DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, Junchi Yan

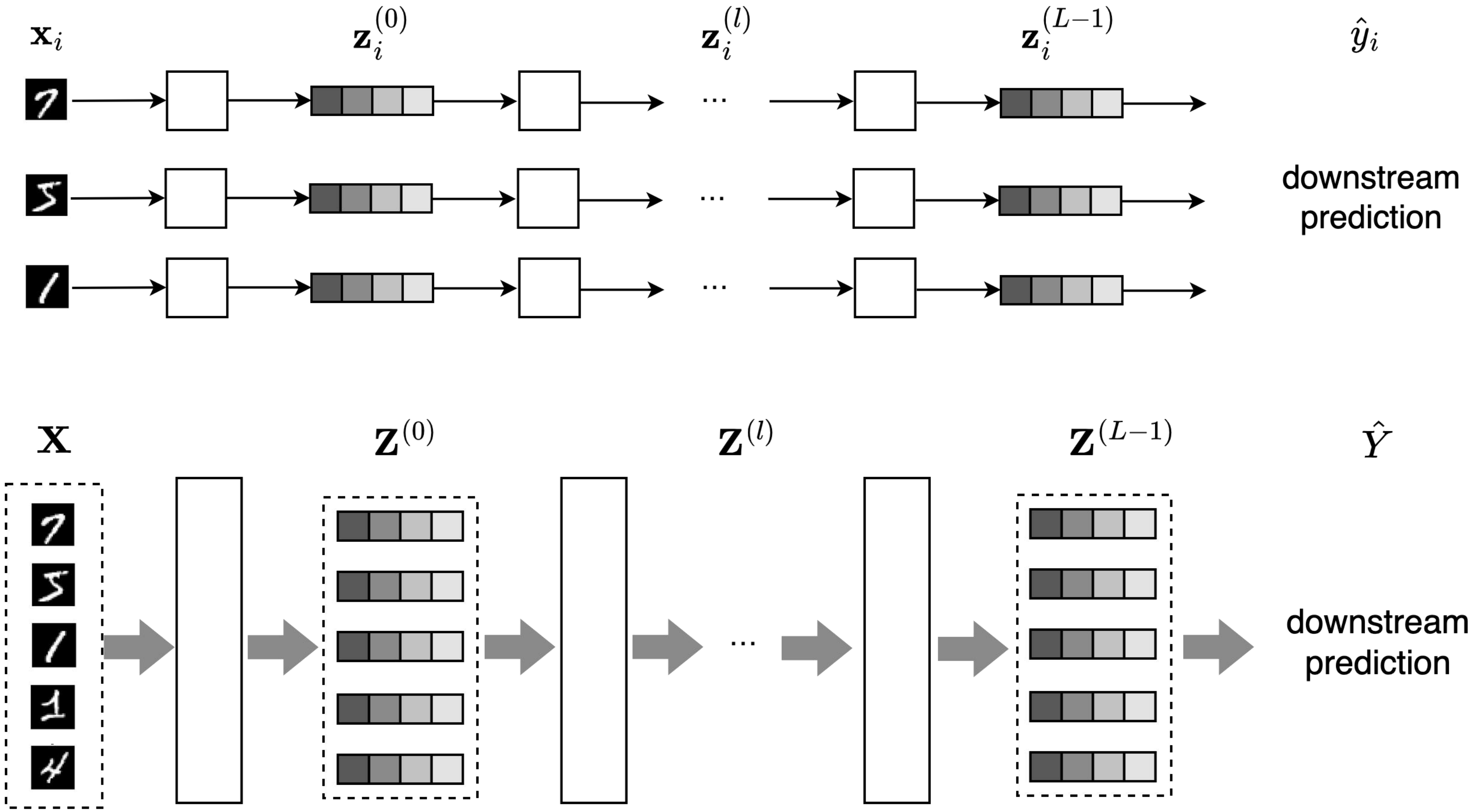
Learning with IID v.s. non-IID Hypothesis

IID Learning:
Independently process each instance

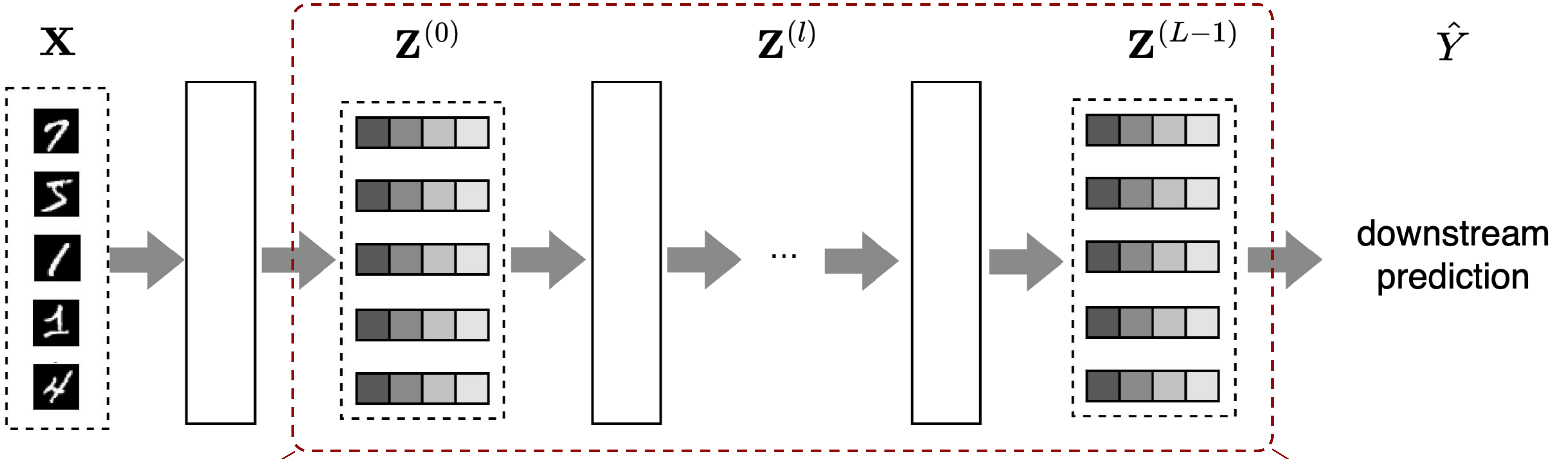
v. s.

Non-IID Learning:
Collectively process a batch of instances

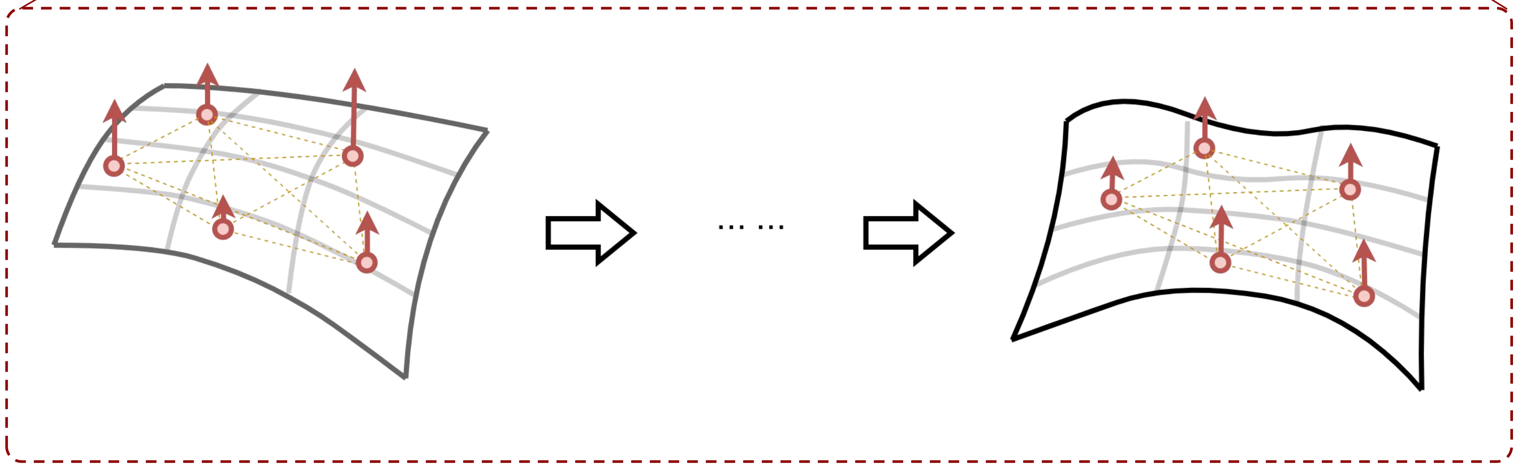
Key question: how to encode the interactions for representations

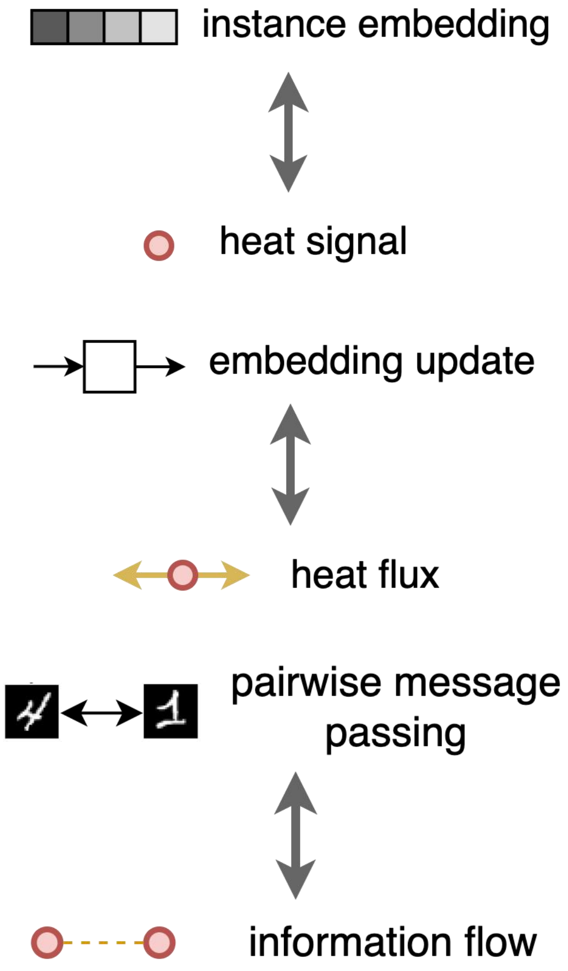


NN Feed-forward as Diffusion Process



Treat the feed-forward update of embeddings as a diffusion process of heat on locations



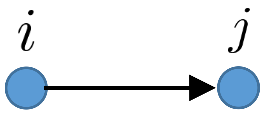


General Formulation of Diffusion Process

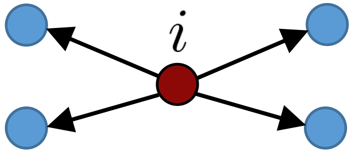
The diffusion process of N particles driven by initial states and pairwise interactions:

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

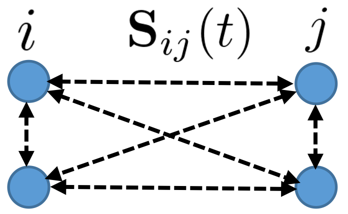
Important concepts:



gradient



divergence



diffusivity function

$$(\nabla \mathbf{Z}(t))_{ij} = \mathbf{z}_j(t) - \mathbf{z}_i(t) \qquad (\nabla^*)_i = \sum_{j=1}^N \mathbf{S}_{ij}(\mathbf{Z}(t), t) \, (\nabla \mathbf{Z}(t))_{ij} \qquad \mathbf{S}(\mathbf{Z}(t), t) : \mathbb{R}^{N \times d} \times [0, \infty) \to [0, 1]^{N \times N}$$

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

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

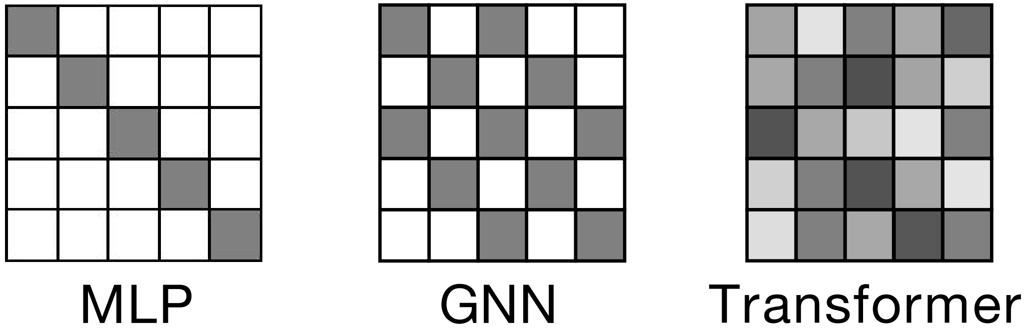
Diffusion with Latent Structures

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 $\mathbf{S}^{(k)}$ is a measure of the rate at which the node signals spread

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



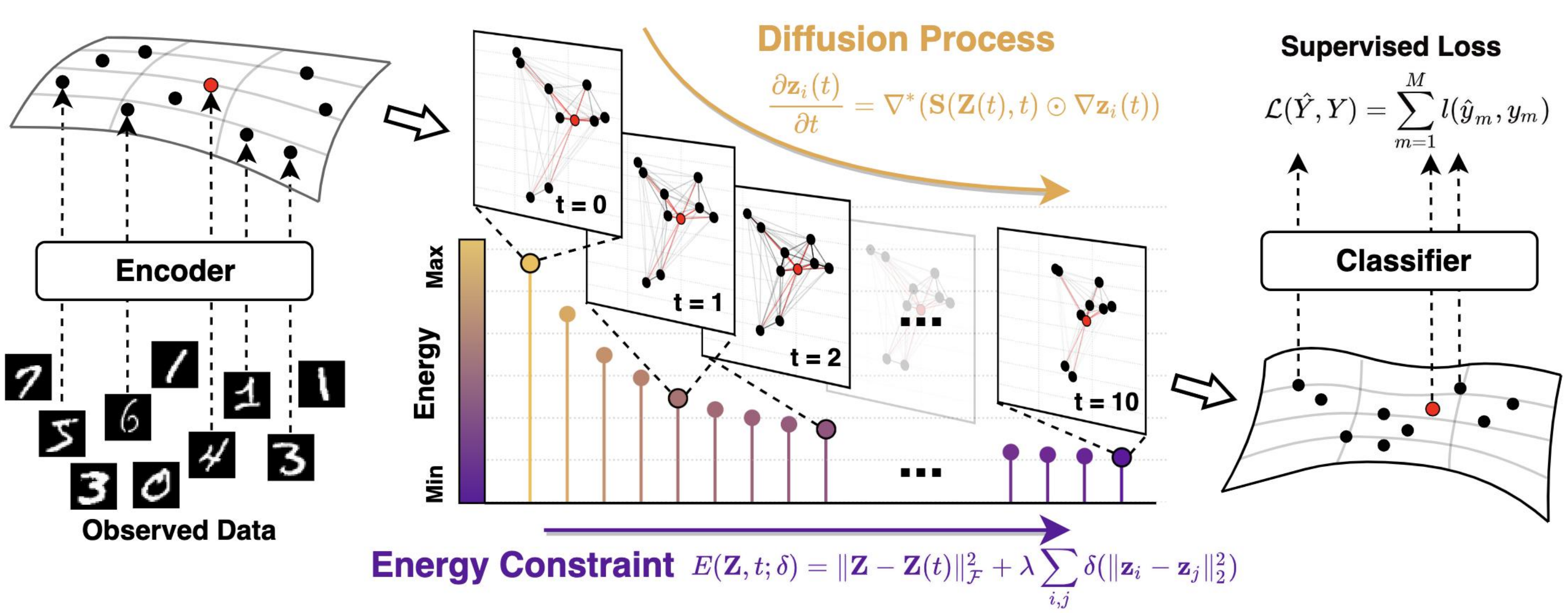
Key question: How to determine a proper diffusivity function for learning desirable node representations?

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

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

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

Closed-Form Solutions for Diffusion Dynamics

Theorem (Optimal Diffusivity Estimates for Energy-Constrained Diffusion)

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., $E(\mathbf{Z}^{(k+1)}, k; \delta) \leq E(\mathbf{Z}^{(k)}, k-1; \delta)$

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

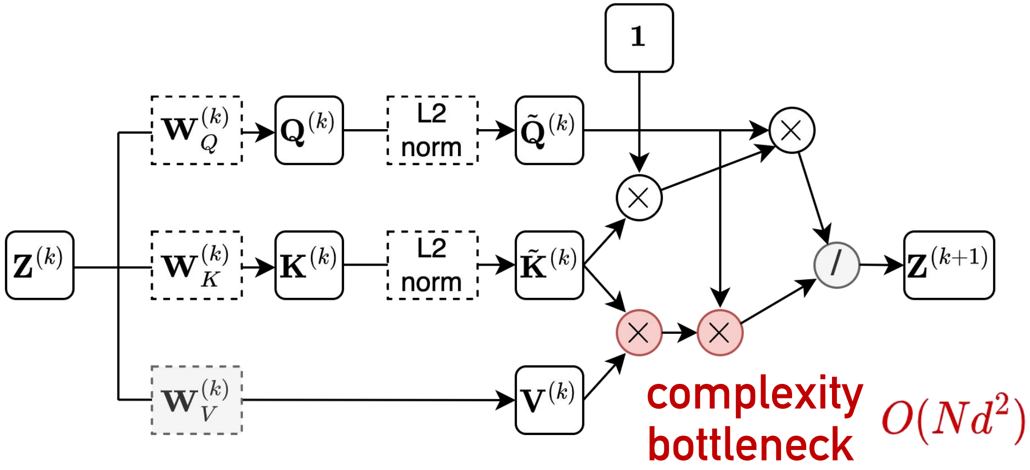
State Update: $$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^N \hat{\mathbf{S}}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^N \hat{\mathbf{S}}_{ij}^{(k)} \mathbf{z}_j^{(k)}, \quad 1 \leq i \leq N$$

DIFFormer: Instantiations of Diffusivity

DIFFormer layer with simple diffusivity (DIFFormer-s):

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

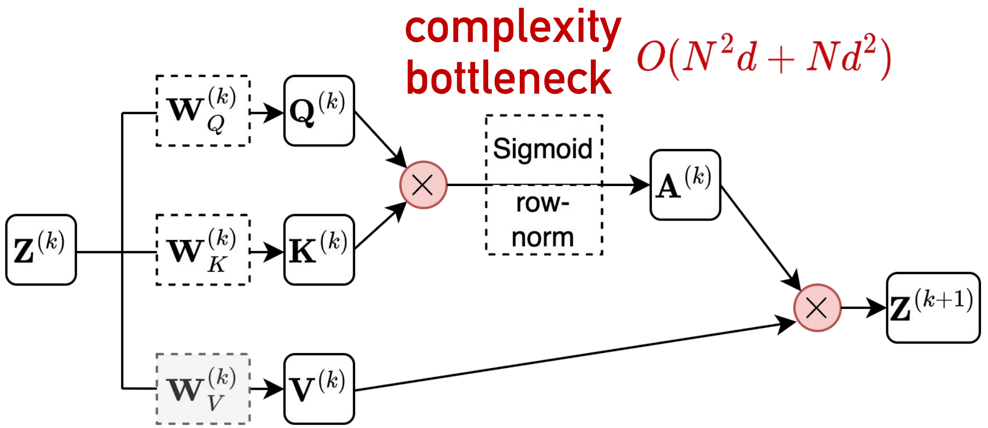
$$\sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)} = \sum_{j=1}^{N} \frac{1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_j^{(k)}}{\sum_{l=1}^{N} \left(1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_l^{(k)}\right)} \mathbf{z}_j^{(k)}$$



DIFFormer layer with advanced diffusivity (DIFFormer-a):

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

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



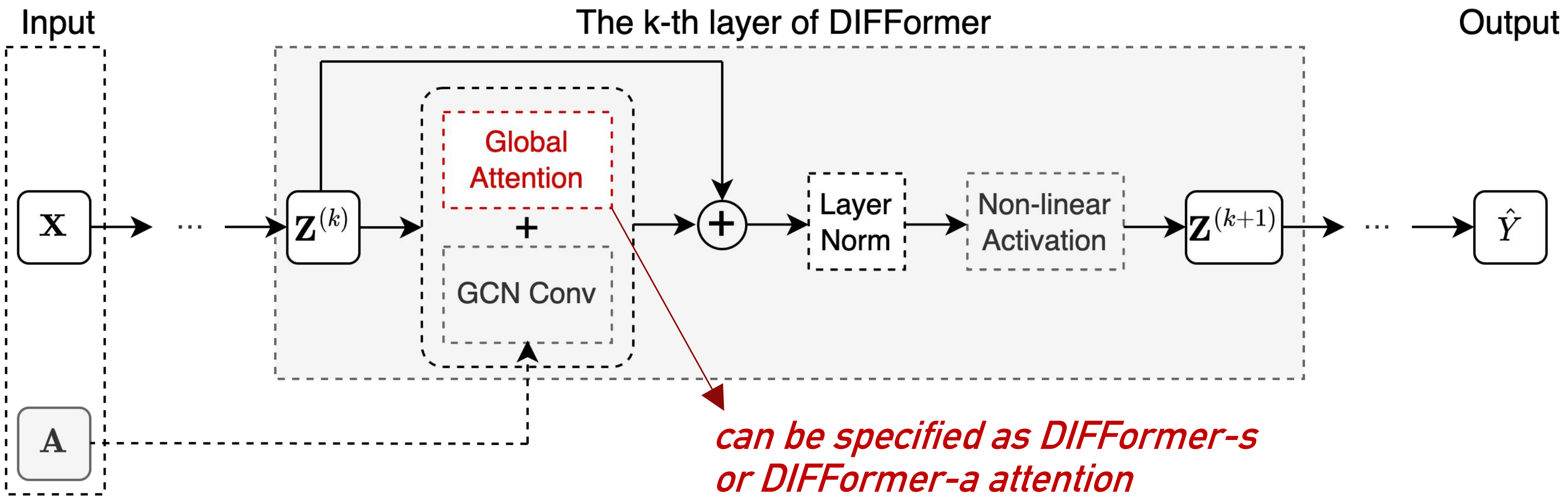
DIFFormer: Extension to a Transformer Layer

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:

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



DIFFormer: Scaling to Large-Scale Datasets

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

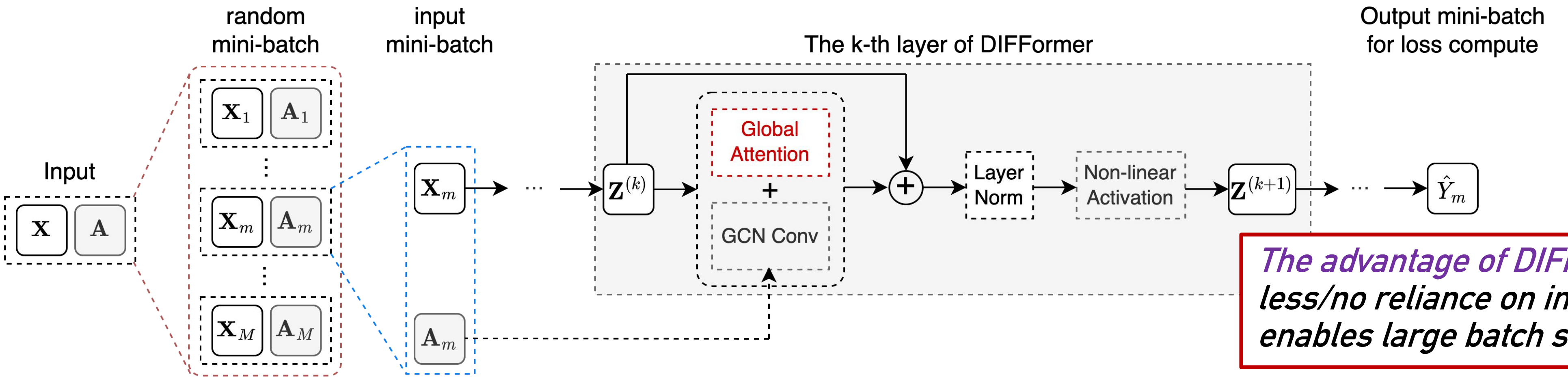
Traditional IID learning enables mini-batch learning with a moderate batch size B << N

How can message passing networks handle large-scale graphs?

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

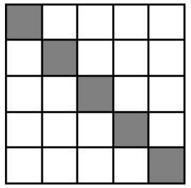
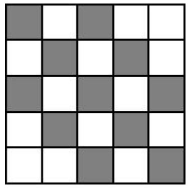
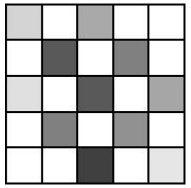
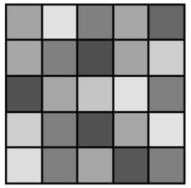
2. graph clustering (time-consuming pre-processing and limited receptive field)

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

Interpretations of MLP/GNNs as Diffusion

	Energy function	Diffusivity	Illustration
MLP	$\|\mathbf{Z} - \mathbf{Z}^{(k)}\|_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} 1, & \text{if } i = j \\ 0, & otherwise \end{cases}$	
GCN	$\sum_{(i,j)\in\mathcal{E}} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{1}{\sqrt{d_i d_j}}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
GAT	$\sum_{(i,j)\in\mathcal{E}} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l:(i,l)\in\mathcal{E}} f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
DIFFormer	$\|\mathbf{Z} - \mathbf{Z}^{(k)}\|_2^2 + \lambda \sum_{i,j} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$	$\mathbf{S}_{ij}^{(k)} = \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l=1}^{N} f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, \quad 1 \le i,j \le N$	

Results on Graph-based Node Classification

Results of testing accuracy on semi-supervised node classification (20 nodes per class for train)

Type	Model	Non-linearity	PDE-solver	Input-G	Cora	Citeseer	Pubmed
Basic models	MLP	R	-	-	56.1 ± 1.6	56.7 ± 1.7	69.8 ± 1.5
	LP	-	-	R	68.2	42.8	65.8
	ManiReg	R	-	R	60.4 ± 0.8	67.2 ± 1.6	71.3 ± 1.4
Standard GNNs	GCN	R	-	R	81.5 ± 1.3	71.9 ± 1.9	77.8 ± 2.9
	GAT	R	-	R	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
	SGC	-	-	R	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0
	GCN-kNN	R	-	-	72.2 ± 1.8	56.8 ± 3.2	74.5 ± 3.2
	GAT-kNN	R	-	-	73.8 ± 1.7	56.4 ± 3.8	75.4 ± 1.3
	Dense GAT	R	-	-	78.5 ± 2.5	66.4 ± 1.5	66.4 ± 1.5
	LDS	R	-	-	83.9 ± 0.6	**74.8 ± 0.3**	out-of-memory
	GLCN	R	-	-	83.1 ± 0.5	72.5 ± 0.9	78.4 ± 1.5
Diffusion-based models	GRAND-l	-	R	R	83.6 ± 1.0	73.4 ± 0.5	78.8 ± 1.7
	GRAND	R	R	R	83.3 ± 1.3	74.1 ± 1.7	78.1 ± 2.1
	GRAND++	R	R	R	82.2 ± 1.1	73.3 ± 0.9	78.1 ± 0.9
	GDC	R	-	R	83.6 ± 0.2	73.4 ± 0.3	78.7 ± 0.4
	GraphHeat	R	-	R	83.7	72.5	80.5
	DGC-Euler	-	-	R	83.3 ± 0.0	73.3 ± 0.1	80.3 ± 0.1
Graph Transformers	NodeFormer	-	-	-	83.4 ± 0.2	73.0 ± 0.3	**81.5 ± 0.4**
	DIFFormer-s	-	-	-	**85.9 ± 0.4**	73.5 ± 0.3	**81.8 ± 0.3**
	DIFFormer-a	-	-	-	**84.1 ± 0.6**	**75.7 ± 0.3**	80.5 ± 1.2

Results on Graph-based Node Classification

Proteins: 132,534 nodes, 39,561,252 edges
Pokec: 1,632,803 nodes, 30,622,564 edges

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***

We use batch size 10K/100K for training DIFFormer-s using a single GPU on Proteins/Pokec

Test Acc and memory costs of different batch sizes on Pokec

Batch size	5000	10000	20000	50000	100000	200000
Test Acc (%)	65.24 ± 0.34	67.48 ± 0.81	68.53 ± 0.75	68.96 ± 0.63	69.24 ± 0.76	69.15 ± 0.52
GPU Memory (MB)	1244	1326	1539	2060	2928	4011

Results on Image & Text Classification

Results of testing accuracy on semi-supervised image and text classification

Dataset		MLP	LP	ManiReg	GCN-kNN	GAT-kNN	DenseGAT	GLCN	DIFFormer-s	DIFFormer-a
CIFAR	100 labels	65.9 ± 1.3	66.2	67.0 ± 1.9	66.7 ± 1.5	66.0 ± 2.1	out-of-memory	66.6 ± 1.4	**69.1 ± 1.1**	**69.3 ± 1.4**
	500 labels	73.2 ± 0.4	70.6	72.6 ± 1.2	72.9 ± 0.4	72.4 ± 0.5	out-of-memory	72.8 ± 0.5	**74.8 ± 0.5**	**74.0 ± 0.6**
	1000 labels	75.4 ± 0.6	71.9	74.3 ± 0.4	74.7 ± 0.5	74.1 ± 0.5	out-of-memory	74.7 ± 0.3	**76.6 ± 0.3**	**75.9 ± 0.3**
STL	100 labels	66.2 ± 1.4	65.2	66.5 ± 1.9	**66.9 ± 0.5**	66.5 ± 0.8	out-of-memory	66.4 ± 0.8	**67.8 ± 1.1**	66.8 ± 1.1
	500 labels	**73.0 ± 0.8**	71.8	72.5 ± 0.5	72.1 ± 0.8	72.0 ± 0.8	out-of-memory	72.4 ± 1.3	**73.7 ± 0.6**	72.9 ± 0.7
	1000 labels	75.0 ± 0.8	72.7	74.2 ± 0.5	73.7 ± 0.4	73.9 ± 0.6	out-of-memory	74.3 ± 0.7	**76.4 ± 0.5**	**75.3 ± 0.6**
20News	1000 labels	54.1 ± 0.9	55.9	56.3 ± 1.2	56.1 ± 0.6	55.2 ± 0.8	54.6 ± 0.2	56.2 ± 0.8	**57.7 ± 0.3**	**57.9 ± 0.7**
	2000 labels	57.8 ± 0.9	57.6	60.0 ± 0.8	60.6 ± 1.3	59.1 ± 2.2	59.3 ± 1.4	60.2 ± 0.7	**61.2 ± 0.6**	**61.3 ± 1.0**
	4000 labels	62.4 ± 0.6	59.5	63.6 ± 0.7	64.3 ± 1.0	62.9 ± 0.7	62.4 ± 1.0	64.1 ± 0.8	**65.9 ± 0.8**	**64.8 ± 1.0**

For image datasets, use a pretrained network to obtain embeddings of images

Use k-nearest-neighbor to construct a graph for baseline methods GCN-kNN and GAT-kNN

DIFFormer-s and DIFFormer-a without using any graph structure outperform the competitors

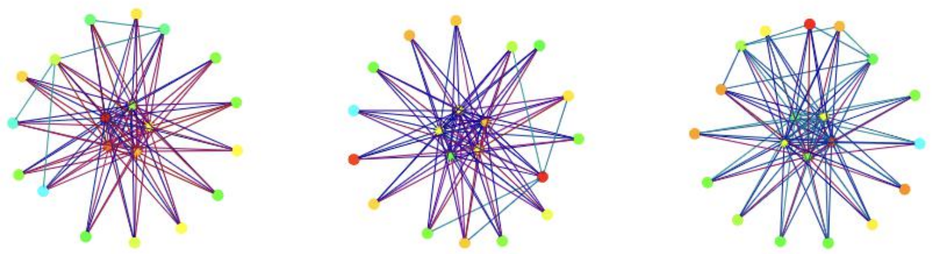
Results on Spatial-Temporal Prediction

Results of testing mean square error for predicting spatial-temporal dynamics based on history

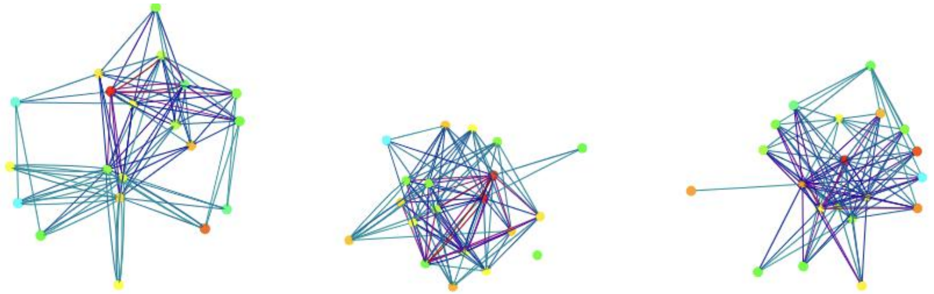
Dataset	MLP	GCN	GAT	Dense GAT	GAT-kNN	GCN-kNN	DIFFORMER-s	DIFFORMER-a	DIFFORMER-s w/o g	DIFFORMER-a w/o g
Chickenpox	0.924 (±0.001)	0.923 (±0.001)	0.924 (±0.002)	0.935 (±0.005)	0.926 (±0.004)	0.936 (±0.004)	**0.914 (0.006)**	**0.915 (0.008)**	0.916 (0.006)	0.916 (0.006)
Covid	0.956 (±0.198)	1.080 (±0.162)	1.052 (±0.336)	1.524 (±0.319)	0.861 (±0.123)	1.475 (±0.560)	0.779 (0.037)	**0.757 (0.048)**	0.779 (0.028)	**0.741 (0.052)**
WikiMath	1.073 (±0.042)	1.292 (±0.125)	1.339 (±0.073)	0.826 (±0.070)	0.882 (±0.015)	1.023 (±0.058)	0.731 (0.007)	0.763 (0.020)	**0.727 (0.025)**	**0.716 (0.030)**

Goal: Given the historical graph snapshot, one needs to predict node labels at the next step

DIFFormer without using graph structure (w/o g) can sometimes yield better prediction

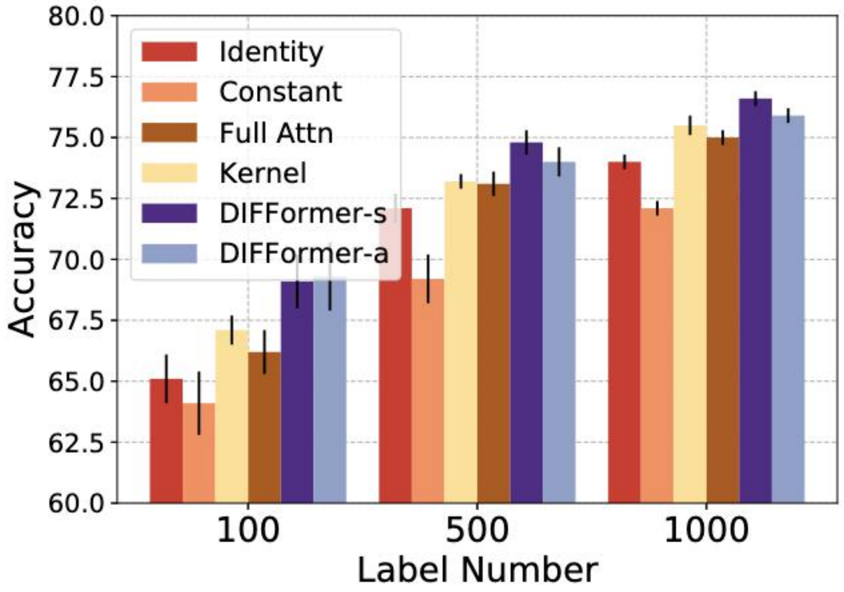


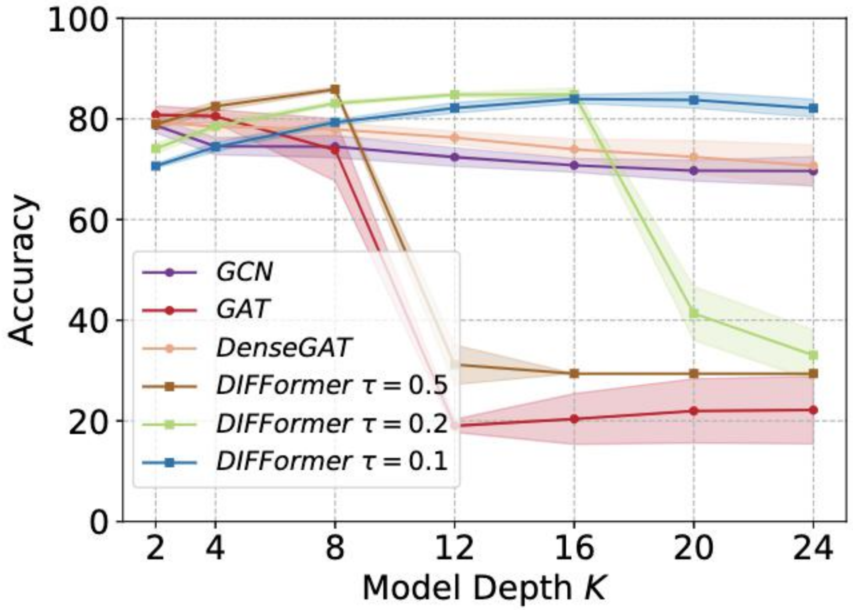
Diffusivity estimates of DIFFormer-s



Diffusivity estimates of DIFFormer-a

Ablation Study and Hyperparameters

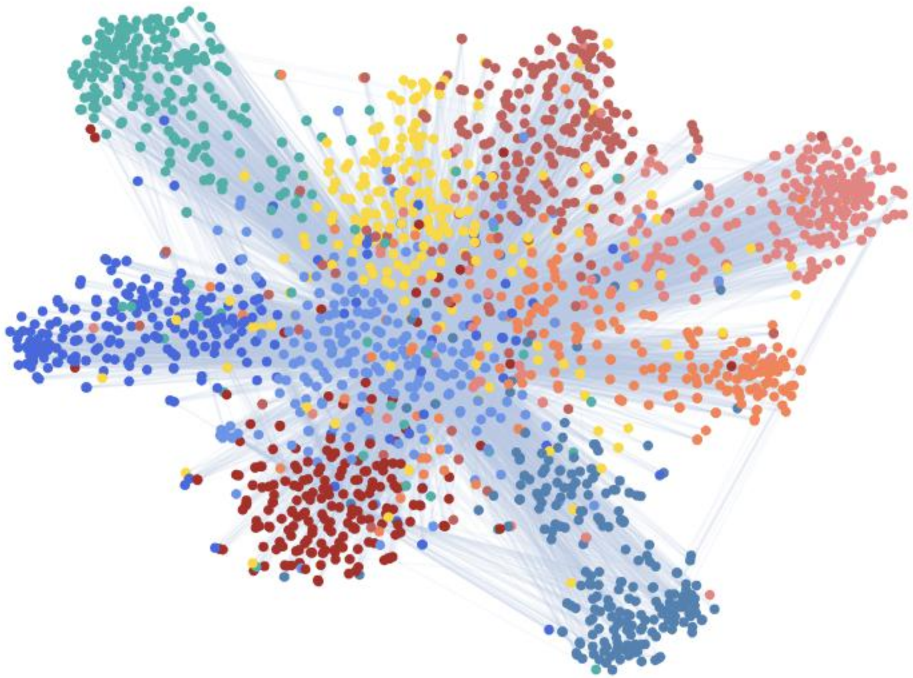


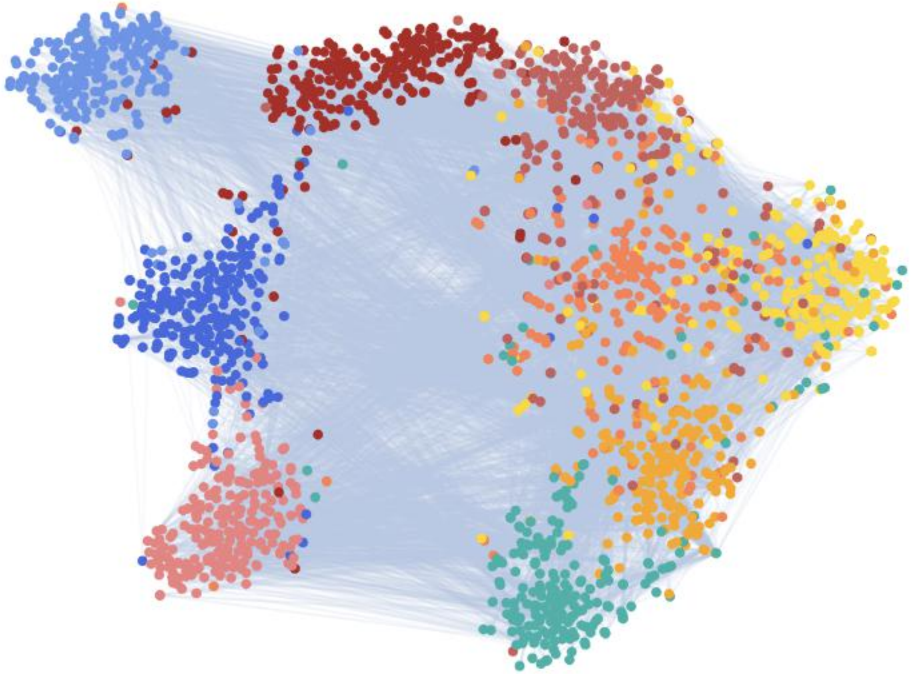


Ablation study on attention functions (i.e., diffusivity parameterization)

Impact of model depth K and step size τ for diffusion iteration

Visualization of Representations

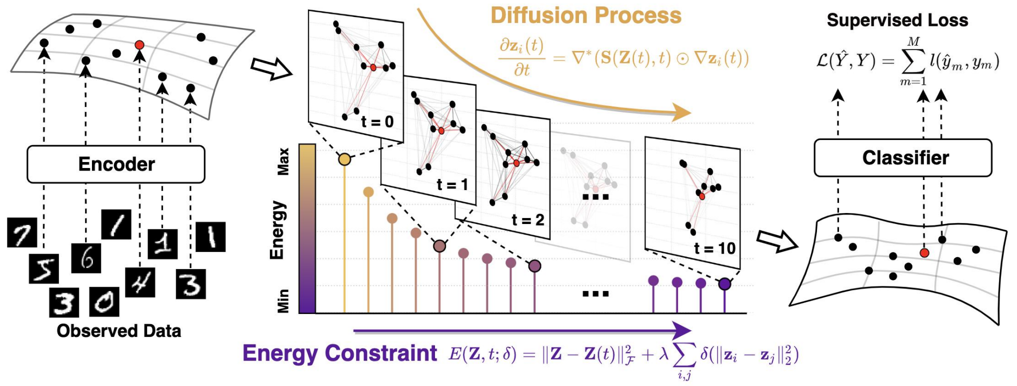




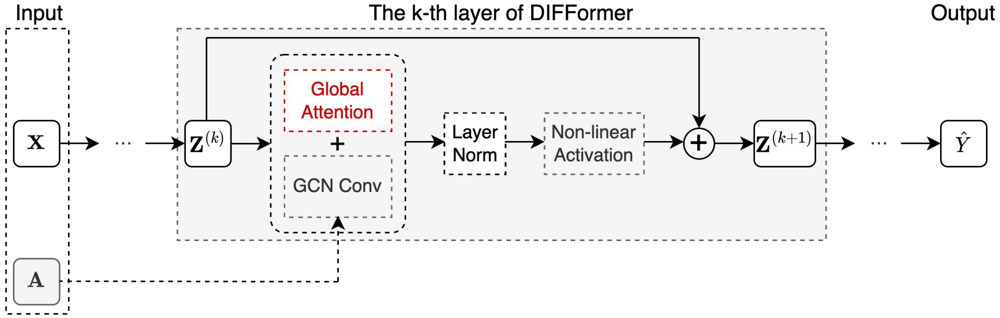
Instance embeddings (colored by different classes) and attention weights (edges with different strengths) on 20News (the left) and STL-10 (the right)

Conclusion

Energy-Constrained Diffusion Model



Diffusion-based Scalable Transformer



Closed-form Estimates for Optimal Diffusivity

Diffusivity Inference: $$\hat{S}_{ij}^{(k)} = \frac{f(\|z_i^{(k)} - z_j^{(k)}\|_2^2)}{\sum_{l=1}^{N} f(\|z_i^{(k)} - z_l^{(k)}\|_2^2)}, \quad 1 \le i, j \le N$$

State Update: $$z_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^{N} \hat{S}_{ij}^{(k)}\right) z_i^{(k)} + \tau \sum_{j=1}^{N} \hat{S}_{ij}^{(k)} z_j^{(k)}, \quad 1 \le i \le N$$

Two instantiations: DIFFormer-s DIFFormer-a

DIFFormer is a general-purpose encoder that accommodates interactions among instances

code *paper* *blog* *tutorial*

Paper: https://arxiv.org/pdf/2301.09474.pdf

Code: https://github.com/qitianwu/DIFFormer

[1] NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification, in NeurIPS 2022

[2] DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, in ICLR 2023

Application Scenarios of DIFFormer

DIFFormer is a general-purpose encoder backbone

DIFFormer can solve predictive tasks with data inter-dependence (i.e., a graph)

Goal: Given node features $\mathbf{X} = \{\mathbf{x}_i\}$ and an input graph $\mathbf{A} = \{\mathbf{a}_{ij}\}$, predict node labels $\hat{Y} = \{\hat{y}_i\}$

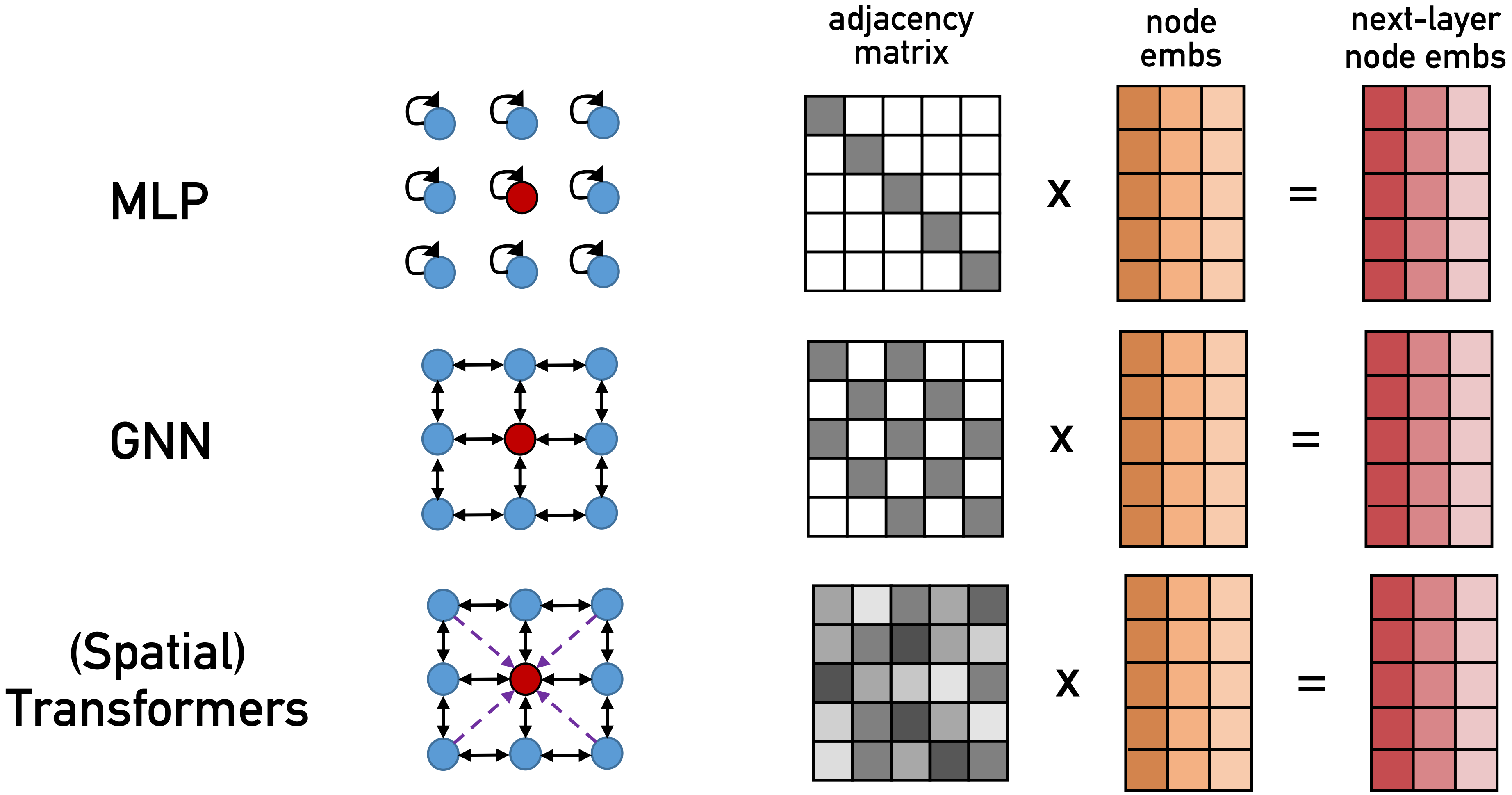
$$\mathbf{Z} = \mathrm{DIFFormer}(\mathbf{X}, \mathbf{A}) \qquad \hat{Y} = \mathrm{FNN}(\mathbf{Z})$$

DIFFormer can model pairwise influence of instances for computing representations

Goal: Training a classifier a dataset of instances $\mathbf{X} = \{\mathbf{x}_i\}$

$$\mathbf{Z} = \mathrm{DIFFormer}(\mathbf{X}, \mathbf{A})$$

DIFFormer can estimate latent interaction graphs over entries in inputs of various forms


adjacency matrix
node embs
next-layer node embs
MLP
x
=
GNN
x
=
(Spatial) Transformers
x
=