A APPENDIX

A EXPERIMENT

A.1 DATASETS

Brain dataset is derived from a real-world fMRI brain scans dataset [1]. In this dataset, nodes represent brain tissues, and edges capture nodes' activation time similarity in the examined period. The node attributes are generated from the fMRI signals Xu et al. (2019). Given the temporal graph, our goal is to predict the functionality of each node, which is one of the **10** categories.

DBLP is an extracted co-author network from DBLP website [2], where nodes are authors and edges represent co-authorship relationships. The extracted authors are from **5** research areas Xu et al. (2019), serving as our class labels. Note attributes are word2vec representations of titles and abstracts from related papers.

Reddit dataset is extracted from a popular online platform Reddit [3] Hamilton et al. (2018), where nodes represent posts, and edges are constructed between nodes having similar keywords. The node attributes are word2vec representations of the comments of a post Xu et al. (2019), and node labels correspond to one of the **4** communities or "subreddits" to which the post belongs.

PeMS04 & PeMS08 These two datasets represent traffic sensor networks in two distinct districts of California during various months. In this context, each node symbolizes a road sensor, while the edge weights denote the geographical distance between two sensors (with uniform weights across all snapshots). Every sensor captures average traffic statistics such as flow, occupancy, and speed over a 5-minute interval. The datasets utilized in this study are identical to those in Gao & Ribeiro (2022), where we exclusively forecast attribute values for a single future snapshot.

England-COVID This temporal graph dataset is taken from Gao & Ribeiro (2022), which is created by aggregating information from the mobility log released by Facebook under Data For Good program [4] for England Panagopoulos et al. (2021). The nodes are cities, and the edges are transportation statistics between them. Given a temporal graph of length 7, which represents the past week, we need to predict the infection rate of the next day.

A.2 EXPERIMENT SETUP

We follow the procedure from Gao & Ribeiro (2022) and utilize the provided code [5] as the code base to compare all the baselines and our method. Specifically, we first split the dataset into three portions with ratios 70%-10%-20% for training, validation, and testing, respectively. Splitting is based on nodes for transductive tasks and time for inductive tasks. We then normalize node attributes and edge attributes with the 0-1 normalization method. We train on the training portion, find the best hyperparameters using the validation set, and report the performance on the test set. We also use ROCAUC score to evaluate classification tasks and mean average percentage error (MAPE) for regression tasks.

A.3 HYPERPARAMETER

For detailed baselines' architecture, please refer to Gao & Ribeiro (2022). Notice that, for all the methods and all task, we fixed the embedding size as 16, and we searched the learning rates from 0.1, 0.01, and 0.001. For Brain10, we observed that our method converged slowly, then we let it run for 1000 epochs. For DBLP5 and Reddit4, we let all the methods run 500 epochs for 5 times for each learning rate and report the performance on the test set where the performance of the validation set is the best. For regression datasets, we run 100 epochs for England-COVID and 10 epochs for PeMS04/PeMS08. The hyperparameter our model $\eta_1 \in \{0.5, 0.7, 0.9, 1\}, \eta_2 \in \{0.001, 0.01, 0.1\}$.

[1] https://tinyurl.com/y4hhw8ro

[2] https://dblp.org/

[3] https://www.reddit.com/

[4] https://dataforgood.fb.com/tools/disease-prevention-maps/

[5] https://www.dropbox.com/s/gqtzpngj5vaah9s/ICML2022-Code-Submit.zip

LEMMA 1 & THEOREM 2

Lemma 3 *If $\|.\|$ is the matrix operator norm on $\mathbb{R}^{n\times n}$, then, for any matrix $A \in \mathbb{R}^{n\times n}$,*

$$\lambda_{PF}(A) \leq \|A\|$$

Proof 1 *Let λ be the eigenvalue of A, and let $x \neq 0$ be the corresponding eigenvector. From $Ax = \lambda x$, we have*

$$AX = \lambda X$$

where each column in X is x. Further, we have

$$|\lambda|\|X\| = \|\lambda X\| = \|AX\| \leq \|A\|\|X\|$$

Since $\|X\| > 0$, taking the maximal λ gives the result.

Lemma 1 *The equilibrium equation $z = \sigma(Mz+b)$ has a unique fixed point solution if $|||M|||_{op} < 1$, where $\|.\|_{op}$ is the operator norm, and $\sigma(\cdot)$ is an element-wise non-expansive function.*

Theorem 2 *Let σ be an element-wise non-expansive non-linear function. The coupled equilibrium equations satisfy the well-posedness condition, namely $|||W|||_{op}\|A\|_{op} < 1$. There exists rescale coupled equilibrium equations, which satisfy $\|W\|_{\infty}\|A\|_{op} < 1$, and the solutions of these two equations are equivalent.*

After applying Lemma 3, Lemma 1 and Theorem 2 are directly from Gu et al. (2020). For Lemma 1, they use a constructive method to prove: generate a sequence via fixed-point iteration, and then this sequence converges to a unique fixed point due to the contractive property. For Theorem 2, they show there exists the corresponding diagonal matrix to rescale the result.

PROOF OF THEOREM 1

Proof 2 *By Lemma. 1, the well-posedness requirement for Formula. (4) is $|||\mathcal{M}\|_{op}| \leq 1$. Since Formula. (4) and (3) are equivalent, the well-posedness requirement for Formula. (3) is also $|||\mathcal{M}|||_{op} < 1$.*

Let $M := \begin{bmatrix} 0 & M_1 \\ \hat{M} & 0 \end{bmatrix}$ *where* $\hat{M} := \begin{bmatrix} M_2 & \dots & \\ 0 & \ddots & 0 \\ 0 & \dots & M_t \end{bmatrix}$. *Let* $\tilde{M} := \begin{bmatrix} \hat{M} & 0 \\ 0 & M_1 \end{bmatrix}$. *Then*

$$\begin{aligned} |||M|||_{op} = |||\tilde{M}|\begin{bmatrix} 0 & I_m \\ I_n & 0 \end{bmatrix}\|_{op} &\leq |||\tilde{M}|||_{op} \cdot \|\begin{bmatrix} 0 & I_m \\ I_n & 0 \end{bmatrix}\|_{op} \\ &= |||\tilde{M}|||_{op} = \max\{|||M_1|||_{op}, ..., |||M_t|||_{op}\} \end{aligned}$$

This means if all subsystems satisfy the largest eigenvalue constraint, then the coupled equilibrium equation has a fixed point solution by Lemma 1.

PROOF OF LEMMA 2

Proof 3 *Recall that we have the following formula*

$$z_j = \sigma(M_{j+T}\sigma(M_{j+T-1}\cdots\sigma(M_{j+1}z_i + \textbf{\textit{vec}}(VX_{j+1}))\cdots + \textbf{\textit{vec}}(VX_{j+T-1})) + \textbf{\textit{vec}}(VX_{j+T}))$$

$\Leftarrow$: Suppose formula (9) converges to $z_1, ..., z_T$, and for any arbitrary j, formula (3) converges to $\hat{z}_j$, where $z_j \neq \hat{z}_j$. If we substitute $\hat{z}_j$ into formula (9), we obtain a sequence $\{\hat{z}_j, ..., \hat{z}_T, \hat{z}_1, ..., \hat{z}j-1\}$. This sequence also serves as a fixed-point of formula (9), given that $\hat{z}_j$ is the fixed-point of formula (3) and formula (3) is a composition of different levels of GCN. According to our assumption, formula (9) converges to a unique solution, implying that these two expressions share the same fixed point. $\Rightarrow$ part follows a similar line of reasoning.

B NAIVE GRADIENT DESCENT

Recall that the parameters satisfy the following equations

$$\begin{aligned} F_2(\boldsymbol{z}, W, V) &= z_2 - \sigma(M_2 z_1 + \mathbf{vec}(VX_2)) = 0 \\ &\vdots \\ F_T(\boldsymbol{z}, W, V) &= z_T - \sigma(M_T z_{T-1} + \mathbf{vec}(VX_T)) = 0 \\ F_1(\boldsymbol{z}, W, V) &= z_1 - \sigma(M_1 z_t + \mathbf{vec}(VX_1)) = 0 \end{aligned}$$

In total, there are tnd equations (nd components inside each z). We can calculate the gradient via implicit differentiation. For any equation $Z_{ij}^k = \sigma(\sum_l \sum_n W_{il}^k Z_{ln}^{k-1} A_{nj}^k + \sum_l V_{il} X_{lj}^k)$, taking derivative with respect to W_{bc}^a. Let W_{bc}^a denote the b-th row c-th column of the weight matrix at a-th layer.

$$\frac{\partial Z_{ij}^k}{\partial W_{bc}^a} - \sigma_k'(i,j) \left(\sum_n \delta_{ak} \delta_{bi} Z_{cn}^{k-1} A_{nj}^k + \sum_l \sum_n W_{il}^k \frac{\partial Z_{ln}^{k-1}}{\partial W_{bc}^a} A_{nj}^k \right) = 0$$

Let $H^k := (Z^{k-1} A^k)^T$ and e_b be the a column vector that has value 1 at b-th entry but 0 at others.

$$\frac{\partial \boldsymbol{z}_k}{\partial W_{bc}^a} - \mathbf{vec}(\sigma_k') \odot \left(\delta_{ak} e_b \otimes H_{\cdot c}^k + M^k \frac{\partial \boldsymbol{z}_{k-1}}{\partial W_{bc}^a} \right) = 0$$

Further,

$$\frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{w}_a} - \Sigma_k' \odot \left(\delta_{ak} H^k \otimes I + M^k \frac{\partial \boldsymbol{z}_{k-1}}{\partial \boldsymbol{w}_a} \right) = 0 \tag{9}$$

Similarly, we can compute the gradient of Z_{ij}^k w.r.t. Ω_{bc}^a.

$$\frac{\partial Z_{ij}^k}{\partial V_{bc}} - \sigma_k'(i,j) \left(\delta_{bi} X_{cj}^k + W_{il}^k \frac{\partial Z_{ln}^{k-1}}{\partial W_{bc}^a} A_{nj}^k \right) = 0$$

$$\frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{v}} - \Sigma_k' \odot \left((X^k)^T \otimes I + M^k \frac{\partial \boldsymbol{z}_{k-1}}{\partial \boldsymbol{v}} \right) = 0 \tag{10}$$

C ABLATION STUDY

In this section, we will delve into the various configurations of our model. Drawing from the properties mentioned earlier, our model can be represented by the formula (2). We have made the deliberate decision to assign different weights (W) to each timestamp while maintaining weight-tied for V. Alternatively, the model comprises T layers of GCN and one linear layer. The linear layer serves to aggregate static information, while the GCNs handle dynamic information. To validate our architecture choice, we conducted a thorough comparison of our model against other configurations:

- Tie both: model consists of one layer of GCN and one linear layer.
- Tie W: model consists of one layer of GCN and T linear layer.
- Untied: model consists of T layer of GCN and T linear layer.

The results are presented in Table 4. As observed, the tie-both model exhibits the poorest performance. We believe this is due to the limited number of free variables available for tuning, which makes the training process challenging. In our approach and the tie-W method, we achieve very similar results. Our model utilizes T layers of GCN for dynamic information and one linear layer for static information, while the tie-W method employs one GCN and T linear layers. The similarity in results suggests that these models may be equivalent. The untied method achieves the best result, although the improvement is negligible. However, it doubles the parameter size, resulting in significant computational overhead. Hence, we opt for the current configuration, as it delivers good performance while minimizing parameter size, especially since the number of attributes may exceed the hidden dimension.

Table 4: Evaluating different variants of our model. Presenting AUC results on Brain10 datasets

	AUC
Tie both	95.29±0.26
IDGNN	95.87±0.13
Tie W	95.87±0.08
Untied	95.90±0.08

Layers	GCN-GRU	T-GCN	IDGNN
8	0.6217	0.9934	1.1113
16	0.5204	0.8719	1.0982
32	0.0077	0.7176	1.0019

Table 5: Smoothness of embeddings. (The larger the better)

D EMBEDDING VISUALIZATION

In this section, we explore an interesting aspect of our method that can provide empirical insights into its ability to mitigate oversmoothing. We conduct experiments on a synthetic dataset that bears resemblance to toy datasets.

The dataset comprises 10 snapshots, with each snapshot representing a clique of 10 nodes. Each node is associated with 10 attributes. The nodes fall into two distinct classes, but we deliberately conceal the label information in the attributes of the first snapshot. Specifically, the first two dimensions of the attributes represent the one-hot encoding of the labels, while the remaining dimensions are set to zero. Additionally, we assign unique IDs to the nodes in sequential order. Nodes with IDs ranging from 0 to 4 belong to class 0, while those with IDs ranging from 5 to 9 belong to class 1. To assess the effectiveness of our method, we visually compare the embedding results with those obtained from TGCN.

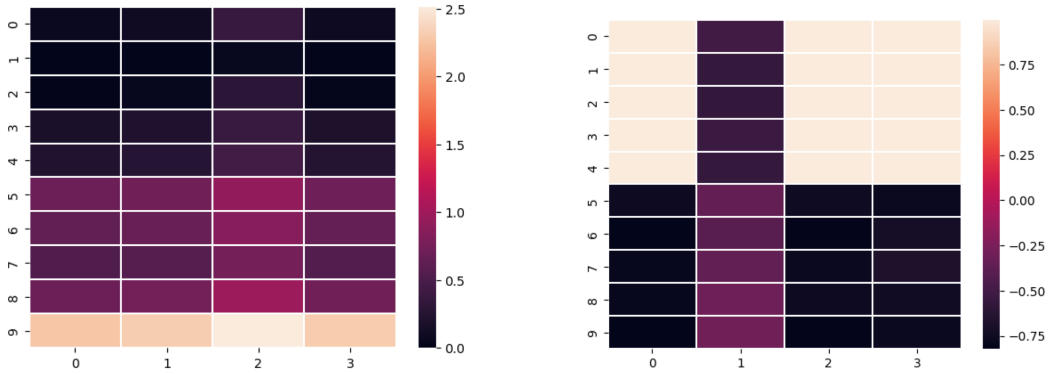
Upon examining the visualizations, we observe that our model's embeddings exhibit gradual changes, whereas TGCN's embeddings remain consistent for nodes belonging to the same class. From a node-centric perspective, TGCN's embeddings seem reasonable. Nodes of the same class possess identical features and exhibit the same topological structure. Therefore, it is logical for them to share a common embedding. However, our embeddings tend to differentiate each individual node. We believe that this characteristic plays a role in mitigating the oversmoothing problem within our model.

Furthermore, we conduct an additional experiment on quantitatively evaluating our model's ability to tackle over-smoothing. This experiment is conducted on the binary toy dataset: the toy data we constructed consists of a dynamic graph with a maximum of 64 snapshots (adapt to layers), with each snapshot being a clique of 10 nodes. Each node has 10 associated attributes. The task is binary classification where each node's class information is hidden in its first time-stamp's attributes. Attributes of other time stamps are randomly sampled from the normal distribution. All methods are trained with a maximum of 2000 epochs and a learning rate of 0.001. At last, we evaluate their smoothness by Mean Average Distance (MAD). Results are summarized as follows

E COMPLEXITY

For an arbitrary temporal graph, we denote the total number of snapshots as T, the total number of nodes as n, the number of edges of time t as E_t, and E_{agg} as the number of aggregated edges, which satisfies $E_{agg} \ll \sum_t E_t$ or $E_{agg} \approx \sum_t E_t$. Some basic complexities: GRU, self-attention, and LSTM have complexity $O(T^2)$ if the input sequence has length T; GCN and SpectralGCN have complexity $O(nd^2 + Ed)$; GAT has complexity $O(nd + Ed^2)$. The complexities of all models are summarized as follows

Based on the complexity results, IDGNN is faster than EvolveGCN-O, EvolveGCN-H, GCN-GRU, GCRN-M2, and DCRNN due to the absence of RNN in our model. TGN and TGAT are the slowest since they have Ed^2 terms. For GRU-GCN, its runtime depends on the magnitude of E_t



(a) Our model (b) TGCN

Figure 2: The embedding visualization of our method and TGCN

EvolveGCN-O	$O(Td^2 + Tnd^2 + \sum_t E_t d)$
EvolveGCN-H	$O(Td^2 + Tnd^2 + \sum_t E_t d)$
GCN-GRU	$O(Td^2 + Tnd^2 + \sum_t E_t d)$
DySAT	$O(Td^2 + Tnd + \sum_t E_t d^2)$
GCRN-M2	$O(Td^2 + Tnd^2 + \sum_t E_t d)$
DCRNN	$O(Td^2 + Tnd^2 + \sum_t E_t d)$
TGAT	$O(Tnd + \sum_t E_t d^2)$
TGN	$O(E_{agg} d + Tnd^2 + \sum_t E_t d^2)$
GRU-GCN	$O(E_{agg} d + Tnd^2 + TE_{agg} d^2)$
IDGNN	$O(Tnd^2 + \sum_t E_t d)$

Table 6: Summary of all models' complexities

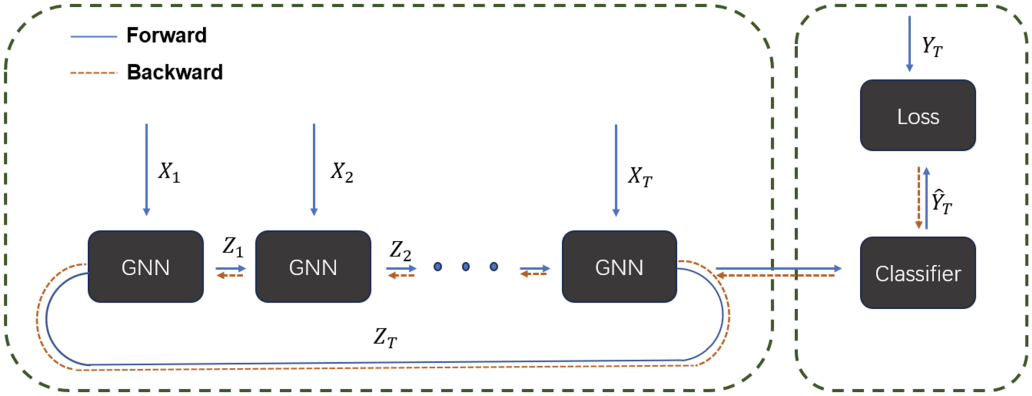


Figure 3: Model overview

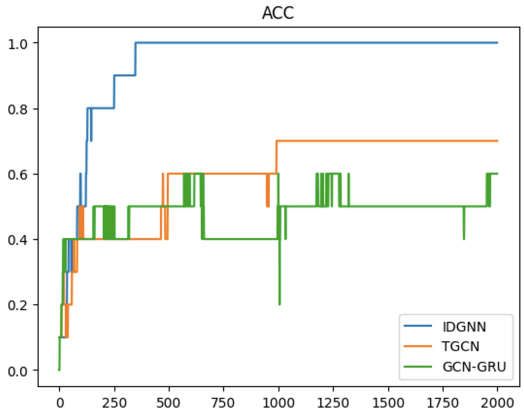


Figure 4: Additional result.

F Model Overview

Here, we draw a model diagram for our training algorithm. The blue line denotes the forward process, and the orange dashed line denotes the backward process.

G Hessian Vector Product

To compute the product of Hassian and a vector: Hv, and $H = \frac{\partial^2 f}{\partial x^2}$. We compute the product by $Hv = \frac{\partial (\frac{\partial f}{\partial x})^T v}{\partial x}$. In this way, we are not explicitly computing the Hessian. [6]

H Additional Result on Toy Data

This synthetic experiment shifts label information from time stamp 1 to time stamp 5. This adjustment ensures uniform difficulty in utilizing label information across all models. Implementing this change postpones our method towards achieving 100% accuracy by approximately 50 epochs. However, even with this modification, the baselines still struggle to fit the data.

[6] https://jax.readthedocs.io/en/latest/notebooks/autodiff_cookbook.html