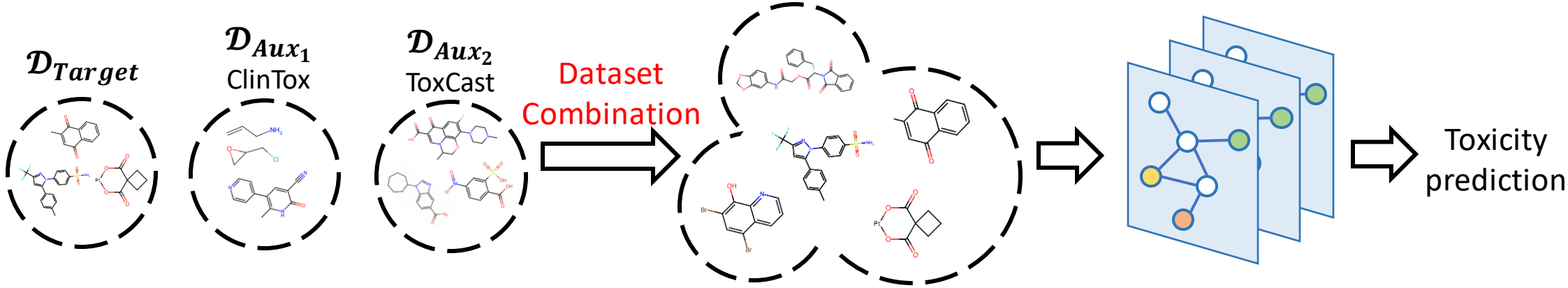
Challenge (2)

[1] Wang, Yuyang, et al. "Molecular contrastive learning of representations via graph neural networks." *Nature Machine Intelligence* 4.3 (2022): 279-287

- Even with pretraining, model struggle to effectively generalize on **small molecule datasets**
  - E.g., SOTA pretrained model MolCLR [1] only achieves
    - 1.2% improvement on BBBP (2,039 instances)
    - 1.5% improvement on BACE (1,513 instances)
    - 0.6% improvement on ClinTox (1,478 instances)
  - Compared with non-pretrained models

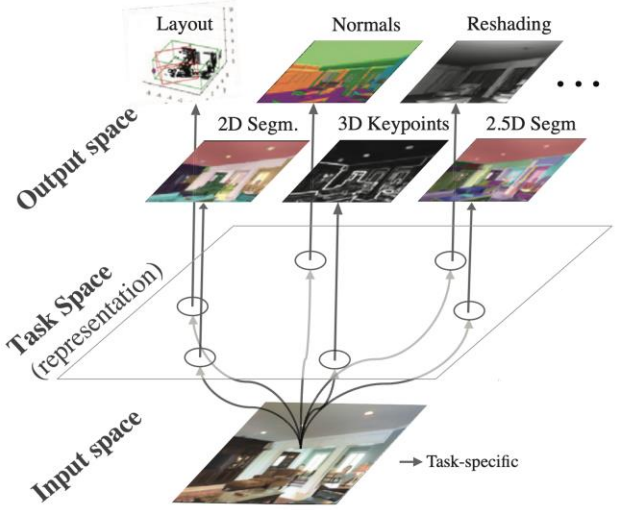
Auxiliary Molecule Dataset (1)

- One common strategy is to collaborate with **auxiliary datasets**
  - $\mathcal{D}_{Target}$: Target dataset, $\mathcal{D}_{Aux}$: Auxiliary dataset
  - Such as augmenting the current drug data from other known drugs
    - ToxCast: A toxicology data based on in vitro high-throughput screening
    - ClinTox: A set of drugs approved by FDA
  - Significantly improve performance by introducing out-of-distribution knowledge

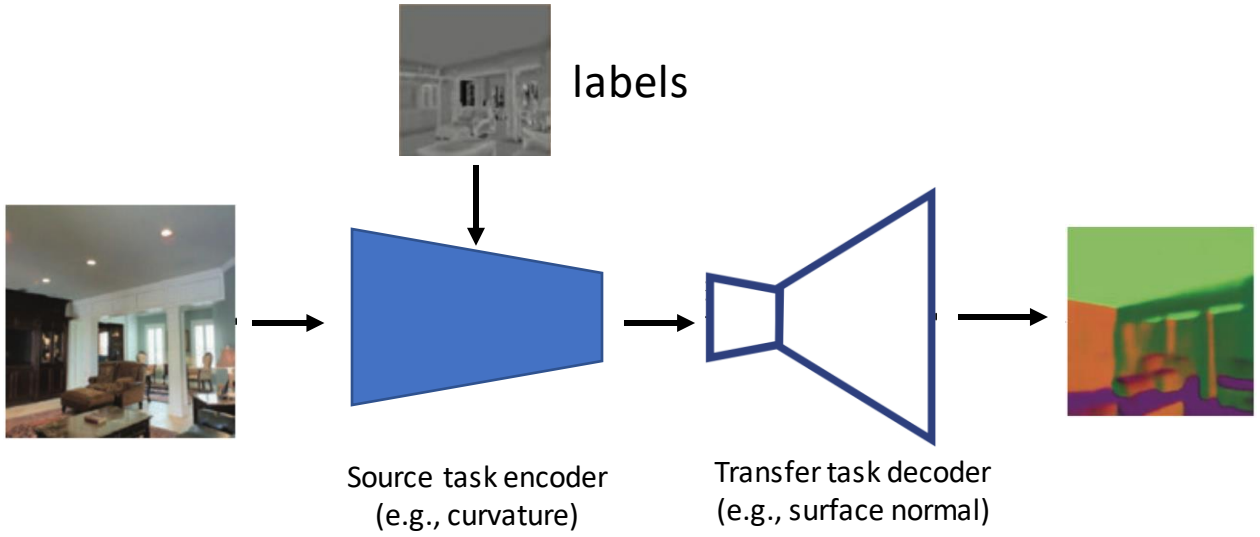


Auxiliary Molecule Dataset (2)

- Such a strategy can also be demonstrated by other domains
  - Incorporating additional supervision signals to enhance the performance



Multi-task learning: Multiple annotations are applied to each training instance



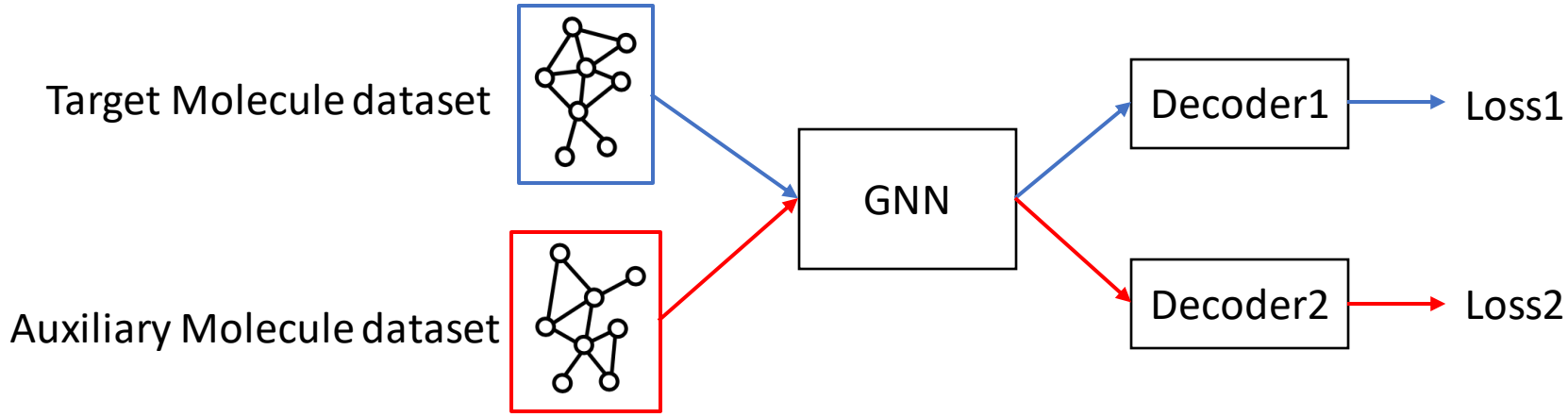
Transfer learning: Model will first be optimized on another dataset/task

Auxiliary Molecule Dataset (3)

- However, having more data does not always guarantee improvements
  - **Negative transfer** can occur when the knowledge in the target dataset differs or contradicts that of the auxiliary molecule datasets
- Let's do an empirical study to verify
  - Totally 15 molecule datasets with 11 small datasets as target datasets
    - Domain: Medication, quantum chemistry, chemical analysis
  - Pair each target dataset with every other dataset and measure the relative improvement achieved by the combination

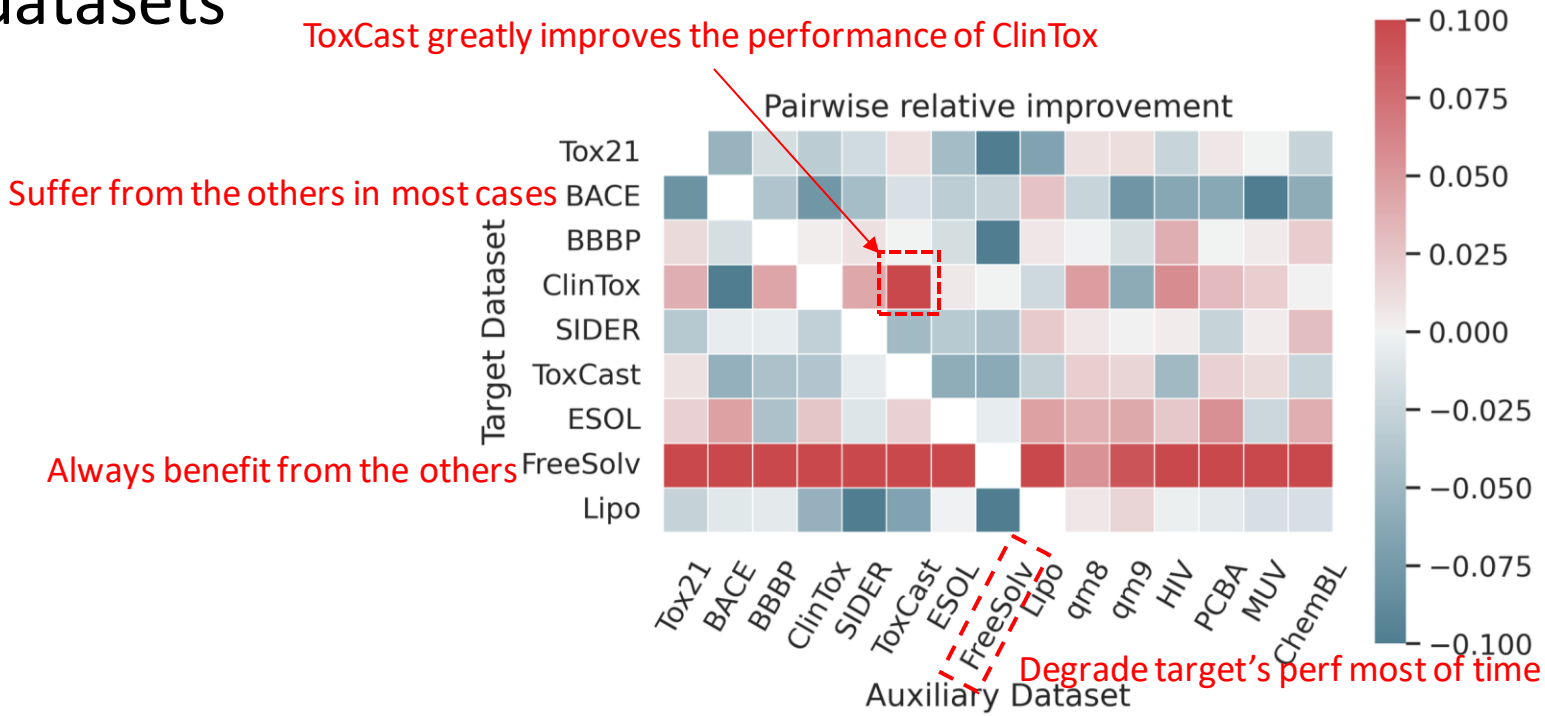
Auxiliary Molecule Dataset (4)

- Train the model using the combined datasets and assess its performance on the target datasets
  - The overall training loss is calculated as the unweighted mean of the losses for all included tasks
  - Backbone: GIN



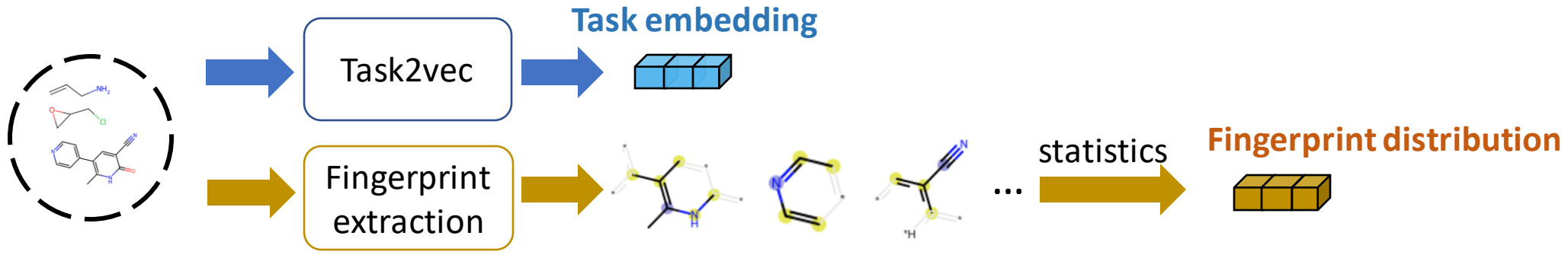
Auxiliary Molecule Dataset (5)

- Relative improvement between pairwise performance and single-train performance: (a-b)/b
- These findings highlight the presence of underlying affinity between different datasets



Understanding Relationship between Datasets (1)

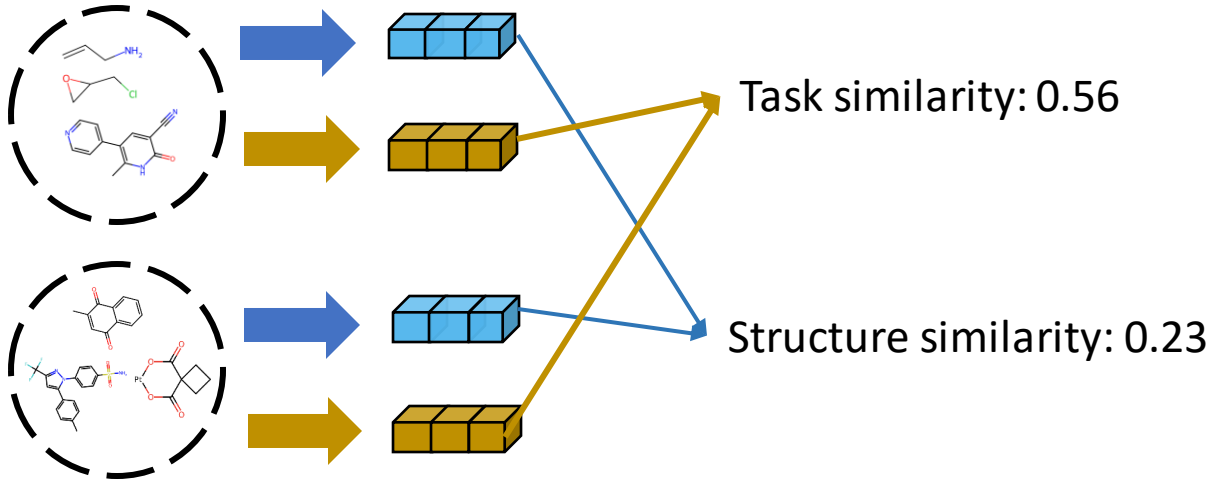
- How can we measure the affinity of auxiliary datasets to a target dataset?
- We analyze the relationship between molecule datasets by dividing them into two dimensions
  - Structural characteristics: **Fingerprint features**
  - Associated predictive task: **Task embedding** extracted by Task2vec[1]
- We can extract each dataset's fingerprint distribution and task embedding



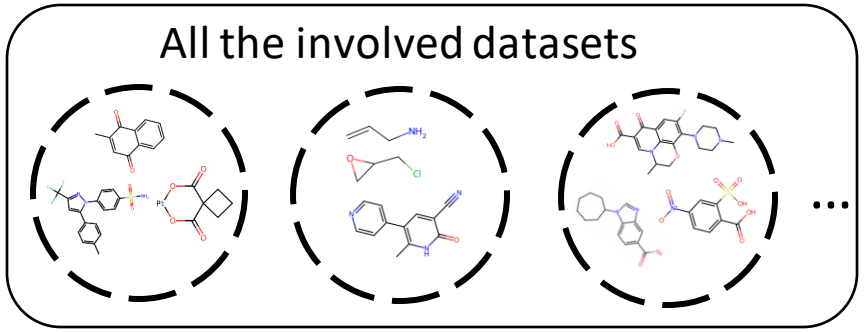
[1] Task2vec: Task embedding for meta-learning

Understanding Relationship between Datasets (2)

- Asymmetric KL divergence is used as similarity metric
  - Considering the asymmetric nature of the impact between molecule datasets
- For each pair of dataset, we can measure the similarity in terms of structure and task



Understanding Relationship between Datasets (3)

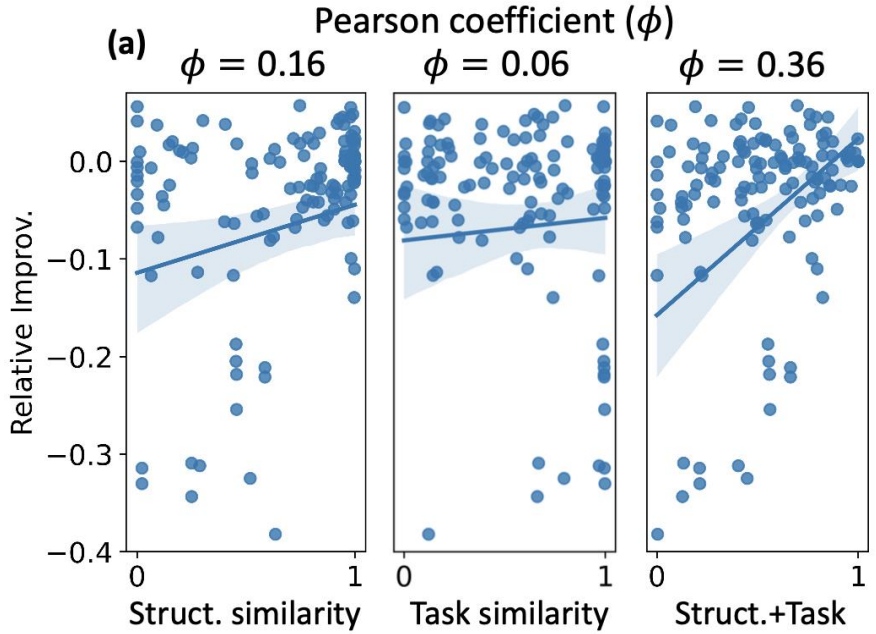


Relative perf. of all pairs ($n \times (n-1)$ cases)

Similarity measure of all pairs ($n \times (n-1)$ cases)

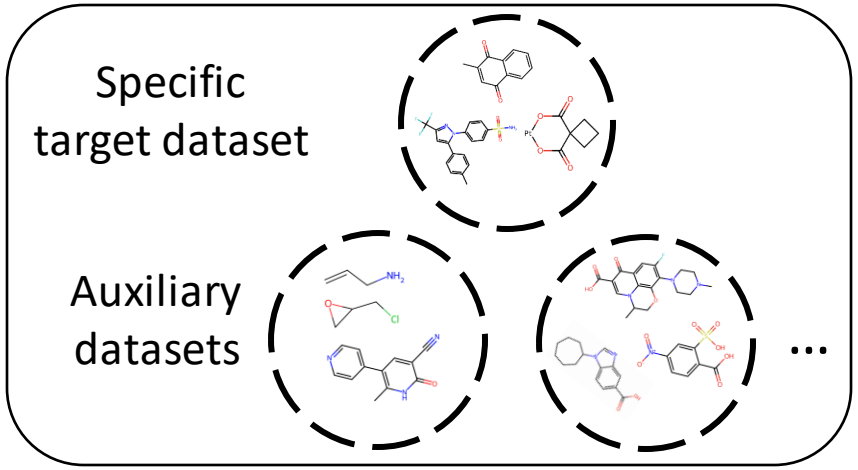
Pearson Correlation ϕ

- Similarity measure
  - (1) Structure similarity
  - (2) Task similarity
  - (3) (Structure similarity + Task similarity) / 2
- **Finding: Combination of task and structure leads to stronger correlation**



Understanding Relationship between Datasets (4)

- Analysis on individual dataset
  - Compute **structural** and **task** similarity between each target dataset and the other 14 datasets individually



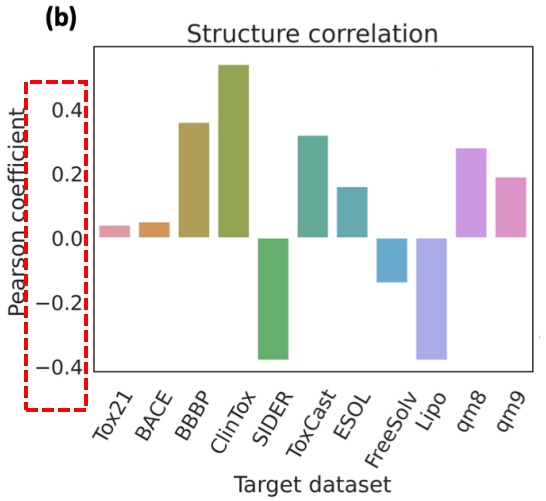
Relative perf. of all pairs between target dataset and auxiliary dataset ($(n-1)$ cases)

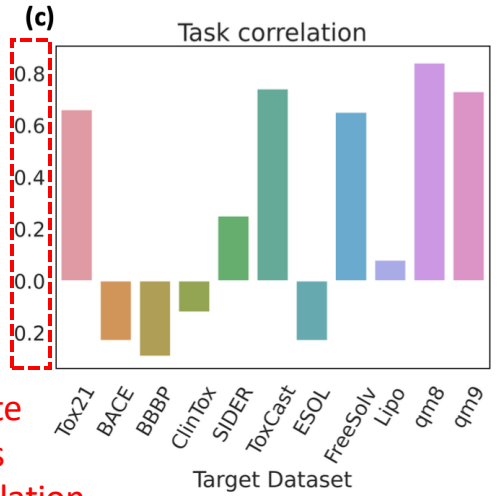
Similarity measure of all pairs between target dataset and auxiliary dataset ($(n-1)$ cases)

Pearson Correlation ϕ

Understanding Relationship between Datasets (3)

- Analysis on individual dataset
  - Compute **structural** and **task** similarity between each target dataset and the other 14 datasets individually
- **Finding: Both similar and dissimilar structures and tasks can benefit target dataset**
  - Although most of cases show a positive correlation, there are also negatively related cases
  - Additional information required from the other sources of data varies across different target datasets



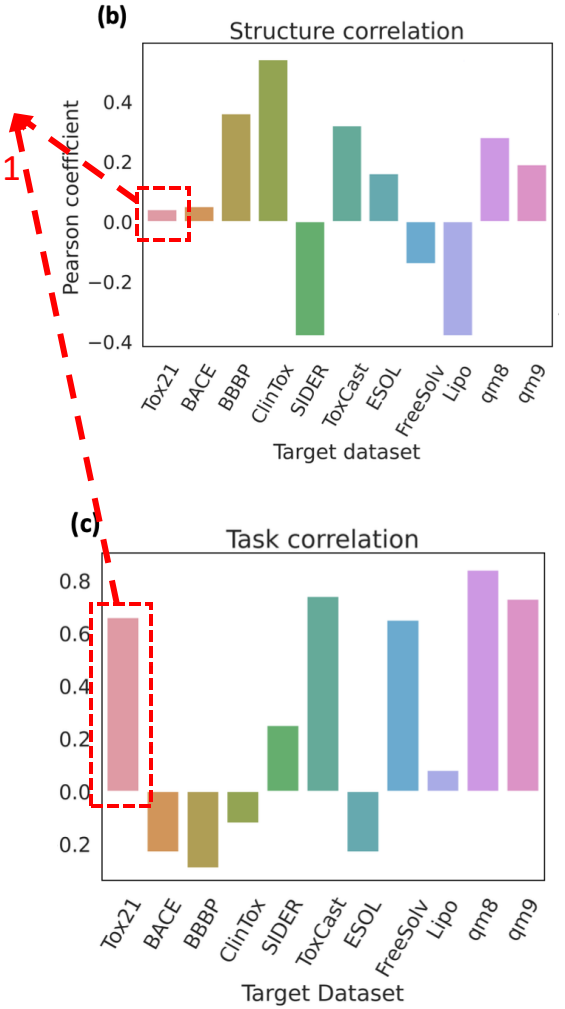


Higher absolute score indicates stronger correlation

Understanding Relationship between Datasets (4)

- **Finding: Structure and task are compensatory**
  - A dataset may present a low **structural** correlation but a high correlation in **task** similarity
    - E.g., Tox21
  - These two sources of information contribute to the performance gain in a complementary manner

Week structural correlation but strong task correlation in Tox21



Understanding Relationship between Datasets (5)

- Takeaway message
  - We demonstrate the presence of the underlying affinity between different molecule datasets
  - 1) Combine graph structure and task similarity can serve as a more reliable indicator
  - 2) Both similar and dissimilar structure and task can potentially benefit the target dataset
  - **Given a target dataset, how to identify a good subset of datasets which can maximize the performance?**

Problem Formulation

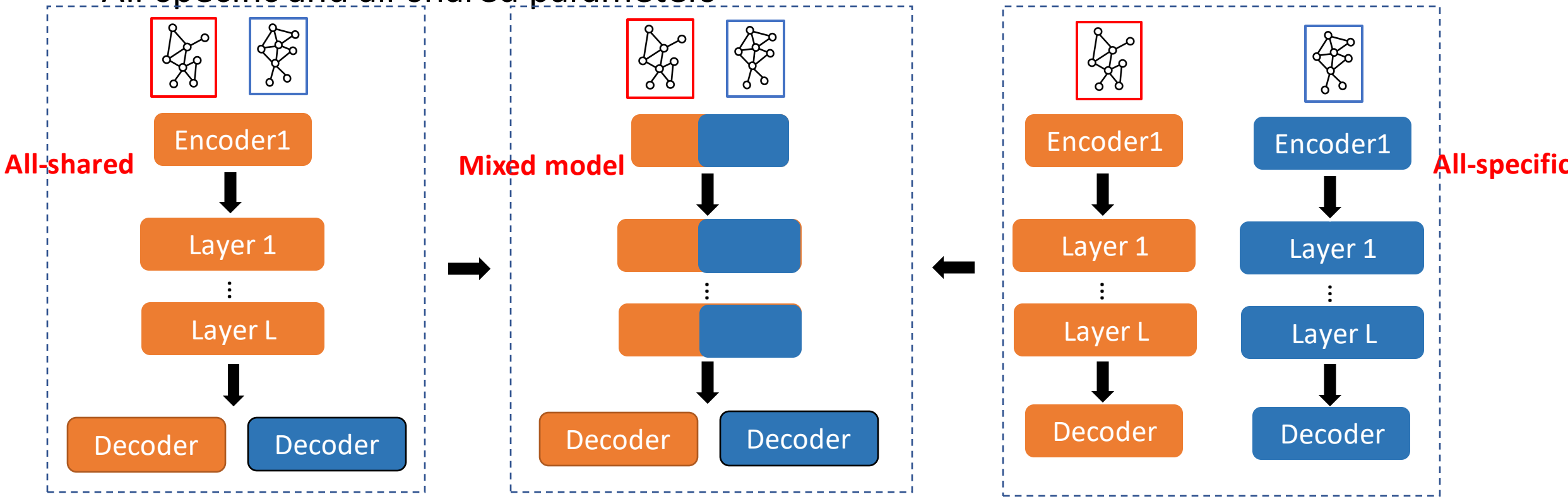
- Problem definition: Auxiliary dataset grouping
  - Given a target dataset $\mathcal{D}_T$ and a set of auxiliary datasets $\{\mathcal{D}_{Aux}\}_M$
  - Select a subset of the auxiliary datasets that can maximize the performance of $\mathcal{D}_T$ when train together
- Training pipeline with multiple datasets
  - In each training step, we sample data from each dataset with equal probability to form a mini-batch
  - Each batch includes $\mathcal{B}_T, \{\mathcal{B}_m\}_M$, so we obtain losses $l_T, \{l_m\}_M$

batch of data from m-th auxiliary dataset

  - The overall training loss is calculated as the unweighted mean of the losses for all included datasets

MolGroup: Routing Mechanism (1)

- We apply routing mechanism to quantify the affinity between datasets
- Let's consider two extreme situations of knowledge transfer
  - All-specific and all-shared parameters



MolGroup: Routing Mechanism (2)

- Intuition
  - Parameters of a positively transferred pair should be more "shared"
  - Parameters of a negatively transferred pair should be more "specific"
- We use a routing function $g(\cdot)$ to determine such a relation
  - θ_T: target dataset's parameter
  - θ_m: m-th auxiliary dataset's parameter

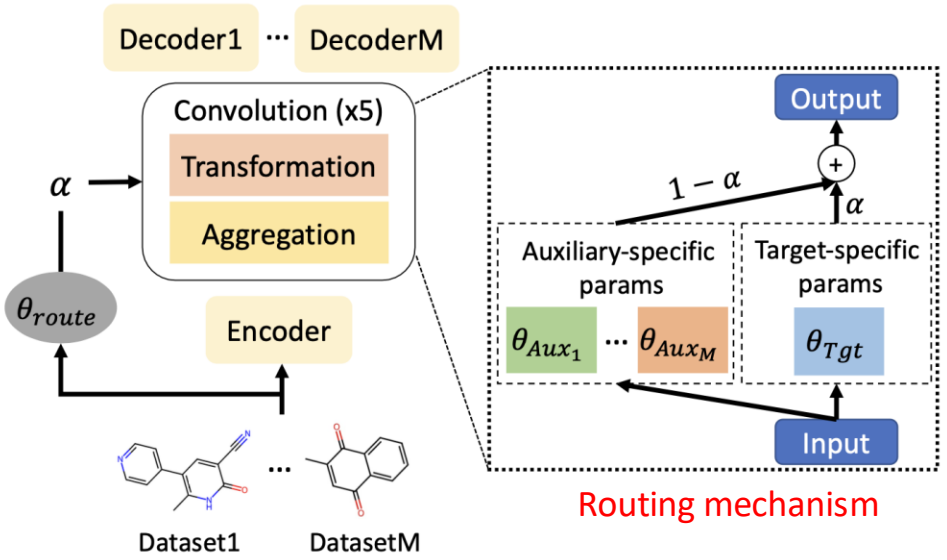
Original layer: $\mathbf{z}^{(l+1)} = f_\theta^{(l)}(\mathbf{z}^{(l)})$

$f_\theta^{(l)}$: l-th layer
$\mathbf{z}^{(l)}$: l-th input

Current layer: $\mathbf{z}^{(l+1)} = \alpha_m f_{\theta_T}^{(l)}\left(\mathbf{z}_m^{(l)}\right) + (1-\alpha_m) f_{\theta_m}^{(l)}\left(\mathbf{z}_m^{(l)}\right)$

with $\alpha_m = g_m(\mathcal{B}_T, \mathcal{B}_m)$

Input batch



Routing mechanism

MolGroup: Routing Mechanism (3)

- We assign different parameters to different datasets
  - Target dataset's parameter: θ_T, Auxiliary dataset's parameter: $\{\theta_m\}_M$
- Routing mechanism $g(\cdot)$ determines the impact of m-th auxiliary dataset on the target dataset's parameter

$$\alpha_m = g(\mathcal{B}_T, \mathcal{B}_m)$$

  - $\alpha_m = 0$ indicates all-specific architecture
  - $\alpha_m = 1$ indicates all-sharing architecture

MolGroup: Routing Mechanism (4)

- The calculation of $g(\cdot)$ is divided into two calculations
  - For **task** affinity, we assign learnable embeddings e_T^{task}, e_m^{task}
  - For **structure** affinity
    - Extract fingerprint feature for each molecule in the batch
    - Embed it using a weight matrix and apply Set2Set to obtain $e_T^{struct.}, e_m^{struct.}$
  - Finally, given an input batch of target dataset and m-th auxiliary dataset, the gating score α_m is computed as:

$$\alpha_m = g(\mathcal{B}_T, \mathcal{B}_m) = \sigma\left(\lambda e_T^{\text{task}} \cdot e_m^{\text{task}} + (1-\lambda) e_T^{\text{struct.}} \cdot e_m^{\text{struct.}}\right)$$

Hyperparameter

MolGroup: Bi-level Optimization (1)

- Original optimization framework with routing mechanism

target dataset loss $\min_{\theta_T} L_m(\mathcal{B}_T; \theta_T)$

m-th auxiliary dataset loss $\min_{\theta_T, \theta_m} L_m(\mathcal{B}_m; \theta_T, \theta_m, \alpha_m)$ (gate score)

  - Optimize the model towards minimizing the loss functions
- What is the desired objective for auxiliary dataset grouping?
  - Optimize the routing mechanism toward minimizing target dataset's loss
  - The auxiliary dataset with great benefit should be assigned a larger gating score

MolGroup: Bi-level Optimization (2)

- Desired objective
  - Optimize the routing mechanism toward minimizing target dataset's parameters
- At each learning step, we explicitly represent target parameter optimized by m-th auxiliary dataset as $\theta_T(\alpha_m)$
- Let's break the optimization into **two steps** to meet the objective
  - 1) Obtain $\theta_T(\alpha_m)$: Optimize the target parameter θ_T with $\mathcal{B}_m$
  - 2) Optimize $g(\cdot)$ based on the performance $\theta_T(\alpha_m)$ on $\mathcal{B}_T$
    - If it lowers the target dataset's loss, let's optimize it toward greater gating score

MolGroup: Bi-level Optimization (3)

- The bi-level optimization for MolGroup
  - **Lower level**: Optimize the target parameter θ_T with $\mathcal{B}_m$ and obtain $\theta_T(\alpha_m)$

$$\theta_T(\alpha_m) = \text{argmin}_{\theta_T} L_m(\mathcal{B}_m; \theta_T, \theta_m, \alpha_m)$$

  - **Higher level**: Optimize $g(\cdot)$ based on the loss of $\theta_T(\alpha_m)$ on $\mathcal{B}_T$

$$\min_{\alpha_m} L_T(\mathcal{B}_T; \theta_T(\alpha_m))$$

  - Usually, we call the gradient of higher-level objective with respect to the lower-level parameters as **meta gradient**

MolGroup: Overall Pipeline

- Main idea: quantify the affinity score as gating score produced by routing mechanism
  - Input: Target dataset, candidate auxiliary datasets, threshold β, number of iteration n
  - Output: a subset of auxiliary datasets
- Iterative filtering
  - Step1: initialize and train the model with routing function on all the datasets
    - 1) Update target parameters through routing mechanism with auxiliary dataset
    - 2) Optimize routing mechanism with updated target parameters
  - Step2: filter out the dataset that contains gate scores above β
  - Step3: go to step4 if iteration number == n or go back to step1
  - Step4: pick the auxiliary datasets with topk affinity

Benchmarking (1)

- Dataset: 15 molecule datasets
  - 11 target datasets
- Setting: Given a target dataset, select the auxiliary datasets and evaluate the performance when they train together
- Backbone: GIN and pretrained Graphormer
- Baseline
  - Search-based method
  - Grouping-based method
  - Multi-task learning method
    - The method that train on all datasets

Benchmarking (2)

- Grouping good auxiliary datasets can improve pretrained model Graphormer

Method	BBBP(↑)	ClinTox(↑)	Tox21(↑)	BACE(↑)	FreeSolv(↓)	qm8(↓)
Only-target	$66.62_{0.028}$	$56.45_{0.023}$	$74.23_{0.005}$	$75.02_{0.026}$	$3.842_{1.579}$	$0.0385_{0.001}$
Beam search(P)	$66.02_{0.015}$	$57.86_{0.068}$	$74.71_{0.004}$	$67.34_{0.039}$	$3.271_{0.479}$	$0.0553_{0.002}$
Beam search(P+S)	$67.69_{0.034}$	$57.63_{0.036}$	$74.36_{0.003}$	$69.74_{0.056}$	$3.331_{0.287}$	$0.0494_{0.000}$
TAG	$60.66_{0.014}$	$57.98_{0.028}$	$70.32_{0.006}$	$70.02_{0.076}$	$3.922_{0.748}$	$0.0637_{0.001}$
Task2vec	$68.18_{0.011}$	$47.30_{0.031}$	$68.00_{0.005}$	$74.71_{0.032}$	$3.383_{0.766}$	$0.0635_{0.001}$
MTDNN	$66.56_{0.021}$	$52.90_{0.039}$	$71.87_{0.003}$	$69.91_{0.026}$	$3.428_{0.733}$	$0.0523_{0.002}$
UA	$60.41_{0.008}$	$51.99_{0.078}$	$68.16_{0.004}$	$61.75_{0.018}$	$4.095_{0.334}$	$0.0625_{0.001}$
Gradnorm	$61.21_{0.007}$	$53.08_{0.070}$	$59.40_{0.041}$	$64.83_{0.028}$	$4.356_{0.589}$	$0.0657_{0.006}$
Pretrain-Finetune	$56.59_{0.026}$	$56.00_{0.037}$	$50.64_{0.015}$	$64.80_{0.052}$	$4.391_{0.043}$	$0.0637_{0.001}$
MolGroup	$\mathbf{68.36_{0.016}}$	$\mathbf{59.77_{0.027}}$	$\mathbf{75.66_{0.004}}$	$\mathbf{77.33_{0.015}}$	$\mathbf{3.116_{0.279}}$	$\mathbf{0.0385_{0.001}}$

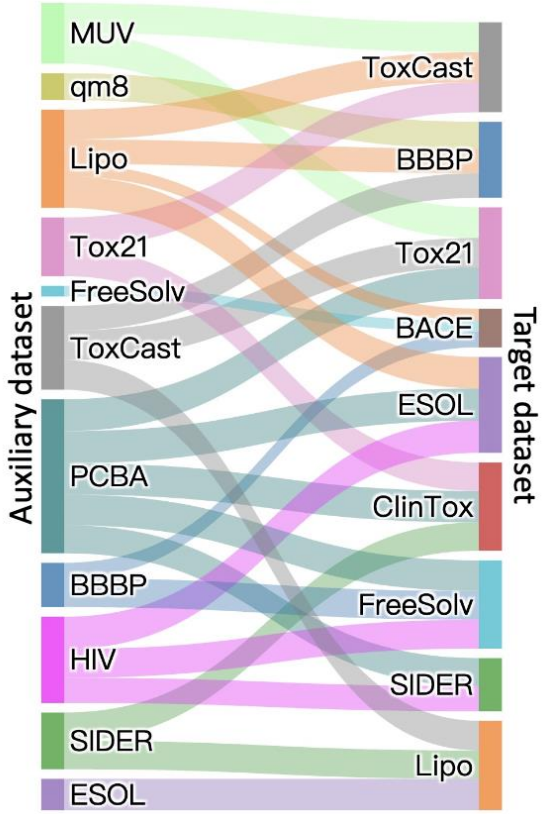
Table 1: Performance comparison of GIN on target molecule datasets, with ↑ indicating higher is better and ↓ indicating lower is better.

Method	BBBP(↑)	ClinTox(↑)	Tox21(↑)	BACE(↑)	FreeSolv(↓)	qm8(↓)
Only-target	$66.40_{0.019}$	$77.59_{0.028}$	$75.97_{0.009}$	$78.91_{0.023}$	$2.004_{0.088}$	$0.0372_{0.002}$
Beam search(P)	$67.53_{0.010}$	$76.77_{0.117}$	$76.61_{0.008}$	$81.58_{0.038}$	$2.129_{0.215}$	$0.0473_{0.001}$
Beam search(P+S)	$67.15_{0.030}$	$77.70_{0.076}$	$76.83_{0.007}$	$80.68_{0.024}$	$2.312_{0.191}$	$0.0458_{0.001}$
TAG	$69.62_{0.008}$	$78.08_{0.036}$	$76.57_{0.006}$	$80.10_{0.016}$	$2.038_{0.178}$	$0.0537_{0.001}$
Task2vec	$66.93_{0.022}$	$72.92_{0.049}$	$75.15_{0.007}$	$80.84_{0.017}$	$1.941_{0.123}$	$0.0530_{0.001}$
MTDNN	$66.71_{0.039}$	$71.97_{0.078}$	$76.84_{0.008}$	$79.79_{0.001}$	$2.173_{0.321}$	$0.0512_{0.001}$
UW	$65.37_{0.000}$	$80.52_{0.027}$	$72.16_{0.009}$	$75.64_{0.000}$	$2.405_{0.415}$	$0.0569_{0.000}$
Gradnorm	$60.40_{0.045}$	$43.46_{0.083}$	$55.06_{0.002}$	$48.00_{0.005}$	$2.552_{0.595}$	$0.1694_{0.014}$
MolGroup	$\mathbf{69.66_{0.009}}$	$\mathbf{81.21_{0.040}}$	$\mathbf{77.34_{0.006}}$	$\mathbf{82.94_{0.017}}$	$\mathbf{1.879_{0.032}}$	$\mathbf{0.0372_{0.002}}$

Table 2: Performance comparison using Graphormer on target molecule datasets.

Grouping Results

- Each edge from auxiliary dataset to target dataset represents a selection
- Auxiliary dataset with top-3 affinity scores
  - PCBA is the most “famous” one
  - Tox21 can benefit ClinTox and ToxCast
  - Some datasets belong to distinct domains but still can benefit the other dataset
    - Qm8 for BBBP
    - ESOL for Lipo



Takeaway Messages

- PCBA is an effective booster
  - It can boost performance of most of the small molecule datasets

	BBBP(↑)	ClinTox(↑)	ToxCast(↑)	Tox21(↑)	ESOL(↓)	FreeSolv(↓)	Lipo(↓)
Only-target	$66.62_{0.028}$	$56.45_{0.023}$	$60.69_{0.010}$	$74.23_{0.005}$	$1.563_{0.040}$	$3.842_{1.579}$	$0.8063_{0.015}$
+PCBA	$67.11_{0.023}$	$57.77_{0.028}$	$62.05_{0.007}$	$74.81_{0.006}$	$1.463_{0.020}$	$3.563_{0.989}$	$0.8021_{0.009}$

- Grouping more high-affinity datasets improves performance

Combinations	ClinTox	Tox21	FreeSolv	BBBP	BACE	ToxCast	ESOL	Lipo
Only Target	$56.45_{0.023}$	$74.23_{0.005}$	$3.842_{1.579}$	$66.62_{0.028}$	$75.02_{0.026}$	$60.69_{0.010}$	$1.563_{0.040}$	$0.8063_{0.015}$
+ Top{1}	$57.77_{0.028}$	$74.81_{0.006}$	$3.563_{0.989}$	$67.36_{0.023}$	$71.27_{0.045}$	$62.66_{0.002}$	$1.502_{0.043}$	$0.8096_{0.014}$
+ Top{1,2}	$57.48_{0.032}$	$75.22_{0.003}$	$3.462_{0.970}$	$\mathbf{68.64_{0.012}}$	$71.13_{0.019}$	$63.18_{0.006}$	$1.524_{0.077}$	$0.8078_{0.011}$
+Top{1,2,3}	$\mathbf{59.77_{0.027}}$	$\mathbf{75.66_{0.004}}$	$\mathbf{3.116_{0.279}}$	$68.36_{0.016}$	$\mathbf{77.33_{0.015}}$	$\mathbf{63.91_{0.005}}$	$\mathbf{1.402_{0.010}}$	$\mathbf{0.7996_{0.005}}$

Summary

- Given a target dataset, how to identify a good subset of auxiliary datasets which can maximize the performance?
- Preliminary analysis
  - Investigate the relationship between molecule datasets in terms of structure and task similarity
- MolGroup, a routing-based grouping method
  - Learn to route between auxiliary-specific or target-specific parameters
  - Routing mechanism is optimized toward maximizing the target dataset's performance