Efficient and Effective Implicit Dynamic Graph Neural Network

Yongjian Zhong[1], Hieu Vu[1], Tianbao Yang[2], Bijaya Adhikari[1]

[1]University of Iowa
[2]Texas A&M University

SIGKDD, Barcelona
August 28, 2024

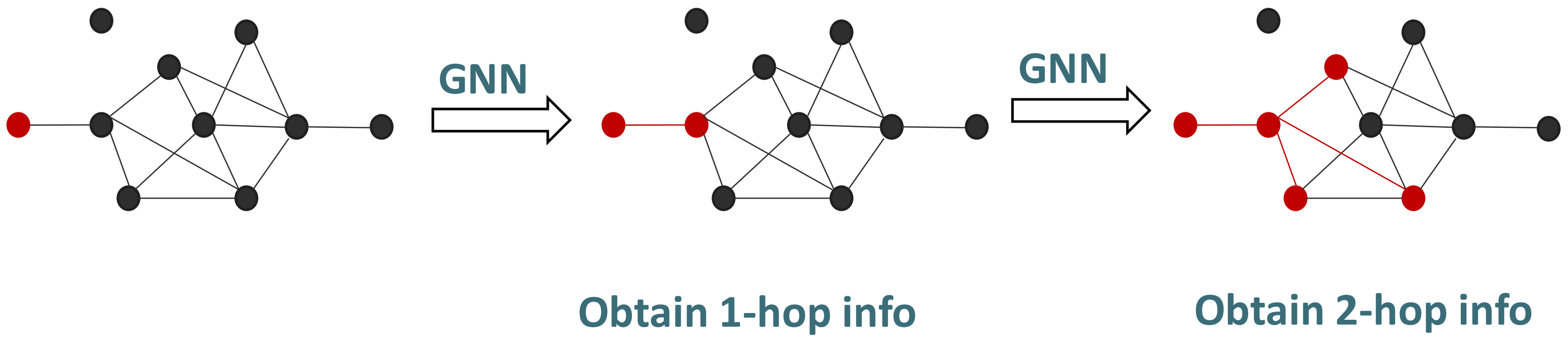
University of Iowa
comp|epi
computational epidemiology research

The University of Iowa

Outline

- **Background & Challenges**
- Problem Formulation
- Our Method
- Experiment
- Conclusion & Future Work

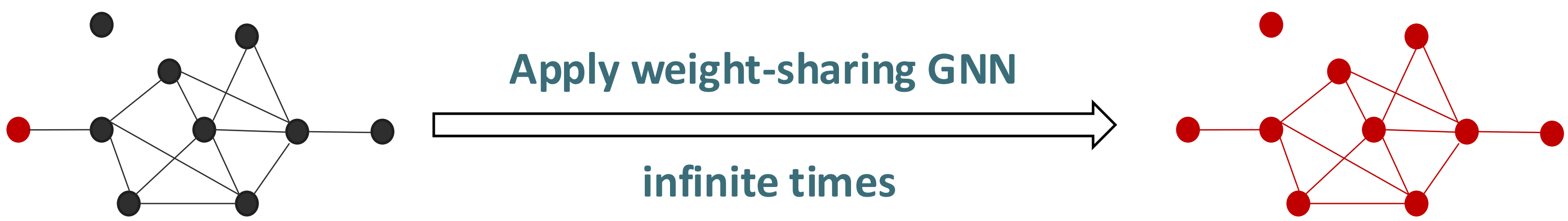
Graph Neural Networks



Stack multiple layers of GNN to reach information which are topologically far

But stacking hurts performance [Li+ 2018, AAAI]

Capturing Long-range Dependency



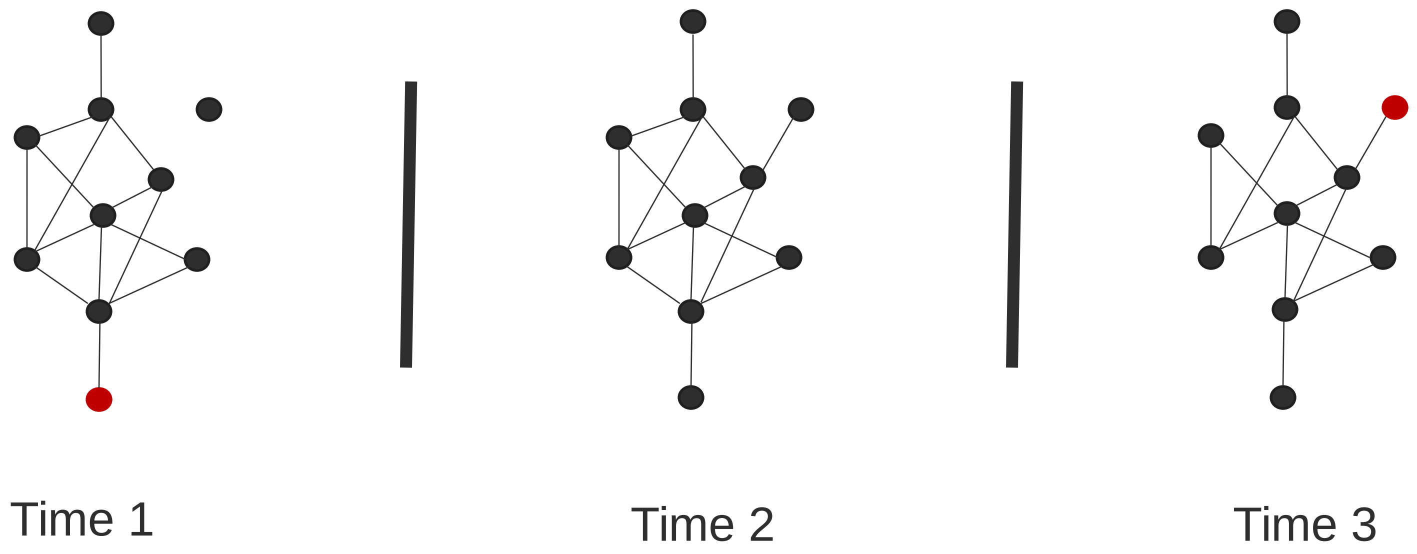
Obtain all info

$$Z^{t+1} = \sigma(WZ^tA + VX) \xrightarrow{t \to \infty} Z^* = \sigma(WZ^*A + VX)$$

Once converged

Key Idea behind Implicit Graph Neural Networks
[Gu+ 2020, NeurIPS; Liu+ 2021, NeurIPS]

Discrete-time Temporal Graphs



- Imagine ● in time 1 and ● in time 3 have a strong dependency.
- Need to fetch information across *Topology* and *Time*.
- Which requires more GCNs stacking.

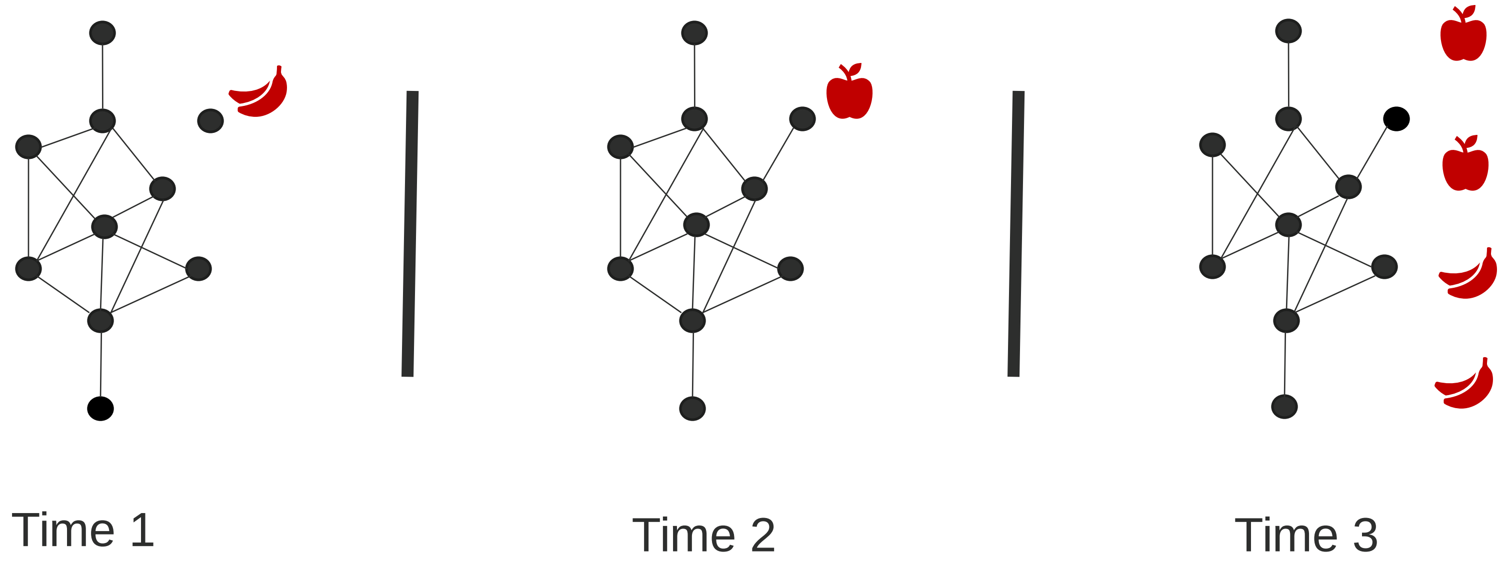
How to capture long-range dependency w/o sacrificing performance?

Outline

- Background & Challenges
- **Problem Formulation**
- Our Method
- Experiment
- Conclusion & Future Work

Problem Setup

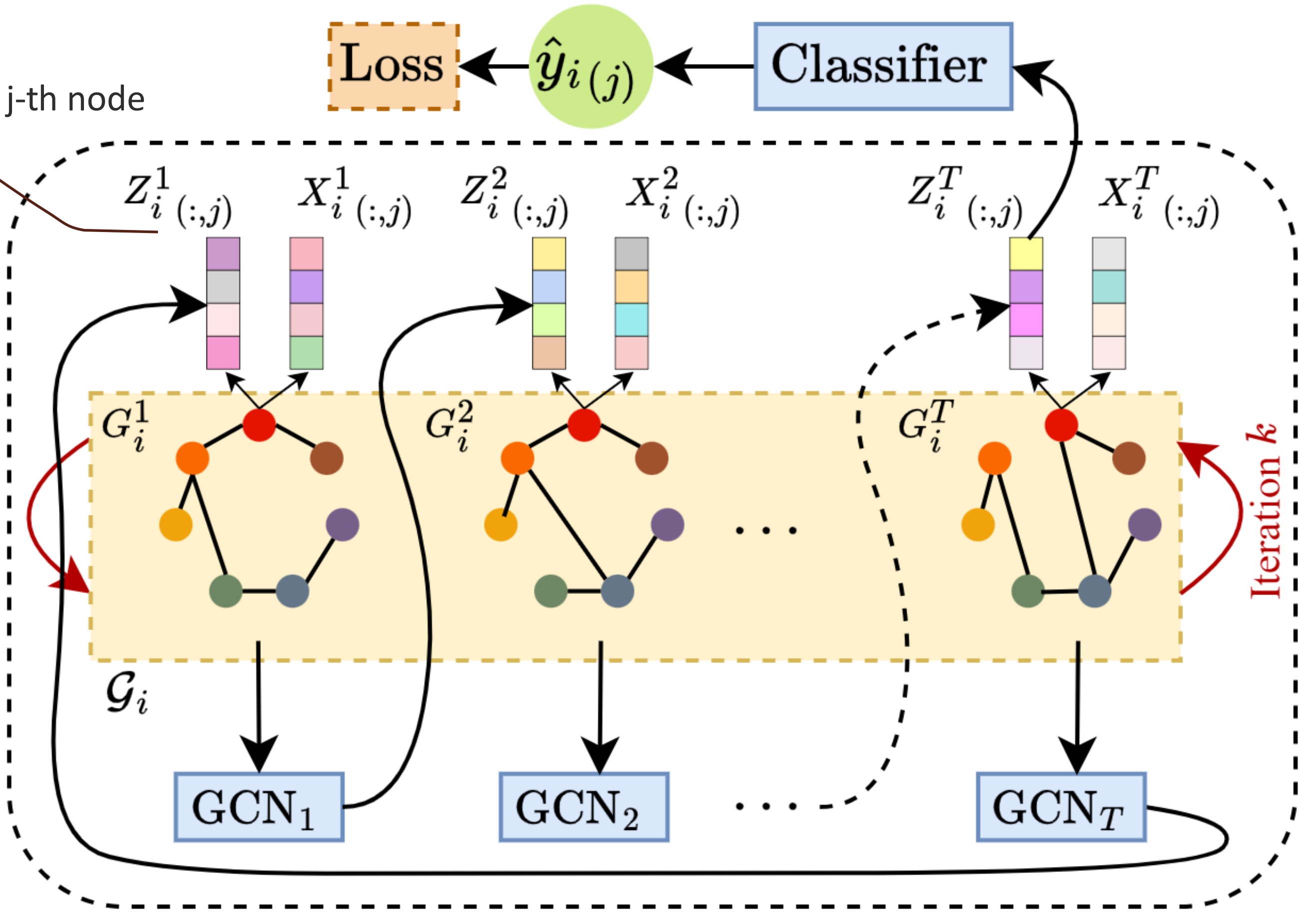
- **Given**
  - Discrete-time Dynamic Networks
  - Labels of nodes
- **Do**
  - Node Classification at the Last Snapshot



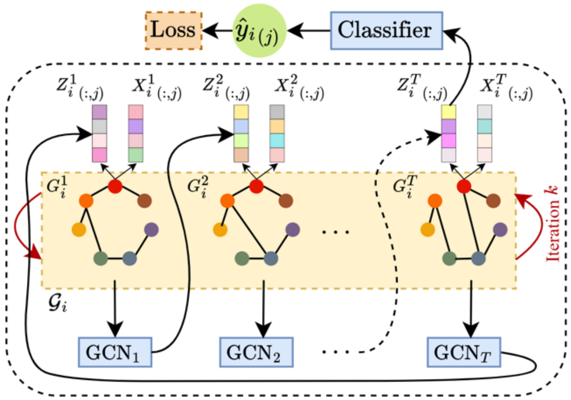
Overview

Our model

i-th dynamic graph, j-th node



Convergent Embeddings



- The convergent embeddings must satisfy...

$$Z^1 = \sigma(W^1 Z^T A^1 + V X^1)$$

$$Z^2 = \sigma(W^2 Z^1 A^2 + V X^2)$$

Element-wise non-expansive function, e.g. ReLU, Sigmoid ...

$$\cdots$$

$$Z^T = \sigma(W^T Z^{T-1} A^T + V X^T)$$

Depends on previous and current time information

GCN

Feature Injection

Let's express this in the matrix form.

The University of Iowa

Convergent Embeddings

The matrix form

Need to be non-expansive

Non-expansive

$$\begin{bmatrix} z^1 \\ z^2 \\ z^3 \\ \vdots \\ z^T \end{bmatrix} = \sigma\left(\begin{bmatrix} 0 & 0 & \cdots & 0 & M^1 \\ M^2 & 0 & \cdots & 0 & 0 \\ 0 & M^3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & M^T & 0 \end{bmatrix}\begin{bmatrix} z^1 \\ z^2 \\ z^3 \\ \vdots \\ z^T \end{bmatrix} + \begin{bmatrix} \mathbf{vec}(VX^1) \\ \mathbf{vec}(VX^2) \\ \mathbf{vec}(VX^3) \\ \vdots \\ \mathbf{vec}(VX^T) \end{bmatrix}\right)$$

where $z = \mathbf{vec}(Z)$**,** $M^i = (A^i)^\top \otimes W^i$**,** $\otimes$ **is the Kronecker product.**

Ensuring Convergence: by the Banach's fixed point theorem, the matrix in red needs to be non-expansive, which can be enforced by ensuring the following

$$\|W^t\|_\infty \|A^t\|_{\text{op}} \leq 1 \qquad t = 1, ..., T$$

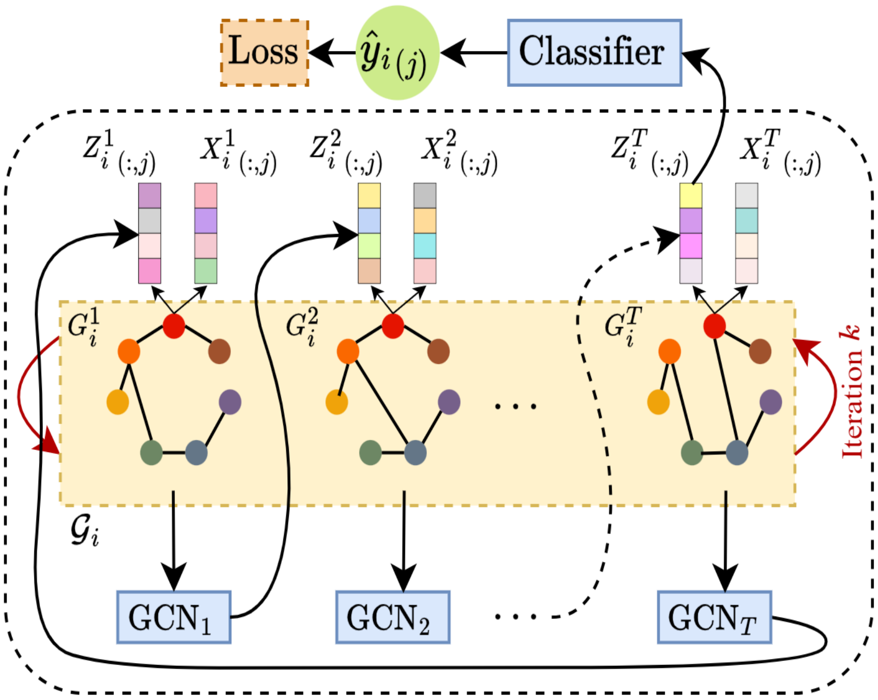
Outline

- Background & Challenges
- Problem Formulation
- **Our Method**
- Experiment
- Conclusion & Future Work

Our Method

☐ Summary of our model

- One layer per snapshot
- Feature injected GCN
- Constrained Weight
- GCN-only framework



More details in the paper

Training

- **The classic way:** differentiate through fixed point.

$t-1$ if $t \in [2, T]$,
T if $t = 1$.

$(Z^{\tau(t)} A^t)^\top$

Element-wise multiplication

Fixed-point embedding of snapshot t.

$$\frac{\partial z^t}{\partial w^a} - \Xi^t \odot \left(\delta_{at} H^t \otimes I + M^t \frac{\partial z^{\tau(t)}}{\partial w^a} \right) = 0$$

Weight of a-th GCN layer.

Indicator function

Each column is
$\mathbf{vec}(\sigma'(W^t Z^{\tau(t)} A^t + V X^t))$

$(A^i)^\top \otimes W^i$

Solving these equations are extremely *inefficient*.

The University of Iowa

Bilevel Optimization

☐ Objective under bilevel optimization perspective

Classification loss function

Classifier

Label of i-th dynamic graph

$$\min_{\theta,\mathcal{W},V} \mathcal{L}(\theta,\mathcal{W},V) = \sum_{i=1}^{N} \ell(f_\theta(z_i^T, \mathbf{y}_i))$$

Implicit condition

$$\text{s.t.} \quad z_i^T = \arg\min_z \|z - \phi(z,\mathcal{W},V;\mathcal{G}_i)\|_2^2$$

Convergence conditions

$$\|W^t\|_\infty \leq \frac{\kappa}{\|A_i^t\|_{op}}, \forall i = 1, \ldots, N, t = 1, \ldots, T$$

where ϕ is a nested function $\quad \phi(z,\mathcal{W},V;\mathcal{G}_i) = \sigma(M_i^T \ldots \sigma(M_i^1 z + \mathbf{vec}(VX_i^1)) \ldots + \mathbf{vec}(VX_i^T))$

> **We propose a bilevel optimization algorithm that solve this objective *efficiently*.**

Algorithm

- **The key is to estimate the Hyper-gradient**

$$\nabla \mathcal{L}(\theta, \omega) = \frac{1}{N} \sum_{i=1}^{N} \nabla \ell_i(z_i^T, \omega) - \nabla^2_{\omega z} g_i(z_i^T, \omega) \boxed{\left[\nabla^2_{zz} g_i(z_i^T, \omega)\right]^{-1}} \nabla_z \ell_i(z_i^T, \omega)$$

The optimal result of lower-level problem

Inverse Hessian

Both are expensive to compute.

Algorithm

More details in the paper

The stochastic bilevel algorithm

Min batch

$$\nabla \mathcal{L}(\theta, \omega) = \frac{1}{N}\sum_{i=1}^{N} \nabla \ell_i(z_i^T, \omega) - \nabla^2_{\omega z} g_i(z_i^T, \omega)\boxed{\left[\nabla^2_{zz} g_i(z_i^T, \omega)\right]^{-1} \nabla_z \ell_i(z_i^T, \omega)}$$

Replaced by inexact result

We use a Hessian-vector product to approximate

Algorithm 1 Bilevel Optimization for IGDNN

Require: $\mathcal{D} = \{(\mathcal{G}_i, \boldsymbol{y}_i)\}_{i=1}^N, \eta_1, \eta_2, \gamma$
Ensure: ω, θ
Randomly initialize z_j^0 and v_j^0 for $j = 1, ..., N$
for $k = 0, 1, ..., K$ **do**
Sample a batch data $\mathcal{B}$
$$\hat{z}_j^{k+1} = \begin{cases} (I - \eta_1)\hat{z}_j^k + \eta_1 \phi(\hat{z}_j^k, \omega^t; \mathcal{G}_i) & j \in \mathcal{B} \\ \hat{z}_j^t & \text{o.w.} \end{cases}$$
$$\hat{v}_j^{k+1} = \begin{cases} (I - \eta_2 \boxed{\nabla^2_{zz} g(\hat{z}_j^k, \omega^k))\hat{v}_j^k} + \eta_2 \nabla_z \ell_j(\hat{z}_j^k, \omega^k) & j \in \mathcal{B} \\ \hat{v}_j^t & \text{o.w.} \end{cases}$$
Update gradient estimator
$$\Delta^{k+1} = \frac{1}{|\mathcal{B}|}\sum_{j \in \mathcal{B}} \left[\nabla_\omega \ell_j(\hat{z}_j^k, \omega^k) - \boxed{\nabla^2_{\omega z} g_j(\hat{z}_j^k, \omega^k)\hat{v}_j^k}\right]$$
$$m^{k+1} = (1-\gamma)m^k + \gamma \Delta^{k+1}$$
$$\omega^{k+1} = \Pi_\Omega\left(\omega^k - \eta_0 m^{k+1}\right)$$
end for

Inexact result using only one fixed point iteration

Approximation

Note: use Hessian-vector product to compute to avoid expensive computation [Hu+ NeurIPS, 2022].

Content

- Background & Challenges
- Prize-collecting Steiner Tree
- Our Method
- **Experiment**
- Conclusion & Future Work

Experiment

☐ Data

	N	$\lvert V \rvert$	$\min \lvert E \rvert$	$\max \lvert E \rvert$	T	d	y
Brain10	1	5000	154094	167944	12	20	10
DBLP5	1	6606	2912	5002	10	100	5
Reddit4	1	8291	12886	56098	10	20	4
PeMS04	16980	307	680	680	12	5	3
PeMS08	17844	170	548	548	12	5	3
England-COVID	54	129	836	2158	7	1	1

☐ Tasks

- Node-level *classification* and *regression*
- For regression, there are *transductive* and *inductive* cases.

☐ Evaluation

- *AUROC* for classification
- *MAPE* for regression

The University of Iowa

Classification

Sparse graph. Long-range dependency is not a major bottleneck.

Model	Classification Brain10	 DBLP5	 Reddit4
EvolveGCN-O	0.58±0.10	0.639±0.207	0.513±0.008
EvolveGCN-H	0.60±0.11	0.510±0.013	0.508±0.008
GCN-GRU	0.87±0.07	0.878±0.017	0.513±0.010
DySAT-H	0.77±0.07	**0.917±0.007**	0.508±0.003
GCRN-M2	0.77±0.04	0.894±0.009	<u>0.546±0.020</u>
DCRNN	0.84±0.02	0.904±0.013	0.535±0.007
TGAT	0.80±0.03	0.895±0.003	0.510±0.011
TGN	<u>0.91±0.03</u>	0.887±0.004	0.521±0.010
GRU-GCN	<u>0.91±0.03</u>	0.906±0.008	0.525±0.006
IDGNN	**0.94±0.01**	<u>0.907±0.005</u>	**0.556±0.017**

Our method outperforms other baselines except on DBLP5

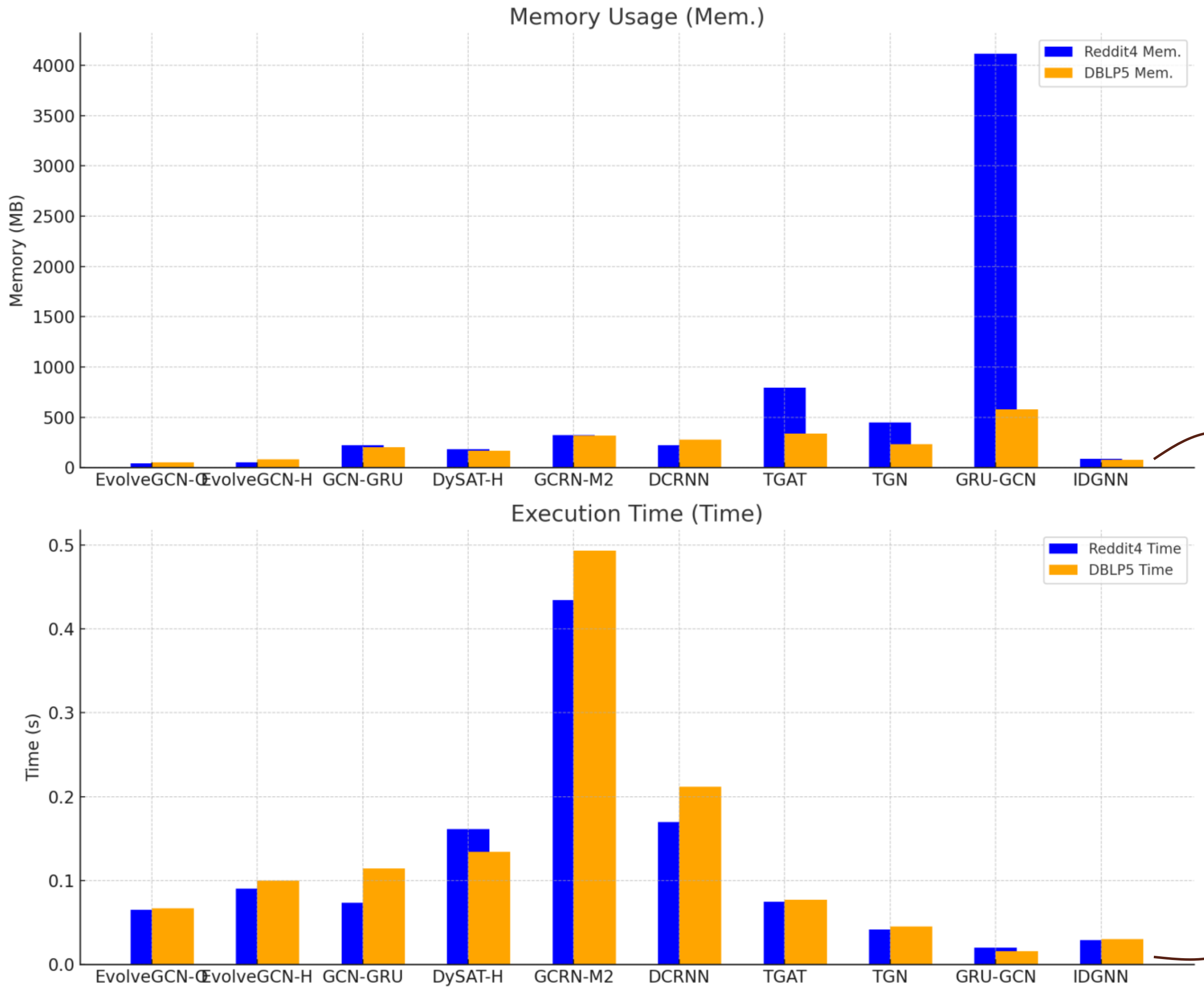
Regression

Model	Regression					
	England-COVID		PeMS04		PeMS08	
	Trans.	Induc.	Trans.	Induc.	Trans.	Induc.
EvolveGCN-O	4.07±0.73	3.88±0.47	3.20±0.25	2.61±0.42	2.65±0.12	2.40±0.27
EvolveGCN-H	4.14±1.14	3.50±0.42	3.34±0.14	2.84±0.31	2.81±0.28	2.81±0.23
GCN-GRU	3.56±0.26	2.97±0.34	1.60±0.14	1.28±0.04	1.40±0.26	1.07±0.03
DySAT-H	3.67±0.15	3.32±0.76	1.86±0.08	1.58±0.08	1.49±0.08	1.34±0.03
GCRN-M2	3.85±0.39	3.37±0.27	1.70±0.20	1.20±0.06	1.30±0.17	1.07±0.03
DCRNN	3.58±0.53	3.09±0.24	1.67±0.19	1.27±0.06	1.32±0.19	1.07±0.03
TGAT	5.44±0.46	5.13±0.26	3.11±0.50	2.25±0.27	2.66±0.27	2.34±0.19
TGN	4.15±0.81	3.17±0.23	1.79±0.21	1.19±0.07	1.49±0.26	0.99±0.06
GRU-GCN	3.41±0.28	**2.87±0.19**	1.61±0.35	1.13±0.05	1.27±0.21	0.89±0.07
IDGNN	**2.65±0.25**	3.05±0.25	**0.53±0.05**	**0.63±0.04**	**0.45±0.11**	**0.50±0.05**

Our method outperforms other baselines

The University of Iowa

Runtime & Memory



As low as EvolveGCN

Our method is fast and efficient

Second fast

THE UNIVERSITY OF IOWA

SGD vs. Bilevel

1600x speed-up

	Runtime (s/win)		Performance	
	SGD	Bilevel	SGD	Bilevel
Brain10	624	**0.390**	94.7	**94.9**
PeMS04	0.72	**0.049**	**0.63**	**0.63**
PeMS08	0.29	**0.046**	0.56	**0.50**
England-COVID	0.092	**0.030**	**2.97**	3.05

Similar performance

Bilevel algorithm achieves similar performance as exact SGD, but is much faster

The University of Iowa

Outline

- Background & Challenges
- Problem Formulation
- Our Method
- Experiment
- **Conclusion & Future Work**

Conclusion & Future Work

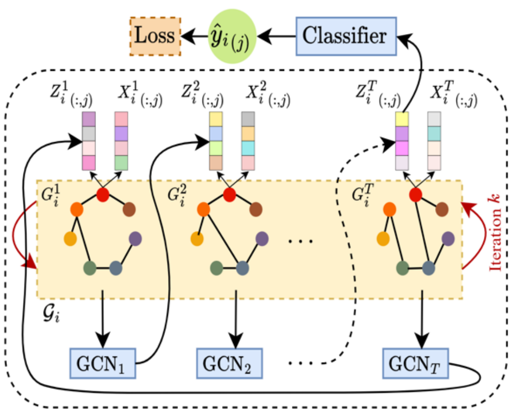
- We proposed a novel implicit graph neural network for dynamic graphs. As far as we know, this is the first implicit model on dynamic graphs.

- We would like to develop an optimization algorithm with convergence guarantee for the bilevel algorithm.

- Global optimization framework for implicit models.

The University of Iowa

Questions?



Fixed-point embedding of snapshot t.

Weight of a-th GCN layer.

Element-wise multiplication

$(Z^{\tau(t)} A^t)^\top$

$t-1$ if $t \in [2, T]$,
T if $t = 1$.

$$\frac{\partial z^t}{\partial w^a} - \Xi^t \odot \left(\delta_{at} H^t \otimes I + M^t \frac{\partial z^{\tau(t)}}{\partial w^a} \right) = 0$$

Indicator function

Each column is
$\mathbf{vec}(\sigma'(W^t Z^{\tau(t)} A^t + V X^t))$

$(A^i)^\top \otimes W^i$