上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Learning with Non-IID Data from Physics Principles

Qitian Wu

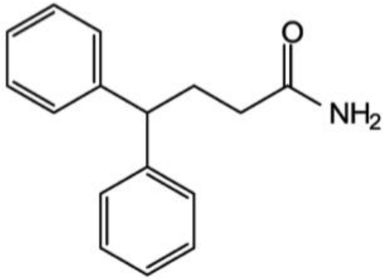
https://qitianwu.github.io/

饮 水 思 源 • 爱 国 荣 校

Data with Observed Geometry (Graphs)

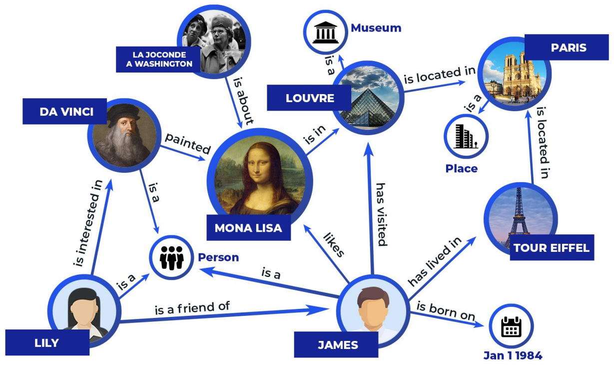
- ❑ **Graph-structured data are ubiquitous in various domains**

molecular

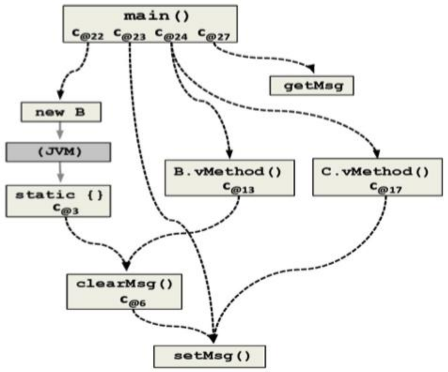


social network

knowledge graph



code



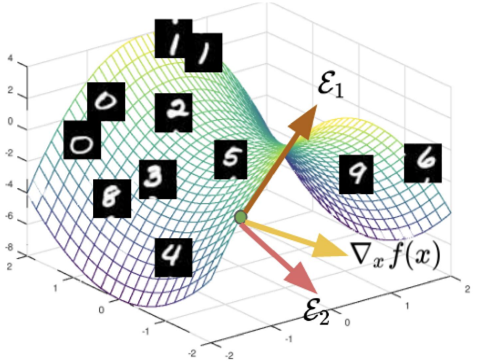
- ❑ **How to leverage the relational information of inter-dependent data?**

Challenge: 1) Arbitrary size and geometric symmetry

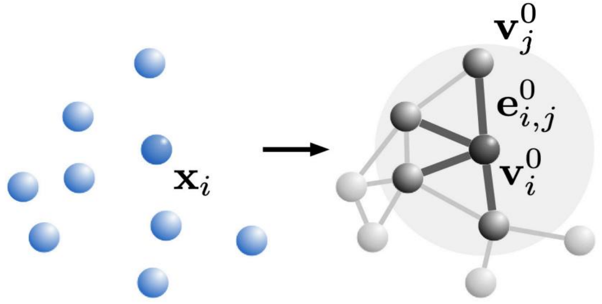
2) Complex topological structure

Data with Unobserved Geometry

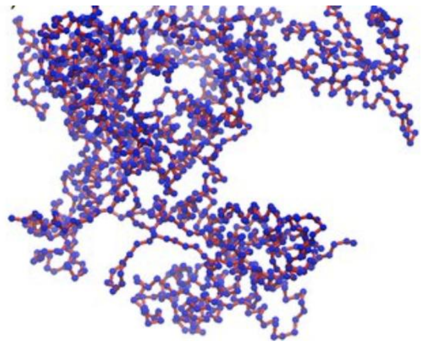
- Real-world data generation involves hidden interactions



Observed data lies on low-dimensional manifold [Sebastian et al., 2021]



Physical interactions affect data generation yet are not observed [Alvaro et al., 2020]



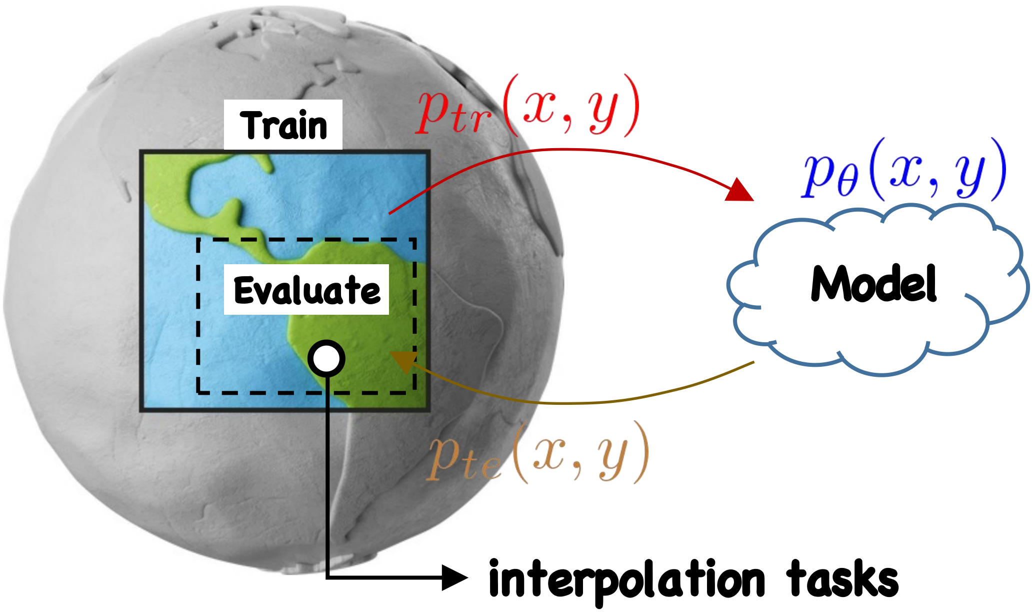
Complex hidden structures in scientific applications [Xu et al., 2020]

- How to learn and leverage latent structures from observed data?

Challenge: 1) Combinatorial searching space

2) Scalability for large-scale systems

Learning under Closed-World Assumptions

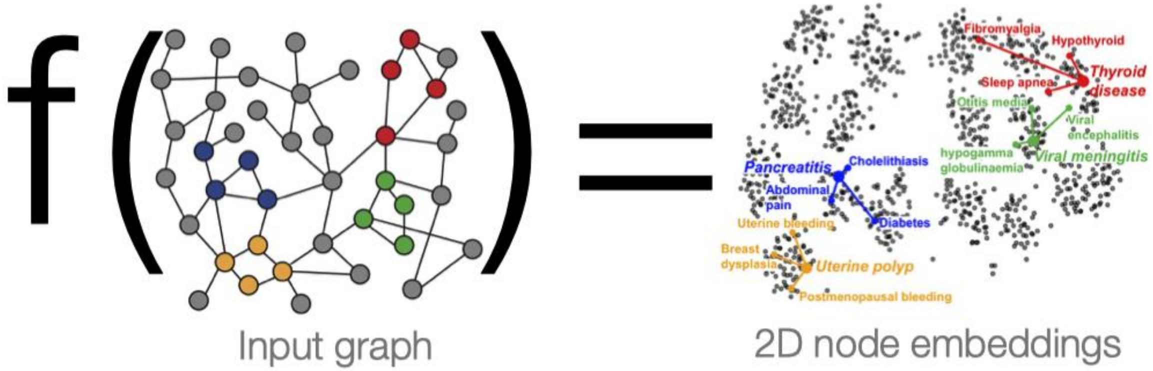


$$\overbrace{\mathcal{D}(p_\theta(x,y), p_{te}(x,y))}^{\text{model performance}} \leq$$

$$\underbrace{\mathcal{D}_1(p_\theta(x,y), p_{tr}(x,y))}_{\text{fitting error}} + \underbrace{\mathcal{D}_2(p_{tr}(x,y), p_{te}(x,y))}_{\text{generalization gap}}$$

model expressivity matters!

negligibly small

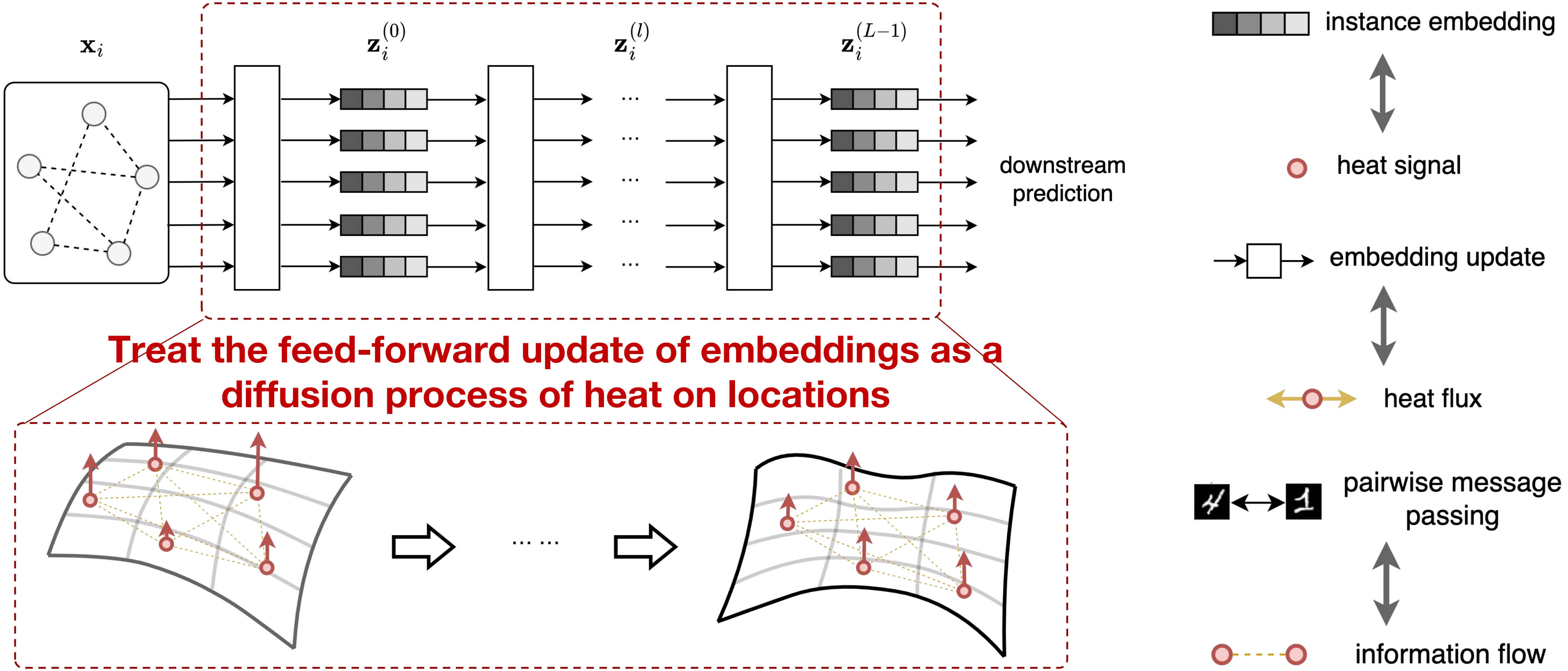


Open research question:

Q1: What is the underlying mechanism of existing models (e.g., GNNs) ?

Q2: Is there any principled guideline for designing new models?

GNN Feed-forward as Diffusion Process

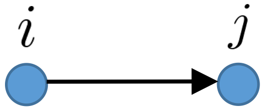


Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

General Formulation of Diffusion Process

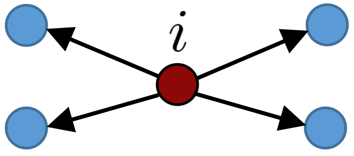
The diffusion process of N particles driven by initial states and pairwise interactions:

$$\frac{\partial \mathbf{Z}(t)}{\partial t} = \nabla^* \left(\mathbf{S}(\mathbf{Z}(t), t) \odot \nabla \mathbf{Z}(t) \right), \quad \text{s. t.} \quad \mathbf{Z}(0) = [\mathbf{x}_i]_{i=1}^N, \quad t \geq 0$$



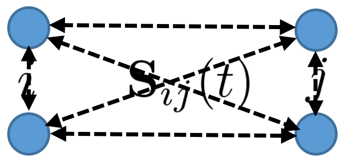
gradient

$$(\nabla \mathbf{Z}(t))_{ij} = \mathbf{z}_j(t) - \mathbf{z}_i(t)$$



divergence

$$(\nabla^*)_i = \sum_{j=1}^N \mathbf{S}_{ij}(\mathbf{Z}(t), t) \, (\nabla \mathbf{Z}(t))_{ij}$$



diffusivity function

$$\mathbf{S}(\mathbf{Z}(t), t) : \mathbb{R}^{N \times d} \times [0, \infty) \rightarrow [0, 1]^{N \times N}$$

Diffusion over discrete space composed of N instances with latent structures:

$$\frac{\partial \mathbf{z}_i(t)}{\partial t} = \sum_{j=1}^N \mathbf{S}_{ij}(\mathbf{Z}(t), t)(\mathbf{z}_j(t) - \mathbf{z}_i(t))$$

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

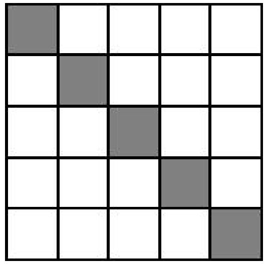
Diffusion with Latent Structures

The iterative dynamics (by explicit scheme) of diffusion induce feed-forward layers:

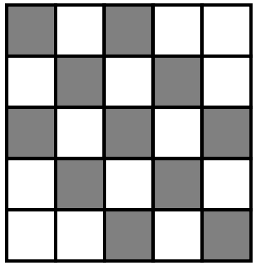
$$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)}$$

The $N \times N$ diffusivity $\mathrm{S}^{(k)}$ is a measure of the rate at which the node signals spread

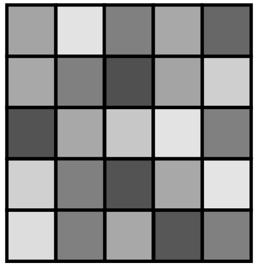
- $\mathrm{S}^{(k)}$ **is an identity matrix: message passing only through self-loops**
- $\mathrm{S}^{(k)}$ **only has non-zero values for observed edges: message passing over a graph**
- $\mathrm{S}^{(k)}$ **can have non-zero values for all entries: all-pair message passing**



MLP



GNN



Transformer

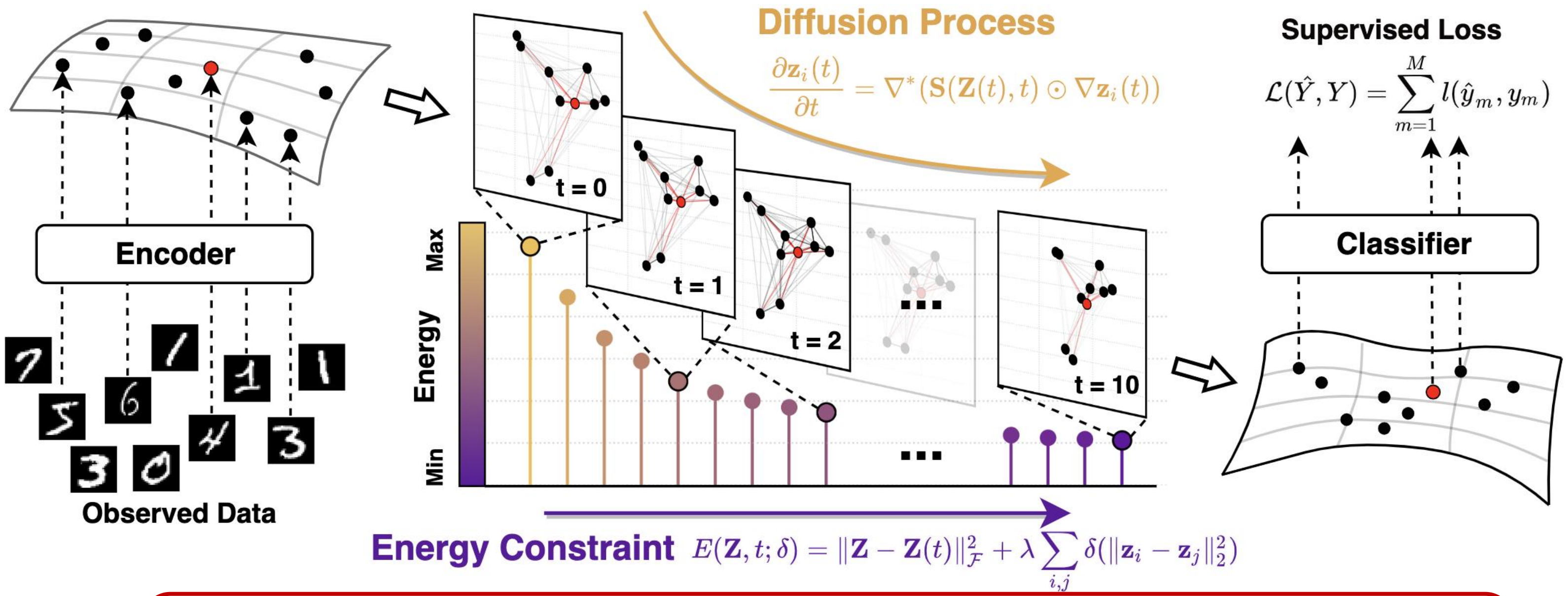
Key question: How to determine a proper diffusivity function for learning desirable node representations?

Energy-Constrained Diffusion Process

Principle 1: particle states evolution described by a diffusion process

+

Principle 2: the evolutionary directions towards descending the global energy



Key insight: treat diffusivity as latent variables whose optimality is given by descent criteria w.r.t. a principled global energy

$$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)}$$

$$\text{s. t.} \quad \mathbf{z}_i^{(0)} = \mathbf{x}_i, \quad E(\mathbf{Z}^{(k+1)}, k; \delta) \le E(\mathbf{Z}^{(k)}, k-1; \delta), \quad k \ge 1.$$

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Diffusion Equation v.s. Energy Minimization

Theorem 1 (Diffusion equation with fixed diffusivity as energy minimization dynamics)

The diffusion equation of node embeddings $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ with fixed diffusivity matrix

$$\frac{\partial \mathbf{z}_i(t)}{\partial t} = \sum_{j \in \mathcal{V}} \mathbf{S}_{ij}(\mathbf{z}_j(t) - \mathbf{z}_i(t)) + \beta \mathbf{h}_i \quad \text{where} \quad \mathbf{S} = \{s_{ij}\}_{N \times N}$$

induces dynamics implictly minimizing a global energy function

$$E(\mathbf{Z}, t) = \|\mathbf{Z} - \mathbf{Z}(t) - \eta \mathbf{H}\|_{\mathcal{F}}^2 + \lambda \sum_{i,j} s_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2$$

Graph Convolution Networks [Kipf and Welling, 2017]

$$\mathbf{z}_i^{(k+1)} = (1-\tau)\mathbf{z}_i^{(k)} + \tau \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{z}_j^{(k)} \qquad \mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

Graph Isomorphism Networks [Xu et al., 2019]

$$\mathbf{z}_i^{(k+1)} = (1+\tau)\mathbf{z}_i^{(k)} + \tau \sum_{j \in \mathcal{N}(i)} \mathbf{z}_j^{(k)} \qquad \mathbf{S} = \mathbf{A} + \mathbf{I}$$

PageRank Propagation Networks [Klicpera et al., 2019]

$$\mathbf{z}_i^{(k+1)} = (1-\tau)\mathbf{z}_i^{(k)} + \tau \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{z}_j^{(k)} + \tau \beta \mathbf{z}^{(0)} \qquad \mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Closed-Form Solutions for Diffusion Dynamics

Theorem 2 (Optimal diffusivity estimates for diffusion with time-dependent diffusivity)

For any regularized energy over $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ defined by the form

$$E(\mathbf{Z}, k; \delta) = \|\mathbf{Z} - \mathbf{Z}^{(k)}\|_{\mathcal{F}}^2 + \lambda \sum_{i,j} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$$

where $\delta : \mathbb{R}^+ \to \mathbb{R}$ is a concave, non-decreasing function, the diffusion process with diffusivity

$$\hat{\mathbf{S}}_{ij}^{(k)} = \frac{\omega_{ij}^{(k)}}{\sum_{l=1}^N \omega_{il}^{(k)}}, \quad \omega_{ij}^{(k)} = \left.\frac{\partial \delta(z^2)}{\partial z^2}\right|_{z^2 = \|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2}$$

yields a descent step on the energy, i.e., $E(\mathbf{Z}^{(k+1)}, k; \delta) \leq E(\mathbf{Z}^{(k)}, k-1; \delta)$

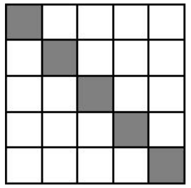
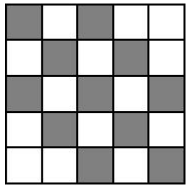
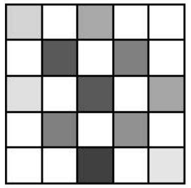
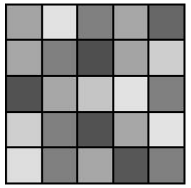
One-layer update of DIFFormer

Diffusivity Inference: $$\hat{\mathbf{S}}_{ij}^{(k)} = \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l=1}^N f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, \quad 1 \leq i, j \leq N$$

State Update: $$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^N \hat{\mathbf{S}}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^N \hat{\mathbf{S}}_{ij}^{(k)} \mathbf{z}_j^{(k)}, \quad 1 \leq i \leq N$$

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Interpretations of MLP/GNNs as Diffusion

	Energy function	Diffusivity	Illustration
MLP	$\lVert \mathbf{Z} - \mathbf{Z}^{(k)} \rVert_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} 1, & \text{if } i = j \\ 0, & otherwise \end{cases}$	
GCN	$\sum_{(i,j)\in\mathcal{E}} \lVert \mathbf{z}_i - \mathbf{z}_j \rVert_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{1}{\sqrt{d_i d_j}}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
GAT	$\sum_{(i,j)\in\mathcal{E}} \delta(\lVert \mathbf{z}_i - \mathbf{z}_j \rVert_2^2)$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{f(\lVert \mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)} \rVert_2^2)}{\sum_{l:(i,l)\in\mathcal{E}} f(\lVert \mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)} \rVert_2^2)}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
DIFFormer	$\lVert \mathbf{Z} - \mathbf{Z}^{(k)} \rVert_2^2 + \lambda \sum_{i,j} \delta(\lVert \mathbf{z}_i - \mathbf{z}_j \rVert_2^2)$	$\mathbf{S}_{ij}^{(k)} = \frac{f(\lVert \mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)} \rVert_2^2)}{\sum_{l=1}^{N} f(\lVert \mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)} \rVert_2^2)}, \quad 1 \le i,j \le N$	

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Scalable All-Pair Message Passing with O(N)

❑ Kernelized softmax message passing

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\boxed{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}}{\sum_{w=1}^{N} \exp(\mathbf{q}_u^\top \mathbf{k}_w)} \cdot \mathbf{v}_v$$

where $\mathbf{q}_u = W_Q^{(l)} \mathbf{z}_u^{(l)}, \quad \mathbf{k}_u = W_K^{(l)} \mathbf{z}_u^{(l)}, \quad \mathbf{v}_u = W_V^{(l)} \mathbf{z}_u^{(l)}$

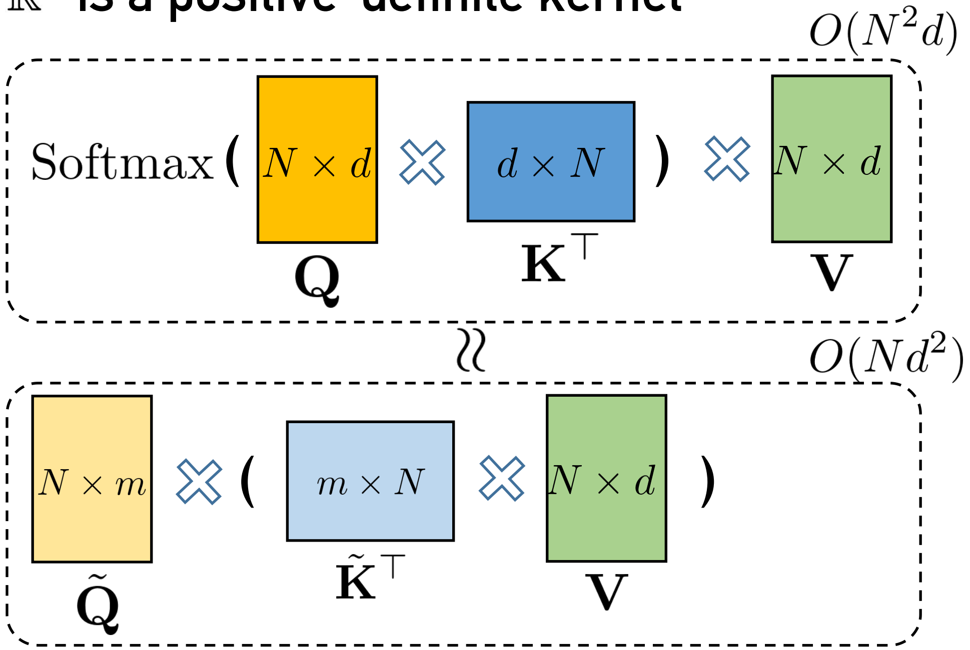
$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\boxed{\kappa(\mathbf{q}_u, \mathbf{k}_v)}}{\sum_{w=1}^{N} \kappa(\mathbf{q}_u, \mathbf{k}_w)} \cdot \mathbf{v}_v$$

where $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ **is a positive-definite kernel**

[Mercer's theorem] $\kappa(\mathbf{a}, \mathbf{b}) = \langle \Phi(\mathbf{a}), \Phi(\mathbf{b}) \rangle_{\mathcal{V}} \approx \phi(\mathbf{a})^\top \phi(\mathbf{b})$

$\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ is a random feature map

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\boxed{\phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_v)}}{\sum_{w=1}^{N} \phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_w)} \cdot \mathbf{v}_v = \boxed{\frac{\phi(\mathbf{q}_u)^\top \sum_{v=1}^{N} \phi(\mathbf{k}_v) \cdot \mathbf{v}_v^\top}{\phi(\mathbf{q}_u)^\top \sum_{w=1}^{N} \phi(\mathbf{k}_w)}}$$



two summation are shared by all nodes (independent of u)

—— only compute once

computation complexity $O(N) + N \cdot O(1) = O(N)$

Qitian Wu et al., NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification, NeurIPS 2022

Results on Large-Graph Benchmarks

Results of testing accuracy on two large-scale graph datasets

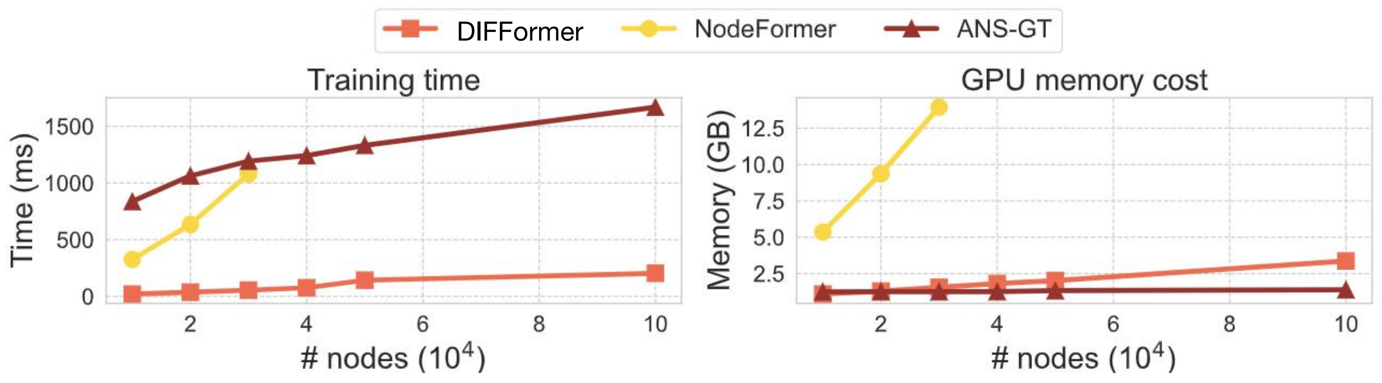
Models	Proteins	Pokec
MLP	72.41 ± 0.10	60.15 ± 0.03
LP	74.73	52.73
SGC	49.03 ± 0.93	52.03 ± 0.84
GCN	74.22 ± 0.49*	62.31 ± 1.13*
GAT	75.11 ± 1.45*	65.57 ± 0.34*
NodeFormer	**77.45 ± 1.15***	**68.32 ± 0.45***
DIFFORMER-s	**79.49 ± 0.44***	**69.24 ± 0.76***

Improve accuracy by +5.8% over GNNs

Original Transformers requires 24TB GPU memory

8000x space reduction

DIFFormer (ours) only requires 3GB GPU memory



30x inference time reduction

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Pytorch Implementation

```python
# qs: [N, H, D], ks: [L, H, D], vs: [L, H, D]

qs = qs / torch.norm(qs, p=2) # [N, H, D]
ks = ks / torch.norm(ks, p=2) # [L, H, D]
N = qs.shape[0]

# numerator
kvs = torch.einsum("lhm,lhd->hmd", ks, vs)
attn_num = torch.einsum("nhm,hmd->nhd", qs, kvs) # [N, H, D]
all_ones = torch.ones([vs.shape[0]])
vs_sum = torch.einsum("l,lhd->hd", all_ones, vs) # [H, D]
attn_num += vs_sum.unsqueeze(0).repeat(vs.shape[0], 1, 1) # [N, H, D]

# denominator
all_ones = torch.ones([ks.shape[0]])
ks_sum = torch.einsum("lhm,l->hm", ks, all_ones)
attn_den = torch.einsum("nhm,hm->nh", qs, ks_sum)  # [N, H]

# attentive aggregated results
attn_den = torch.unsqueeze(attn_den, len(attn_den.shape))  # [N, H, 1]
attn_den += torch.ones_like(attn_den) * N
z_next = attn_num / attn_den # [N, H, D]
```

github repo

tutorial

Experiment Results

Results on large node classification graphs

Method	ogbn-proteins	Amazon2m	pokec	ogbn-arxiv	ogbn-papers100M
# nodes	132,534	2,449,029	1,632,803	169,343	111,059,956
# edges	39,561,252	61,859,140	30,622,564	1,166,243	1,615,685,872
MLP	72.04 ± 0.48	63.46 ± 0.10	60.15 ± 0.03	55.50 ± 0.23	47.24 ± 0.31
GCN	72.51 ± 0.35	83.90 ± 0.10	62.31 ± 1.13	**71.74 ± 0.29**	OOM
SGC	70.31 ± 0.23	81.21 ± 0.12	52.03 ± 0.84	67.79 ± 0.27	63.29 ± 0.19
GCN-NSampler	73.51 ± 1.31	83.84 ± 0.42	63.75 ± 0.77	68.50 ± 0.23	62.04 ± 0.27
GAT-NSampler	74.63 ± 1.24	85.17 ± 0.32	62.32 ± 0.65	67.63 ± 0.23	63.47 ± 0.39
SIGN	71.24 ± 0.46	80.98 ± 0.31	68.01 ± 0.25	70.28 ± 0.25	**65.11 ± 0.14**
NodeFormer	**77.45 ± 1.15**	**87.85 ± 0.24**	**70.32 ± 0.45**	59.90 ± 0.42	-
SGFormer	**79.53 ± 0.38**	**89.09 ± 0.10**	**73.76 ± 0.24**	**72.63 ± 0.13**	**66.01 ± 0.37**

SGFormer can be trained in full-graph manner on obgn-arxiv

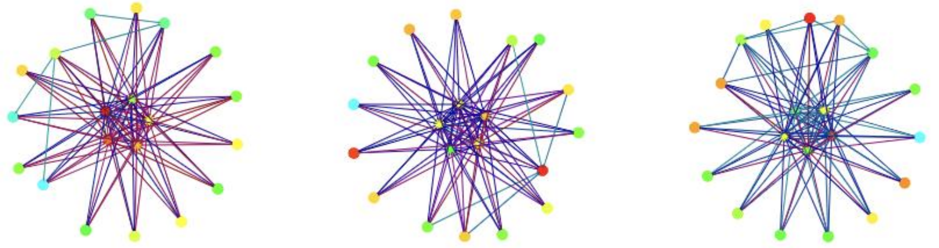
Mini-batch training for proteins, Amazon2M, pokec with batch size 10K/100K

For Papers100M, using batch size 0.4M only requires 3.5 hours on a 24GB GPU

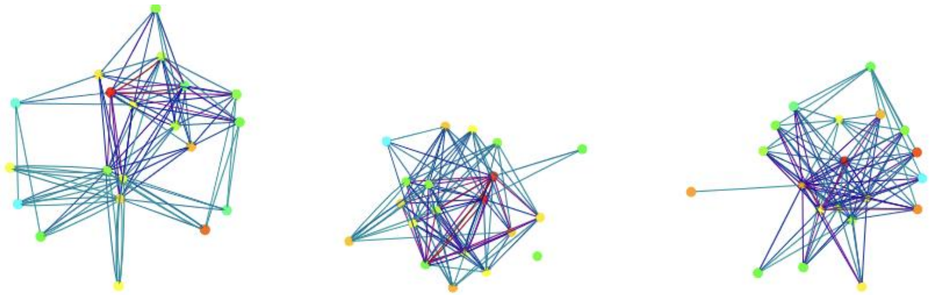
Qitian Wu et al., Simplifying and Empowering Transformers for Large-Graph Representations, NeurIPS 2023

More Application Scenarios

Scenario 1: predicting spatial-temporal dynamics with interpretable latent structures

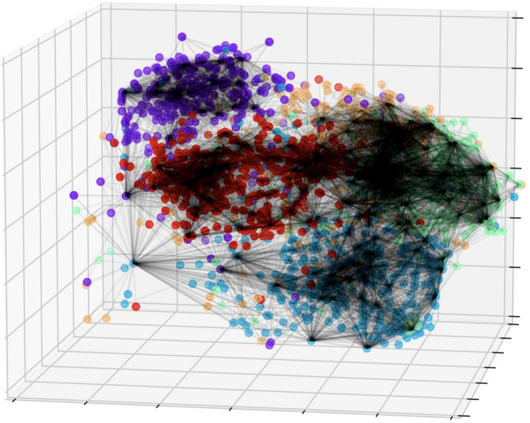


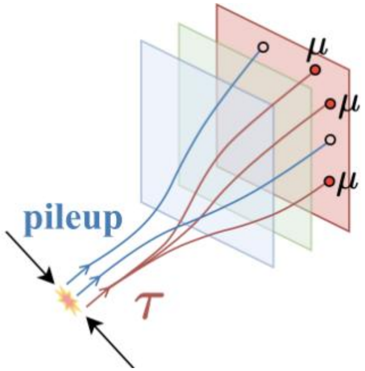
Diffusivity estimates of DIFFormer-s

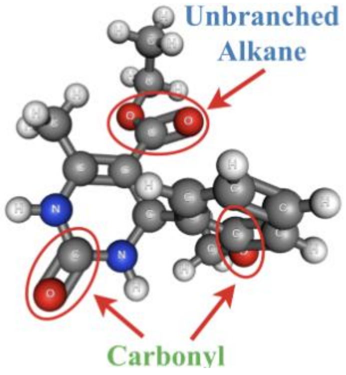


Diffusivity estimates of DIFFormer-a

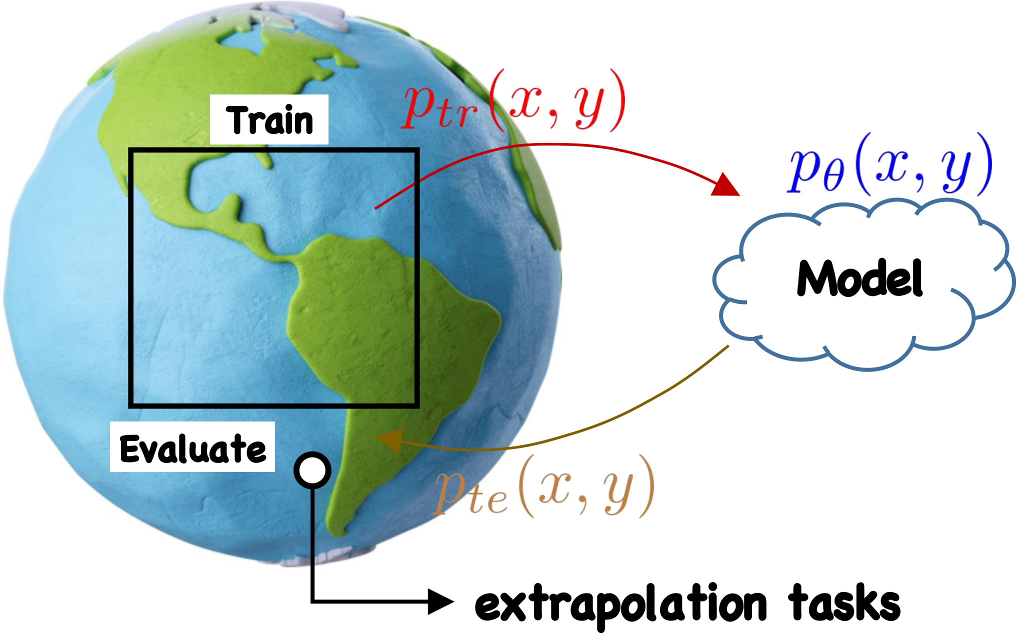
Scenario 2: handling tasks with latent structures in broad areas (particle physics, biochemistry, etc.)







Towards Open-World Learning

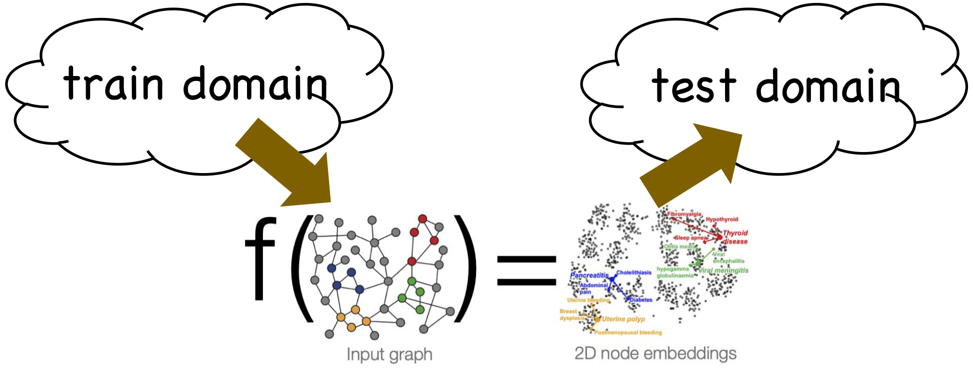


$$\overbrace{\mathcal{D}(p_\theta(x,y), p_{te}(x,y))}^{\text{model performance}} \le$$

$$\underbrace{\mathcal{D}_1(p_\theta(x,y), p_{tr}(x,y))}_{\text{fitting error}} + \underbrace{\mathcal{D}_2(p_{tr}(x,y), p_{te}(x,y))}_{\text{generalization gap}}$$

too small to be good

can be arbitrarily large!

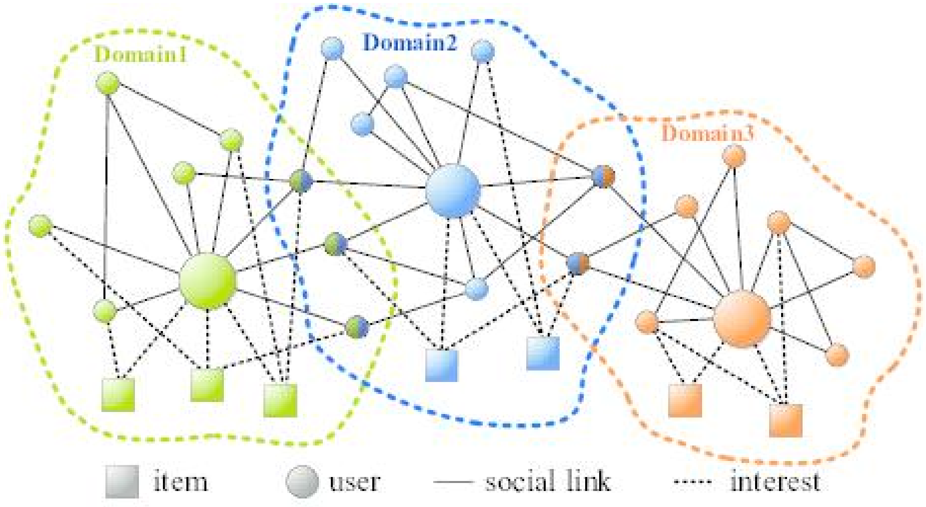


Open research question:

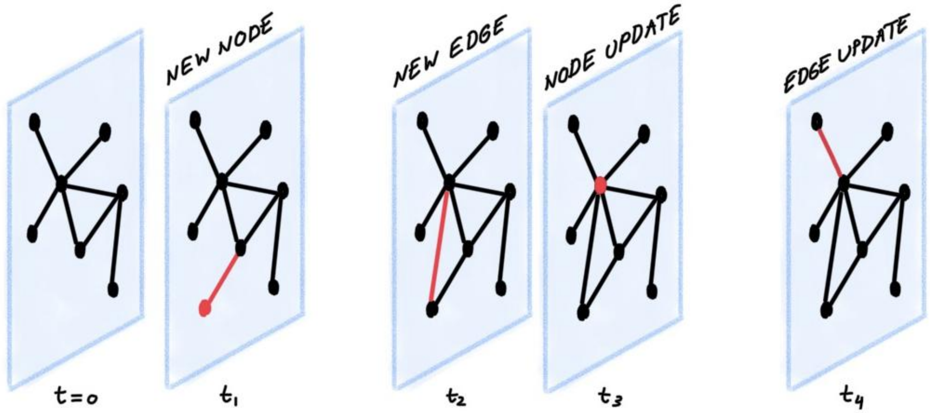
Q1: How powerful are existing models for generalization tasks?

Q2: How to design provably effective generalization approach?

Out-of-Distribution Data from Open World



Graph data from multiple domains

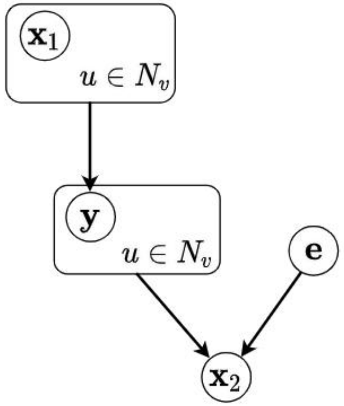


Dynamic temporal networks

- Distribution shifts cause different data distributions $P_{train}(\mathcal{D}) \neq P_{test}(\mathcal{D})$
- New data from unknown distribution are unseen by training

Generalization is impossible w/o any assumption (no free-lunch theorem)

Theoretical Motivation



$\mathbf{x}_1$: publish avenue
$\mathbf{x}_2$: citation index
$\mathbf{y}$: paper's sub-area
$\mathbf{e}$: time of publication

node features $x_v = [x_v^1, x_v^2]$ causal features

predictive model $\hat{y}_v = \frac{1}{|N_v|} \sum_{u \in N_v} \theta_1 \boxed{x_u^1} + \theta_2 \boxed{x_u^2}$

ideal solutions $[\theta_1, \theta_2] = [1, 0]$

Proposition 1 (Failure of Empirical Risk Minimization)

Let the risk under environment e be $R(e) = \frac{1}{|V|} \sum_{v \in V} \mathbb{E}_{\mathbf{y}|\mathbf{G_v}=G_v}[\|\hat{y}_v - y_v\|_2^2]$.

The unique optimal solution for objective $\min_\theta \mathbb{E}_{\mathbf{e}}[R(e)]$ would be $[\theta_1, \theta_2] = [\frac{1+\sigma_e^2}{2+\sigma_e^2}, \frac{1}{2+\sigma_e^2}]$ where $\sigma_e > 0$ denotes the standard deviation of ϵ across environments.

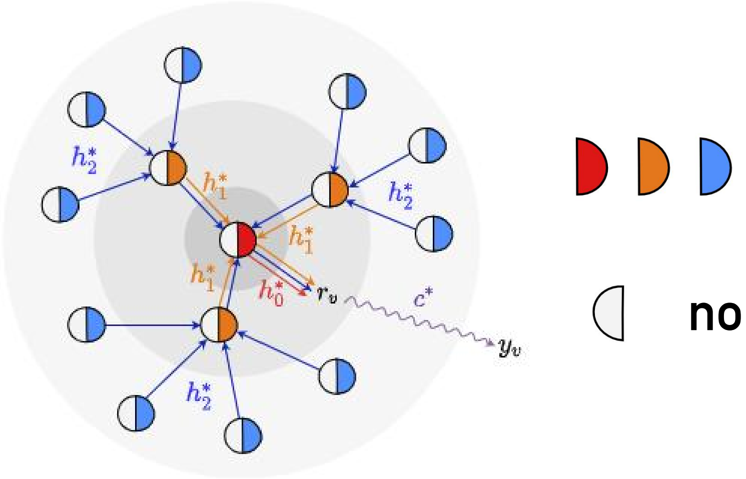
Proposition 2 (Success of Risk Variance Minimization)

The objective $\min_\theta \mathbb{V}_e[R(e)]$ reaches the optimum if and only if $[\theta_1, \theta_2] = [1, 0]$.

Causal Invariance Principle

There exists a portion of causal information within input ego-graph for prediction task of each individual node

The "causal" means two-fold properties:
1) invariant across environments
2) sufficient for prediction



Bernhard Sch¨olkopf, et al., "Invariant models for causal transfer learning".

Theorem 1 (Guarantee of Valid OOD solution)

Under causal assumptions, if the GNN encoder $q(\mathbf{z}|\mathbf{G_v})$ satisfies that 1) $I(\mathbf{y};\mathbf{e}|\mathbf{z})=0$ (invariance condition) and 2) $I(\mathbf{y};\mathbf{z})$ is maximized (sufficiency condition), then the model f^* given by $\mathbb{E}_\mathbf{y}[\mathbf{y}|\mathbf{z}]$ is the solution to the formulated OOD problem.

Qitian Wu, et al., "Handling Distribution Shifts on Graphs: An Invariance Perspective", in ICLR'22

Explore-to-Extrapolate Risk Minimization

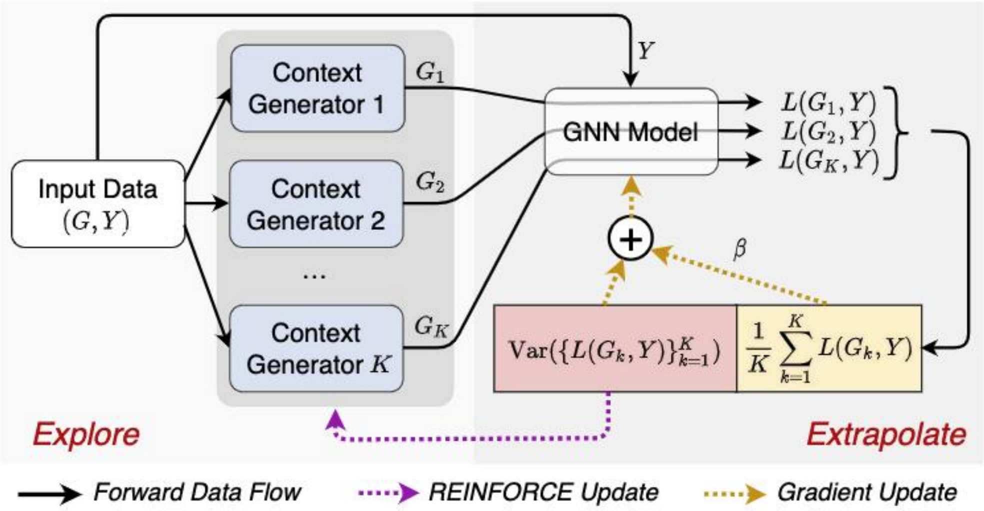
Risk Extrapolation ➡ $$\min_\theta \mathrm{Var}(\{L(g_{w_k^*}(G), Y; \theta) : 1 \le k \le K\}) + \frac{\beta}{K}\sum_{k=1}^{K} L(g_{w_k^*}(G), Y; \theta)$$

Environment Exploration ➡ $$\text{s. t. } [w_1^*, \cdots, w_K^*] = \arg\max_{w_1, \cdots, w_K} \mathrm{Var}(\{L(\boxed{g_{w_k}(G)}, Y; \theta) : 1 \le k \le K\})$$

context generator

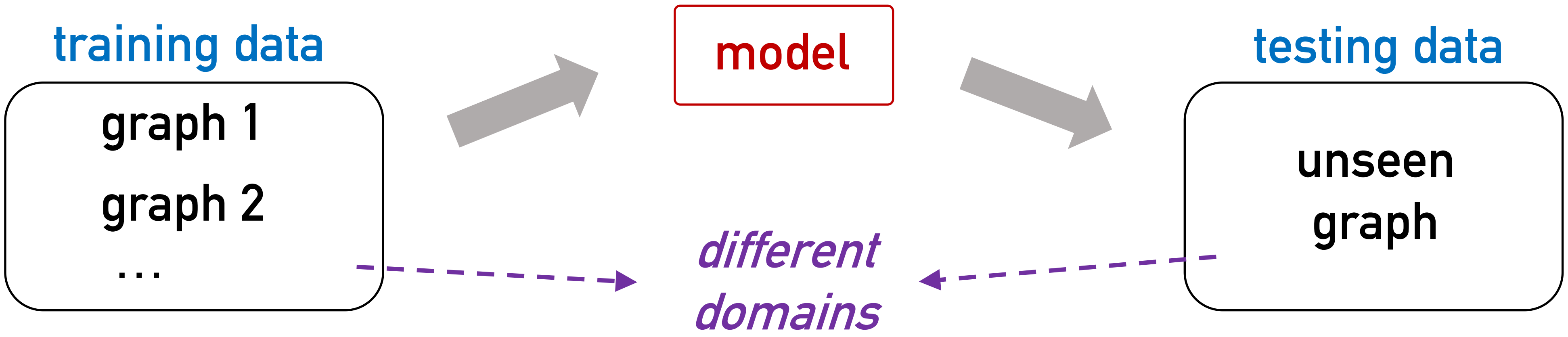
❑ Model instantiations:

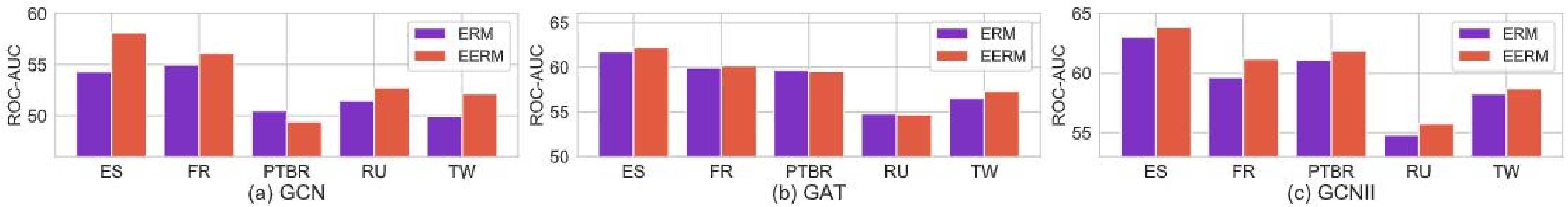
- $f_\theta(\cdot)$: GNN (output node-level prediction)
- $g_{w_k^*}(\cdot)$: Graph Editer (modify graph structures)
- Training: *REINFORCE* + *Gradient Descent*



Qitian Wu, et al., "Handling Distribution Shifts on Graphs: An Invariance Perspective", in ICLR'22

Experiment on Cross-Graph Transfer





EERM achieves up to 7.0% (resp. 7.2%) impv. on ROC-AUC (resp. accuracy) than ERM

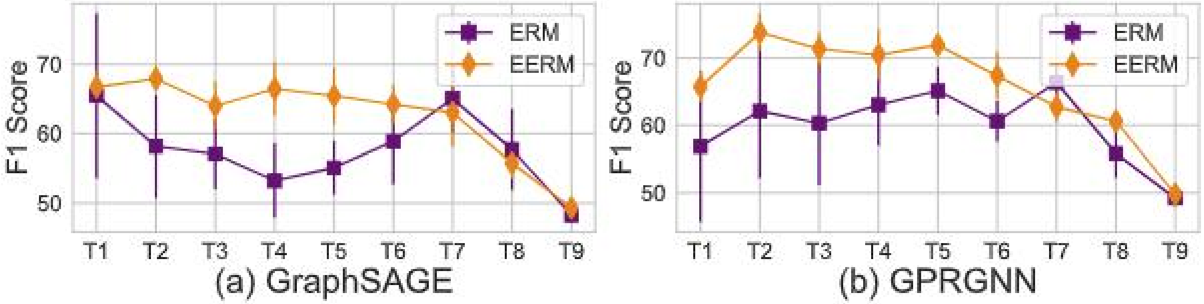
Qitian Wu, et al., “Handling Distribution Shifts on Graphs: An Invariance Perspective”, in ICLR'22

Experiment on Temporal Graph Evolution

training data | | testing data

graph snapshot 1 | graph snapshot 1 | … | graph snapshot M | … … | graph snapshot K

model



(a) GraphSAGE (b) GPRGNN

Method	2014-2016	2016-2018	2018-2020
ERM- SAGE	42.09 ± 1.39	39.92 ± 2.53	36.72 ± 2.47
EERM- SAGE	41.55 ± 0.68	40.36 ± 1.29	38.95 ± 1.57
ERM- GPR	47.25 ± 0.55	45.07 ± 0.57	41.53 ± 0.56
EERM- GPR	49.88 ± 0.49	48.59 ± 0.52	44.88 ± 0.62

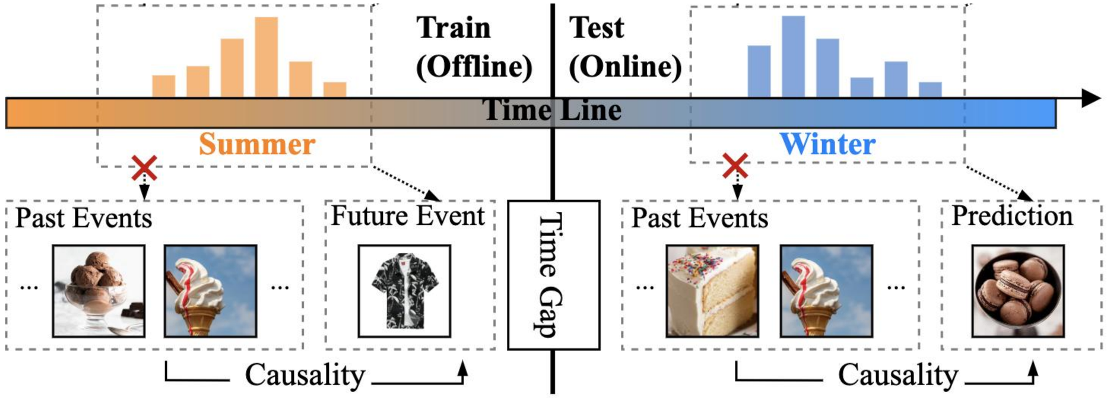
EERM achieves up to 9.6%/10.0% impv using GraphSAGE/GPR-GNN as backbones

Qitian Wu, et al., "Handling Distribution Shifts on Graphs: An Invariance Perspective", in ICLR'22

Applications for Recommender Systems

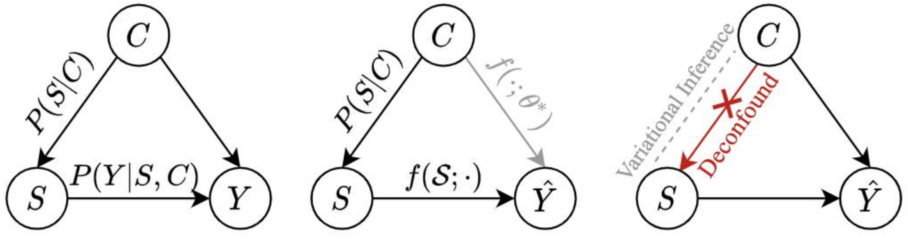
Observation:
There exists latent context (from external effects) that spuriously correlates user clicking behaviors

Key insights:
Learning invariant user interests that causally relate to the clicking behaviors



$$\mathbb{E}_{c\sim Q(C|S=\mathcal{S})}\left[\log P_\theta(Y|S=\mathcal{S},C=c)\right] - \mathcal{D}_{KL}\left(Q(C|S=\mathcal{S})\|P(C)\right)$$

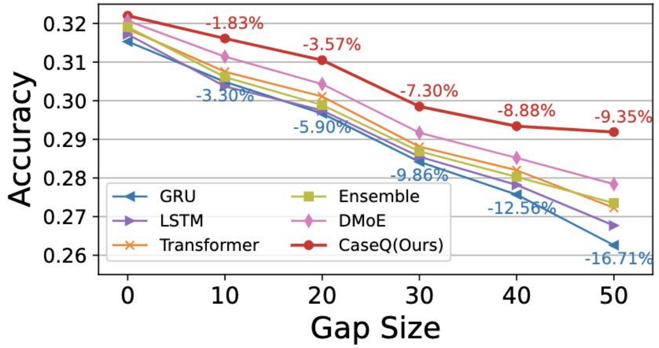
Alleviate drop on
NDCG by **47.77%**
Hit Ratio by **35.73%**



(a) Real-world data generation $P(Y|S)$

(b) Traditional model $P_\theta(Y|S)$

(c) Our interventional model $P_\theta(Y|do(S))$

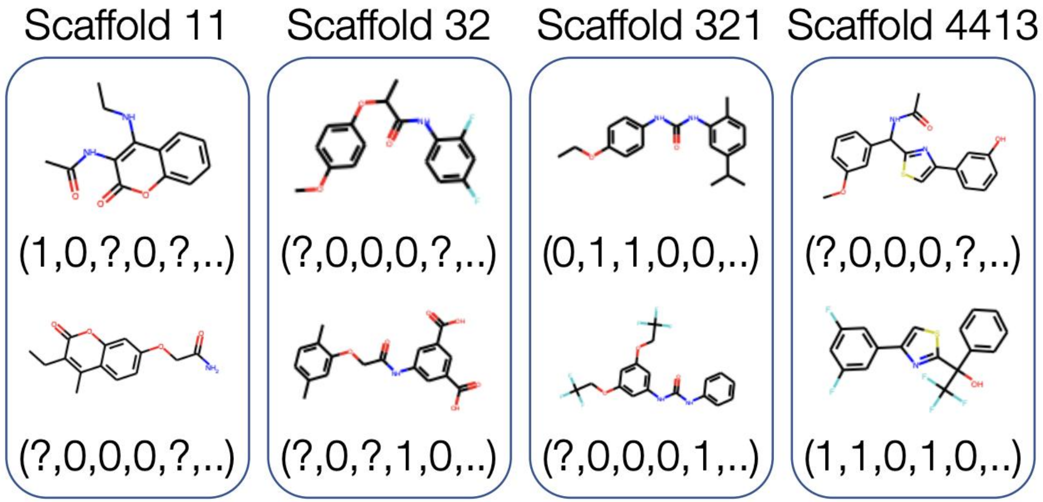


Qitian Wu, et al., “Towards Out-of-Distribution Sequential Event Prediction: A Causal Treatment”, in NeurIPS'22

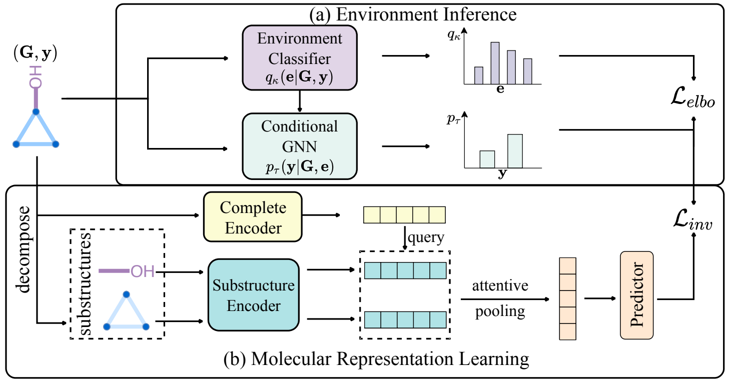
Applications for Molecular Analysis

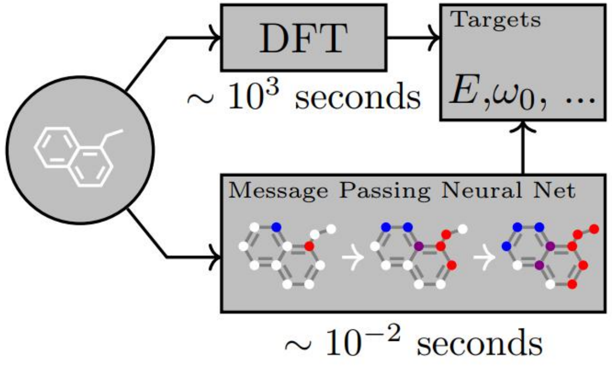
Observation:
There exist certain priviledged substructures that causally relate to the target property

Key insights:
Learning molecular substructures that induce invariant predictive relations with the labels



+ 5.9% and + 3.9% improvement over the strongest baselines on OGB-Mole and DrugOOD





Qitian Wu, et al., "Learning Substructure Invariance for Out-of-Distribution Molecular Representations", in NeurIPS'22

Conclusions

The Open Challenge of Learning with Non-IID Data

Closed-world: representation	Open-world: generalization
Diffusion-inspired graph Transformers [ICLR'23]	Learning with distribution shifts [ICLR'22]
Linearly complex global attention [NeurIPS'22]	Feature space extrapolation [NeurIPS'21]
Simplifying Global Transformers [NeurIPS'23]	Invariant substructure learning [NeurIPS'22]
Universal structure learning [KDD'23]	Theoretical understandings of generalization [ICLR'23]

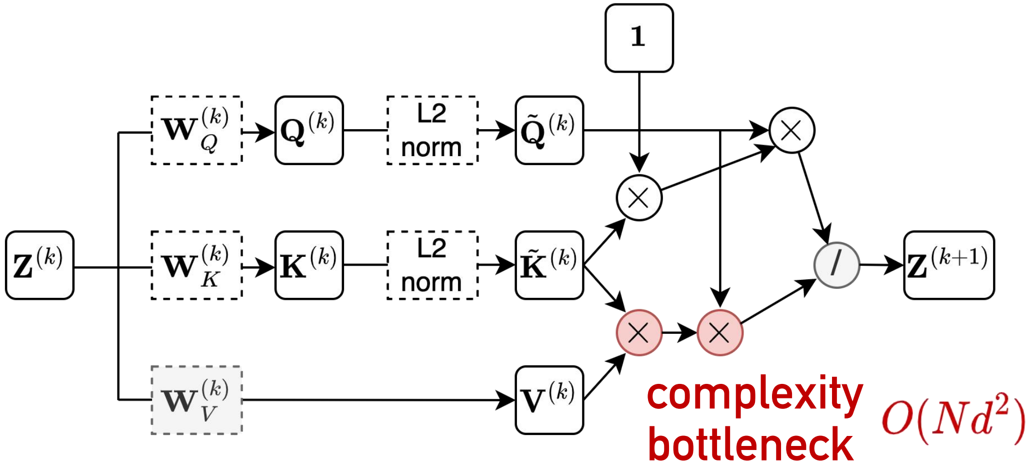
[1] Qitian Wu, et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, in ICLR'23 (spotlight oral)
[2] Qitian Wu, et al., NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification, in NeurIPS'22 (spotlight)
[3] Qitian Wu, et al., Simplifying and Empowering Transformers for Large-Graph Representations, in NeurIPS'23
[4] Qitian Wu, et al., Handling Distribution Shifts on Graphs: An Invariance Perspective, in ICLR'22
[5] Qitian Wu, et al., Energy-based Out-of-Distribution Detection for Graph Neural Networks, in ICLR'23
[6] Qitian Wu, et al,, Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach, in NeurIPS'21
[7] Nianzu Yang, Qitian Wu, et al., Learning Substructure Invariance for Out-of-Distribution Molecular Representations, in NeurIPS'22 (spotlight)
[8] Chenxiao Yang, QItian Wu, et al., Graph Neural Networks are Inherently Good Generalizers: Insights by Bridging GNNs and MLPs, in ICLR'23
[9] Wentao Zhao, Qitian Wu, et al., GraphGLOW: Universal and Generalizable Structure Learning for Graph Neural Networks, in SIGKDD'23 (oral)

DIFFormer: Instantiations of Diffusivity

DIFFormer layer with simple diffusivity (DIFFormer-s):

$$\omega_{ij}^{(k)} = f(\|\tilde{\mathbf{z}}_i^{(k)} - \tilde{\mathbf{z}}_j^{(k)}\|_2^2) = 1 + \left(\frac{\mathbf{z}_i^{(k)}}{\|\mathbf{z}_i^{(k)}\|_2}\right)^\top \left(\frac{\mathbf{z}_j^{(k)}}{\|\mathbf{z}_j^{(k)}\|_2}\right)$$

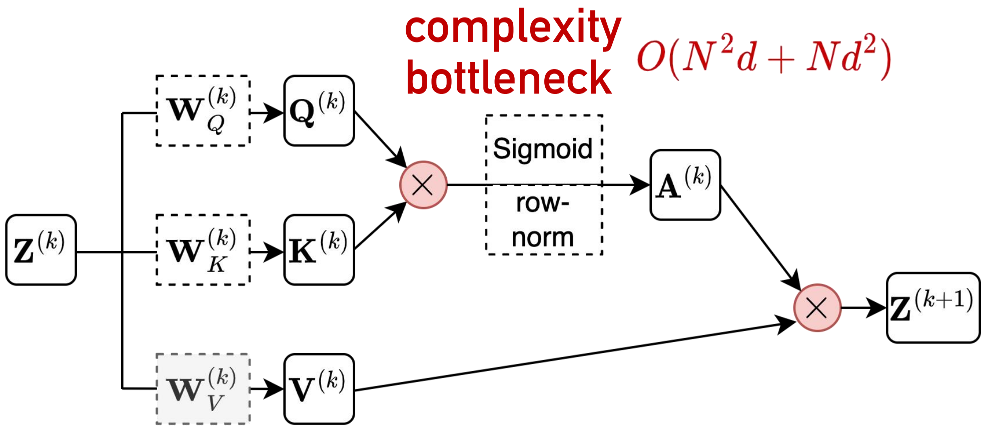
$$\sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)} = \sum_{j=1}^{N} \frac{1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_j^{(k)}}{\sum_{l=1}^{N} \left(1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_l^{(k)}\right)} \mathbf{z}_j^{(k)}$$



DIFFormer layer with advanced diffusivity (DIFFormer-a):

$$\omega_{ij}^{(k)} = f(\|\tilde{\mathbf{z}}_i^{(k)} - \tilde{\mathbf{z}}_j^{(k)}\|_2^2) = \frac{1}{1 + \exp\left(-(\mathbf{z}_i^{(k)})^\top (\mathbf{z}_j^{(k)})\right)}$$

$$\sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)} = \sum_{j=1}^{N} \frac{\text{sigmoid}\left((\mathbf{z}_i^{(k)})^\top \mathbf{z}_j^{(k)}\right)}{\sum_{l=1}^{N} \text{sigmoid}\left((\mathbf{z}_i^{(k)})^\top \mathbf{z}_l^{(k)}\right)} \mathbf{z}_j^{(k)}$$



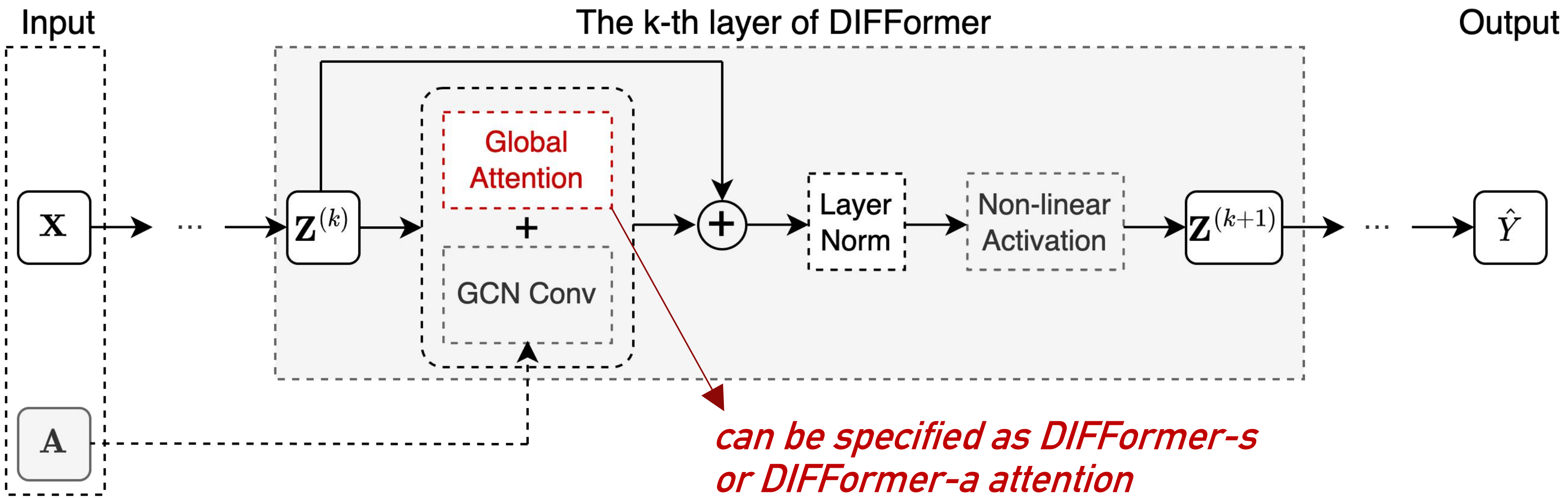
DIFFormer: Extension to a Transformer Layer

Incorporation of input graphs (if available): add graph convolution with global attention

$$\overline{\mathbf{P}}^{(k)} = \frac{1}{2}\left(\hat{\mathbf{S}}^{(k)} + \tilde{\mathbf{A}}\right)\mathbf{Z}^{(k)}$$

DIFFormer layer for updating embedding of the next layer:

$$\mathbf{Z}^{(k+1)} = \sigma'\left(\mathrm{LayerNorm}\left(\tau\overline{\mathbf{P}}^{(k)} + (1-\tau)\mathbf{Z}^{(k)}\right)\right)$$



can be specified as DIFFormer-s or DIFFormer-a attention

DIFFormer: Scaling to Large-Scale Datasets

Large-scale datasets with massive amount of data, e.g., N instances (N can be arbitrarily large)

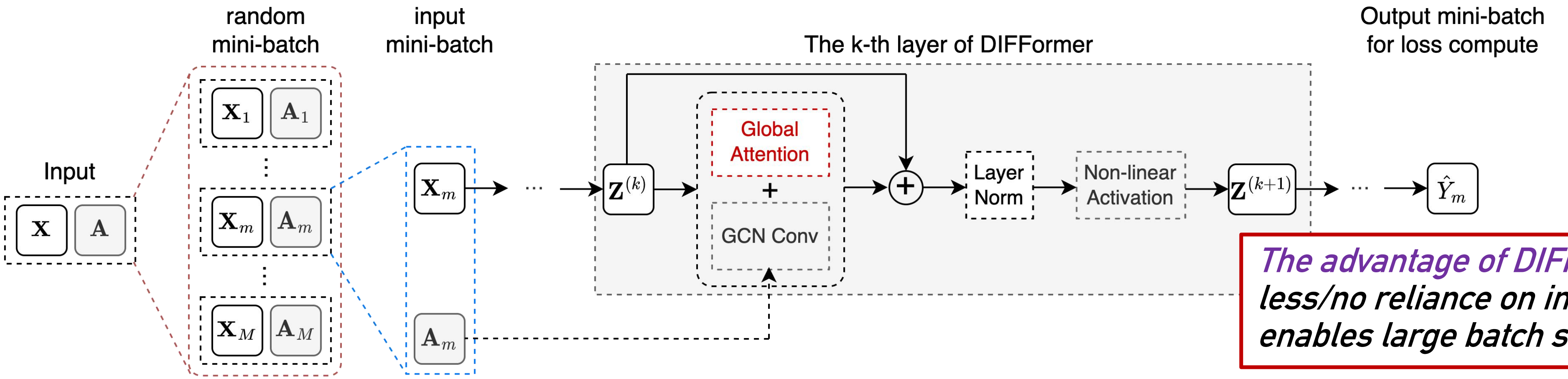
Traditional IID learning enables mini-batch learning with a moderate batch size B << N

How can message passing networks handle large-scale graphs?

Existing solutions: 1. neighbor sampling (slow training and limited receptive field)

2. graph clustering (time-consuming pre-processing and limited receptive field)

Our solution: partition instances into random mini-batches with a large batch size B



The advantage of DIFFormer:
less/no reliance on input graphs enables large batch size

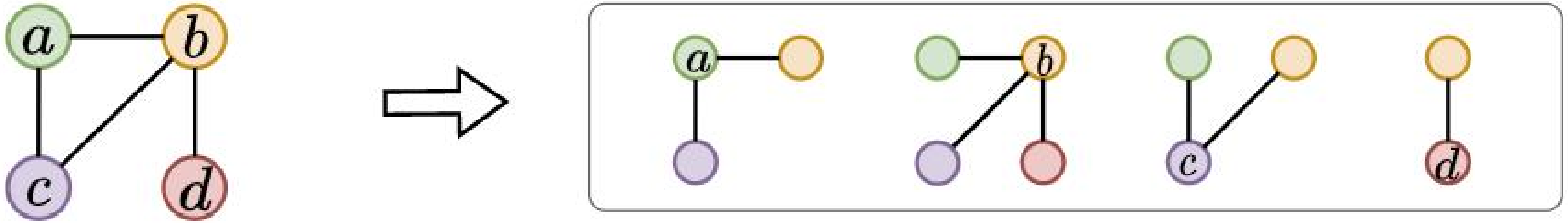
Problem Formulation

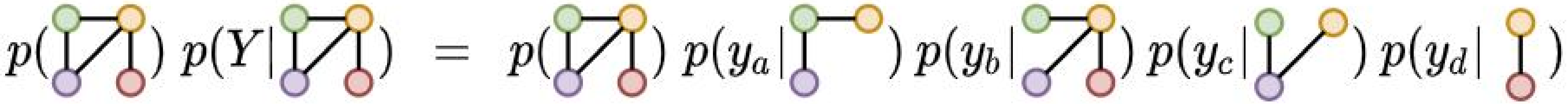
- **Graph notation:** A graph $G = (A, X)$, adjacency matrix $A = \{a_{uv} | v, u \in V\}$ node features $X = \{x_v | v \in V\}$, node labels $Y = \{y_v | v \in V\}$

$$p(\mathbf{G}, \mathbf{Y} | \mathbf{e}) = p(\mathbf{G} | \mathbf{e}) p(\mathbf{Y} | \mathbf{G}, \mathbf{e})$$

where e denotes environment (that affects data generation)

- How to deal with the non-IID nature of nodes in a graph?





$$p(\mathbf{G}|\mathbf{e}) \cdot (\mathbf{Y}|\mathbf{G}, \mathbf{e}) \quad = \quad p(\mathbf{G}|\mathbf{e}) \cdot \prod_{v \in V} p(\mathbf{y}|\mathbf{G_v} = G_v, \mathbf{e})$$

Decompose a graph into pieces of ego-graphs

Problem Formulation

- **Graph notation:** A graph $G=(A,X)$, adjacency matrix $A=\{a_{uv}|v,u\in V\}$ node features $X=\{x_v|v\in V\}$, node labels $Y=\{y_v|v\in V\}$

$$p(\mathbf{G},\mathbf{Y}|\mathbf{e})=p(\mathbf{G}|\mathbf{e})p(\mathbf{Y}|\mathbf{G},\mathbf{e})$$

where e denotes environment (that affects data generation)

- **Out-of-distribution generalization on graphs:**

sample a whole graph from a specific environment

sample node-level label conditioned on ego-graph and environment

learn a classifier robust for worst case

$$\boxed{\min_f \max_{e\in\mathcal{E}}} \mathbb{E}_{\boxed{G\sim p(\mathbf{G}|\mathbf{e}=e)}}\left[\frac{1}{|V|}\sum_{v\in V}\mathbb{E}_{\boxed{y\sim p(\mathbf{y}|\mathbf{G_v}=G_v,\mathbf{e}=e)}}\boxed{[l(f(G_v),y)]}\right]$$

loss function for node-level prediction

  - A graph G can be divided into pieces of ego-graphs $\{(G_v,y_v)\}_{v\in V}$
  - The data generation process: 1) the entire graph is generated via $G\sim p(\mathbf{G}|\mathbf{e})$, 2) each node's label is generated via $y\sim p(\mathbf{y}|\mathbf{G_v}=G_v,\mathbf{e})$
  - Denote $\mathcal{E}$ as the support of env. and $l(\cdot,\cdot)$ as the loss function