Structural Embedding Pre-Training for Deep Temporal Graph Learning

Meng Liu, Wenxuan Tu, Ke Liang, Xinwang Liu*
National University of Defense Technology
Changsha, China
mengliuedu@163.com, xinwangliu@nudt.edu.cn

***Abstract*—Thanks to the fact that many real-world data can be modeled as graph-structured data, graph deep learning is gradually attracting close attention from researchers. As an integral component of graph learning, temporal graph learning abandons the data format of adjacency matrices and instead utilizes interaction sequences to record and observe real-time changes in nodes. However, temporal graph learning both benefits from and is constrained by this approach. The data format of interaction sequences often leads temporal graph learning methods to focus only on the closest neighborhoods, making it difficult to capture high-order structural information and resulting in noticeable information loss. To ensure that temporal graph learning methods can consider both high-order structural information and maintain their flexibility, we propose SET, a temporal graph method which introduces Structural Embedding pre-training to enhance Temporal graph learning. Specifically, we achieve this by introducing classical methods to pre-train the data, generating node initialization embeddings that focus on high-order structural information. Furthermore, we constrain the model to optimize not only based on these embeddings but also to approach them as signals for data augmentation. By incorporating structurally embedded features through pre-training, we are able to obtain a broader receptive field without compromising model efficiency. We conducted experiments on several datasets, the experimental results validate the performance of our proposed method SET.**

***Index Terms*—graph deep learning, temporal graph, pre-training**

I. Introduction

Graph data is frequently encountered in various real-world contexts, including academic collaboration, product consumption, financial transactions, city transportation, and more. In the realm of graph data, individuals are commonly represented as nodes, while the connections between individuals are depicted as edges. Machine learning methods designed for graph data often produce representation vectors, also referred to as embeddings, for nodes. These embeddings find utility in a wide range of graph learning tasks, such as link prediction, node clustering, and classification. Consequently, graph deep learning plays an active role in numerous real-world scenarios, including social recommendation [1], multi-modal fusion [2]–[4], knowledge graph [5]–[7], traffic forecasting [8], multi-view clustering [9], [10], fake news detection [11], social security [12], [13], and image analysis [14], [15], among others.

Traditional graph learning methods typically rely on static graphs, where the graph structure remains unchanged during the training process. However, real-world graph data often contains valuable temporal information, and disregarding this information may result in inadequate learning outcomes. Consequently, researchers have introduced the concept of temporal graphs, which capture the temporal evolution in real-time.

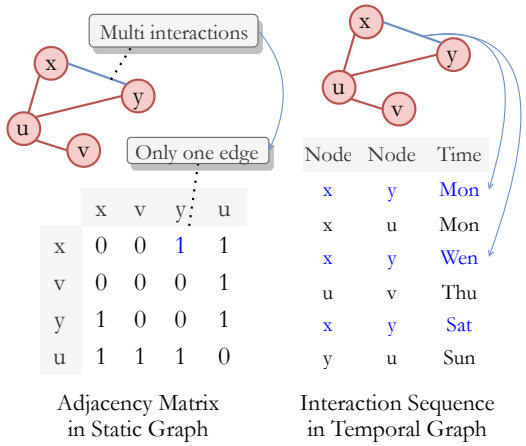


Fig. 1. Difference between static graphs and temporal graphs. Compared to the adjacency matrix, which can store only one edge, the interaction sequence can represent multiple interactions between two nodes.

Temporal graph learning departs from the traditional method of using adjacency matrices to represent graph structure and instead employs interaction sequence (adjacency list) for data storage and retrieval. As shown in Figure 1, Compared with the adjacency matrix-based static graph, the interaction sequence-based temporal graph clearly records each interaction of nodes.

By leveraging interaction sequence, temporal graph methods become more adaptable and lightweight. However, this pattern also restricts their ability to access higher-order structural information. Existing temporal models primarily concentrate on closely connected first-order neighborhood information, neglecting higher-dimensional global structural information.

Although it has been demonstrated that first-order neighborhood information holds significant importance, it is crucial not to overlook the broader perspective offered by higher-order global structural information. Current temporal graph learning methods often overlook global structural information due to the challenges posed by obtaining higher-order sub-graphs involving multiple nodes within the batch training model

based on interaction sequences. To mitigate the computational complexity associated with acquiring higher-order structural information, we propose SET, a novel temporal graph learning method, which enhances temporal graph learning through data augmentation via pre-training on structural embeddings.

In our proposed method, we leverage the Graph AutoEncoder (GAE) [16] as the pre-training model to generate initial node embeddings. GAE is a well-established static graph learning model that effectively captures global structural information. By employing GAE, we obtain initial embeddings for all nodes, which serve as the starting features. During training, we encourage the node embeddings to align with these features.

Furthermore, we enhance the efficiency of deep learning model training by incorporating power constraints into the loss function. This ensures that dimensions of the embeddings that deviate from the supervised signal receive higher optimization weights. As a result, our proposed method, SET, outperforms multiple SOTA baseline methods on various temporal graph datasets, achieving superior performance.

Our contributions can be summarized as follows:

(1) We propose a temporal graph learning method called SET, which introduces structural embedding pre-training to enhance temporal graph learning.

(2) To enhance the effectiveness of deep learning training, we impose constraints on the loss function that focus more on optimizing those dimensions with large variance.

(3) We compare SET with multiple baseline methods on different datasets, and the results validate the effectiveness and performance of our method.

II. Related Work

Graph learning has found wide applications in various fields due to the fact that many relationships between samples in many different domain datasets can be modeled as graphs [17]–[19]. By representing samples as nodes and relations between samples as edges, deep learning techniques can be utilized to extract important latent information from graph data. These information can be utilized for downstream tasks, such as node classification or clustering [20], [21], link prediction [22], [23], community discovery [24]–[26], graph reasoning [27], graph reconstruction and generation [28]–[30], etc.

Among different types of graph data, temporal graphs hold significant importance [31]. Traditional graphs are often static and are commonly represented using adjacency matrices to capture the connections between nodes. However, adjacency matrices fail to capture the temporal aspect of node changes. For instance, the establishment of relationships between nodes may occur at different times, and the same pair of nodes may have multiple relationship occurrences. To address this, researchers begin to focus on temporal graphs, where the entire graph is not observed in the form of an adjacency matrix, but rather the interactions between nodes are recorded in the form of adjacency tables [32], [33]. This allows temporal graphs to capture the sequential order of these interactions, thereby incorporating crucial temporal information.

It is worth noting that temporal graphs, using the interaction sequences (i.e., adjacency lists), no longer require the entire adjacency matrix to be loaded at once. Instead, the interaction records of nodes are divided into multiple batches, avoiding potential memory overflow and increasing flexibility. However, the loss of the adjacency matrix poses challenges for researchers who wish to obtain higher-order neighborhood information through node indexing. As a result, the focus is primarily on capturing first-order neighborhood information.

Through extensive evaluations of various classic temporal graph methods [34]–[36], it has been shown that the strategy of sacrificing higher-order neighborhood information in exchange for temporal information is successful, often yielding desirable results in specific scenarios. However, the absence of higher-order structural information inevitably limits the receptive field of the models. Therefore, we aim to find a strategy that obtains global structural information to the fullest extent possible without significantly increasing the computational complexity of temporal graph models. In the next section, we will provide the details of SET, which incorporates such a strategy.

III. Method

In this section, we will describe the details of SET. First, we give some preliminaries about temporal graph learning. Then we introduce the pre-training process (structural embedding generation) and training process (temporal embedding updating). Finally, we discuss how to control the optimization granularity by scaling the loss function.

A. Preliminaries

Given a temporal graph $G = (V, E, T)$, we define V as the node set, E as the interaction (edge) set, and T as the time set. In the actual graph, one interaction can be formulated as (u, v, t), where u denotes the source node, v denotes the target node, and t denotes the interaction time. Here we define initial features as $\mathbf{X}$. Note that in the field of temporal graph learning, only a small portion of the dataset has initial features, while most of the dataset is lacking initial features. For those datasets that lack initial features, researchers usually use one-hot vectors or positional encoding to generate features.

Temporal graph learning aims to capture important information in the temporal graph to generate embedding for each node. Such low-dimensional node embeddings $\mathbf{Z}$ can be used more flexibly and efficiently in downstream tasks and real-world application scenarios compared to high-dimensional graph data.

B. Structural Embedding Pre-Training

As mentioned above, we introduce the Graph AutoEncoder (GAE) [16] as the pre-training model to generate structural node embeddings as initial features. As one of the classical work in graph learning, GAE is highly regarded for its ability to capture global structural information efficiently and flexibly. GAE can be divided into two parts: encoder and decoder. The encoder can transfer the initial features $\mathbf{X}$ to the hidden

embeddings $\mathbf{H}$, and the decoder can transfer $\mathbf{H}$ to the final embeddings $\mathbf{Z}$.

Specially, the encoder module first passes the input graph data through the encoder to obtain a low-dimensional vector representation, and then reconstructs the low-dimensional vectors into the original graph data through the decoder module. By training the encoder and decoder, GAE can learn a compact representation of the graph data, which contains the structure and feature information of the graph.

Specially, denote $\mathbf{A}$ as the graph adjacency matrix, the encoder can be formulated as follows.

$$\mathbf{H} = f_E(\mathbf{A}, \mathbf{X}, \mathbf{W}^E) \tag{1}$$

f_E denotes the encoder function, which tends to consist of the linear layer and the activation layer (such as ReLU). $\mathbf{W}^E$ is the learnable parameter matrix, which used to adjust the weights of the encoder. Through the encoder, the GAE transforms the initial features $\mathbf{X}$ into the hidden embeddings $\mathbf{H}$ by passing structural information based on the adjacency matrix $\mathbf{A}$.

The decoder function f_D is similar as the encoder, which also tends to consist of the linear layer and the activation layer. In contrast, decoder aims to transforms the hidden embeddings $\mathbf{H}$ into the final node embeddings $\mathbf{Z}$, where $\mathbf{W}^D$ is also a learnable parameter matrix.

$$\mathbf{Z} = f_D(\mathbf{H}, \mathbf{W}^D) \tag{2}$$

After obtain the final node embeddings $\mathbf{Z}$, GAE utilizes them to reconstruct the adjacency matrix $\hat{\mathbf{A}}$ by the sigmoid function σ, where $\mathbf{W}^A$ is the learnable parameter matrix.

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}, \mathbf{Z}^T, \mathbf{W}^A) \tag{3}$$

Given the initial adjacency matrix $\mathbf{A}$ and the reconstruct matrix $\hat{\mathbf{A}}$, GAE utilizes the reconstruction loss as the loss function.

$$L_{rec} = \min_{\mathbf{W}^E, \mathbf{W}^D, \mathbf{W}^A} MSE(\hat{\mathbf{A}}, \mathbf{A}) \tag{4}$$

$MSE(\cdot, \cdot)$ denotes the MSE loss, i.e., mean squared error loss, also called L2 loss, is one of the most common loss functions in deep learning.The purpose of the loss function is to optimize the parameter matrixs $\mathbf{W}^E, \mathbf{W}^D, \mathbf{W}^A$ in the model so that the learned embeddings $\mathbf{Z}$ are as good as possible.

Notice that GAE, as a pre-training model, has a loss function that is not added to the loss function of the proposed SET method, which undergoes an independent and complete training process. Then SET directly uses the node embeddings generated by GAE as initialized features.

C. Temporal Embedding Training

For temporal embedding training, we utilize the classic temporal graph method HTNE [37] as our baseline model. HTNE is one of the early classical work on temporal graph learning, known for its ability to efficiently and flexibly mine the effects from historical events. Note that the inclusion of pre-training with GAE has little effect on the training efficiency and overall runtime of the HTNE model, since GAE runs only a small fraction of the time.

HTNE introduce the Hawkes process [38] as the main contribution, which argues that the historical events will influence the future event, and this influence will decreases with time. In this way, HTNE propose the conditional intensity to evaluate the probability between two nodes at any time.

Given an interaction (u, v, t), HTNE calculate their conditional intensity $\lambda_{u,y,t}$ as follows.

$$\lambda_{u,v,t} = \mu_{u,v} + h_{u,v,t} \tag{5}$$

Here $\mu_{u,v}$ denotes the basic intensity between two nodes without any other influences, i.e., $\mu_{u,v} = -||z_u - z_v||^2$. $h_{u,v,t}$ denotes the Hawkes increment intensity to evaluate the influences from node u's historical neighbors.

$$h_{u,v,t} = \sum_{i \in N_u} \mu_{i,v} \cdot a_{i,u} \cdot f(t - t_i) \tag{6}$$

N_u denotes the historical neighbors of node u. $a_{i,u}$ is the weight of neighbor i in N_u.

$$a_{i,u} = \frac{\exp(\mu_{i,u})}{\sum_{i' \in N_u} \exp(\mu_{i',u})} \tag{7}$$

$f(t - t_i)$ is the time function to calculate the time interval between historical time t_i and current time t, and then transforms the interval into the time weight, where δ is a learnable parameter.

$$f(t - t_i) = \exp(-\delta(t - t_i)) \tag{8}$$

After calculate the conditional intensity, HTNE introduces the negative sampling technology to construct the contrastive loss function as follows.

$$L_{tem} = -\log \sigma(\lambda_{u,v,t}) - \sum_{k \sim P_u} \log \sigma(1 - \lambda_{u,k,t}) \tag{9}$$

In this function, HTNE encourages positive pairs as close as possible and negative pairs as far away as possible. P_u is the sample distribution, which is proportional to node u's degree.

D. Enhanced Loss Function

As mentioned above, we introduce the structural embedding pre-training to enhance the information fusion process. In the HTNE model, we utilize the final node embeddings from GAE as its initial features. But this is not enough, because during model training, the embeddings of nodes are updated, resulting in deviations from structural embeddings. Therefore, we set up additional loss functions that encourage constantly updated time embeddings $\mathbf{Z}$ to align with the initialized structural embeddings $\mathbf{Z}^0$.

$$L_{align} = -||z_u^0 - z_u||^2 - ||z_v^0 - z_v||^2 \tag{10}$$

In addition, we introduce the power scaling error [39] for SET. The intuition of the power scaling error is that for samples or dimensions with small differences, they themselves

TABLE I
DESCRIPTION OF THE DATASETS.

Datasets	DBLP	Brain	BITotc	AMms
# Nodes	28,085	5,000	5,881	74,526
# Interactions	236,894	1,955,488	35,592	89,689
# Timestamps	27	12	27,487	5,082
# Class	10	10	21	5
# Type	Academic	Bioinformatic	Financial	E-commerce

do not require much optimization, but should focus on optimizing those samples or dimensions with large differences. By magnifying these differences to power levels, the model is more likely to optimize where it is not already optimized. To achieve this error, we propose a hyper-parameter γ. The final loss function can be formulated as follows.

$$L = \sum_{E} (L_{tem} + L_{align})^{\gamma} \tag{11}$$

By adjusting the value of γ, we can effectively control the scale of the loss function, so that we can artificially decide how much to strengthen the optimization of the model for the weak position.

IV. EXPERIMENT

A. Datasets

As shown in Table I, we construct experiments on several datasets from different areas (Academic, Bioinformatic, Financial, and E-commerce).

DBLP [37] is a co-author graph, which contains different computer science domains. If two authors appear on the same paper, we assume they have interacted.

Brain [40] is a connectivity graph of brain tissue in humans. If two brain tissues are activated at the same time, we think there is an interaction between them.

BITotc [41] is the transaction records of the Bitcoin exchange platform OTC. The user rates the transaction to get the corresponding label.

AMms [42] is magazine subscription graph on Amazon website, and users rate the magazines they subscribe to.

B. Baselines

To demonstrate the performance of our proposed method SET, we compare it with multiple classic SOTA methods on different tasks. These baseline methods can be divide into two parts.

(1) **Static graph methods**: AutoEncoder [43], DeepWalk [44], node2vec [45], GAE [16], SDCN [46].

(2) **Temporal graph methods**: HTNE [37], JODIE [47], TGN [48], MNCI [49], TREND [50].

C. Experimental Settings

We conduct two tasks on these models: node classification and node clustering. We also study the parameter sensitivity of the enhanced loss parameter γ.

We use default values for all comparison methods. For our proposed method SET, we set epoch number, learning rate, negative sampling number, batch size, embedding size and enhanced parameter γ as 50/100, 0.01, 3, 512, 128, and 2, respectively. All of our experiments are implemented on an NVIDIA RTX-3070Ti GPU.

D. Node Classification

In deep graph learning, node classification is a classical downstream task, which focuses on classify nodes into different categories. As shown in Table II, we report the node classification performance on all datasets. According to the results, the following conclusions can be drawn:

(1) Compared with static graph models, temporal graph models usually achieve better performance. It means that time information is important in graph learning.

(2) On some datasets, several static methods can also obtain good performance, which demonstrates that the global structure information that is often missing from temporal graph models has a negative influence on them (compared to the gains gained by static graph models).

(3) Our proposed model SET achieves the best performance on all datasets, because it combines both temporal and structural information, i.e., combining the advantages of both types of models.

E. Node Clustering

We also conduct node clustering task on DBLP and Brain datasets. Note that the node labels of BITotc and AMms are only suitable for classification, not clustering. Because they are rating tags, they do not represent the attribute information of the node well. Therefore, for an unsupervised task, it is difficult for models to learn the scoring label without reference. This is consistent with the actual results that all the models performed very poorly on these two datasets.

For node clustering tasks, we can see more clearly that the temporal graph model without structural information does not show better performance than the static graph model. This fully demonstrates the necessity of combining the structure information with the time series diagram model, and also proves the effectiveness of our method.

F. Parameter Sensitivity Study

As mentioned above, we introduce the power scaling error into the final loss function to control the optimization strengthen of the model for the weak position. To achieve this, we propose a hyper-parameter γ. Here we aims to discuss the effect of γ on model performance.

In particular, we select different values for γ, includes $\{0.5, 1, 1.25, 1.5, 1.75, 2, 3\}$. By adjusting the values of γ, we can observe the influence of γ on model's performance.

According to the Figure 2, we can observe that the hyper-parameter γ is useful for the model. When $\gamma = 1$, it is equal to we have not use it. Obviously, $\gamma = 1$ is not the best performance. In addition, the value of γ is different on different data sets. This means that these data sets have different distributions, making the model adapt as it learns. This is also consistent with the objective phenomenon that models

TABLE II

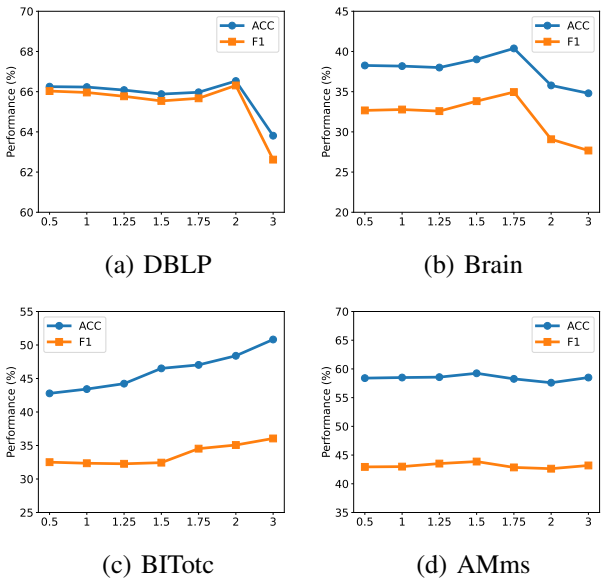
NODE CLASSIFICATION PERFORMANCE ON ALL DATASETS. NOTE THAT THE OPTIMAL RESULTS ARE HIGHLIGHTED IN BOLD, AND THE SUB-OPTIMAL RESULTS ARE UNDERLINED.

Datasets	DBLP		Brain		BITotc		AMms	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
AutoEncoder (2006)	43.17	40.44	25.12	23.15	27.43	28.11	42.41	41.59
DeepWalk (2014)	62.76	62.35	34.54	32.97	42.65	33.81	57.91	43.12
node2vec (2016)	63.87	63.48	38.72	33.17	44.23	32.47	57.93	43.10
GAE (2016)	64.89	65.46	38.16	32.84	41.17	32.59	57.09	42.77
SDCN (2020)	60.56	59.65	37.64	32.60	39.55	30.97	58.51	43.19
HTNE (2018)	65.47	65.32	33.94	29.11	38.84	32.22	57.96	43.04
JODIE (2019)	58.76	55.93	39.93	33.68	47.52	34.41	55.34	42.47
TGN (2020)	55.67	54.52	25.48	24.15	36.27	29.35	58.50	43.19
MNCI (2021)	65.88	65.41	38.54	34.35	44.93	33.61	58.48	43.20
TREND (2022)	61.28	59.88	39.85	34.44	32.14	27.32	58.43	43.16
SET (ours)	**66.53**	**66.31**	**40.38**	**34.95**	**48.39**	**35.07**	**59.24**	**43.87**

TABLE III

NODE CLUSTERING PERFORMANCE ON DBLP AND BRAIN DATASETS. NOTE THAT THE OPTIMAL RESULTS ARE HIGHLIGHTED IN BOLD, AND THE SUB-OPTIMAL RESULTS ARE UNDERLINED.

Datasets	DBLP				Brain			
	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
AutoEncoder	42.16	36.71	22.54	37.84	43.48	50.49	29.78	43.26
DeepWalk	28.95	22.03	13.73	24.79	41.28	49.09	28.40	42.54
node2vec	46.31	34.87	20.40	43.35	43.92	45.96	26.08	46.61
GAE	39.31	29.75	17.17	35.04	31.22	32.23	14.97	34.11
SDCN	46.69	35.07	23.74	40.31	42.62	46.61	27.93	41.42
HTNE	45.74	35.95	22.13	43.98	43.20	50.33	29.26	43.85
JODIE	20.79	11.67	11.32	13.23	19.14	10.50	5.00	11.12
TGN	19.78	9.82	5.46	10.66	17.40	8.04	4.56	13.49
MNCI	46.85	36.28	23.40	42.57	40.42	43.58	26.74	39.63
TREND	25.36	14.25	6.24	19.89	39.83	45.64	22.82	33.67
SET	**48.48**	**39.47**	**24.21**	**44.93**	**44.96**	**50.97**	**30.08**	**47.14**



(a) DBLP (b) Brain

(c) BITotc (d) AMms

Fig. 2. Parameter sensitivity study of different γ values on all datasets.

should choose different strategies when dealing with different scenarios. Based on this, we believe that the proposed γ is successful, which can help the model choose the optimization effort more flexibly.

V. CONCLUSION

In this paper, we discuss the necessity of structure embeddings for temporal graph learning, and point out the difficulty of combining structure information in temporal graph learning. To solve this problem, we propose the SET method, which introduces the pre-training of structure embeddings to avoid the huge computational cost of obtaining structure information in the training process of temporal graph models. Combined with experiments, we prove the effectiveness and feasibility of this strategy. Future, we will focus on the unified generalization framework to combine information that is difficult to obtain by various temporal graph learning models.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (no. 2020AAA0107100), and the National Natural Science Foundation of China (no. 62325604, 62276271).

REFERENCES

[1] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019.

[2] M. Liu, K. Liang, D. Hu, H. Yu, Y. Liu, L. Meng, W. Tu, S. Zhou, and X. Liu, "Tmac: Temporal multi-modal graph learning for acoustic event classification," in *ACM MM*, 2023.

[3] Y. Jin, A. Sharma, and R. T. Tan, "Dc-shadownet: Single-image hard and soft shadow removal using unsupervised domain-classifier guided network," in *ICCV*, 2021.

[4] K. Liang, S. Zhou, Y. Liu, L. Meng, M. Liu, and X. Liu, "Structure guided multi-modal pre-trained transformer for knowledge graph reasoning," *arXiv preprint arXiv:2307.03591*, 2023.

[5] K. Liang, L. Meng, M. Liu, Y. Liu, W. Tu, S. Wang, S. Zhou, X. Liu, and F. Sun, "A survey of knowledge graph reasoning on graph types: Static, dynamic, and multimodal," 2022.

[6] K. Liang, Y. Liu, S. Zhou, W. Tu, Y. Wen, X. Yang, X. Dong, and X. Liu, "Knowledge graph contrastive learning based on relation-symmetrical structure," *TKDE*, 2023.

[7] J. Wang, J. Gao, Y. Yuan, and Q. Wang, "Crowd localization from gaussian mixture scoped knowledge and scoped teacher," *TIP*, 2023.

[8] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *IJCAI*, 2018.

[9] W. Liang, X. Liu, Y. Liu, C. Ma, Y. Zhao, Z. Liu, and E. Zhu, "Consistency of multiple kernel clustering," 2023.

[10] J. Liu, X. Liu, Y. Yang, Q. Liao, and Y. Xia, "Contrastive multi-view kernel learning," *TPAMI*, 2023.

[11] Y. Jin, X. Wang, R. Yang, Y. Sun, W. Wang, H. Liao, and X. Xie, "Towards fine-grained reasoning for fake news detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[12] M. Shen, H. Yu, L. Zhu, K. Xu, Q. Li, and J. Hu, "Effective and robust physical-world attacks on deep learning face recognition systems," *TIFS*, 2021.

[13] J. Shi, Y. Zhang, X. Yin, Y. Xie, Z. Zhang, J. Fan, Z. Shi, and Y. Qu, "Dual pseudo-labels interactive self-training for semi-supervised visible-infrared person re-identification," in *ICCV*, 2023.

[14] J. Zhou, Q. Liu, Q. Jiang, W. Ren, K.-M. Lam, and W. Zhang, "Underwater camera: improving visual perception via adaptive dark pixel prior and color correction," *International Journal of Computer Vision*, 2023.

[15] J. Zhou, B. Li, D. Zhang, J. Yuan, W. Zhang, Z. Cai, and J. Shi, "Ugif-net: An efficient fully guided information flow network for underwater image enhancement," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[16] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[17] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *TKDE*, 2018.

[18] H. Yu, C. Ma, M. Liu, X. Liu, Z. Liu, and M. Ding, "Guardfl: Safeguarding federated learning against backdoor attacks through attributed client graph clustering," *arXiv preprint arXiv:2306.04984*, 2023.

[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *NeurIPS*, 2017.

[20] H. Wu, N. Li, J. Zhang, S. Chen, M. K. Ng, and J. Long, "Collaborative contrastive learning for hypergraph node classification," *Pattern Recognition*, 2023.

[21] Y. Liu, J. Xia, S. Zhou, S. Wang, X. Guo, X. Yang, K. Liang, W. Tu, Z. S. Li, and X. Liu, "A survey of deep graph clustering: Taxonomy, challenge, and application," *arXiv preprint arXiv:2211.12875*, 2022.

[22] W. Tu, S. Zhou, X. Liu, C. Ge, Z. Cai, and Y. Liu, "Hierarchically contrastive hard sample mining for graph self-supervised pre-training," *TNNLS*, pp. 1–14, 2023.

[23] W. Tu, S. Zhou, X. Liu, Y. Liu, Z. Cai, E. Zhu, C. Zhang, and J. Cheng, "Initializing then refining: A simple graph attribute imputation network," in *IJCAI*, 2022.

[24] Y. Jin, Y.-C. Lee, K. Sharma, M. Ye, K. Sikka, A. Divakaran, and S. Kumar, "Predicting information pathways across online communities," in *KDD*, 2023.

[25] Y. Liu, K. Liang, J. Xia, S. Zhou, X. Yang, , X. Liu, and Z. S. Li, "Dink-net: Neural clustering on large graphs," in *Proc. of ICML*, 2023.

[26] Y. Liu, X. Yang, S. Zhou, and X. Liu, "Simple contrastive graph clustering," *TNNLS*, 2023.

[27] L. Meng, K. Liang, B. Xiao, S. Zhou, Y. Liu, M. Liu, X. Yang, and X. Liu, "Sarf: Aliasing relation assisted self-supervised learning for few-shot relation reasoning," *arXiv preprint arXiv:2304.10297*, 2023.

[28] Y. Liu, Z. Liu, S. Li, Z. Yu, Y. Guo, Q. Liu, and G. Wang, "Cloud-vae: Variational autoencoder with concepts embedded," *Pattern Recognition*, 2023.

[29] Z. Dong, J. Jin, Y. Xiao, S. Wang, X. Zhu, X. Liu, and E. Zhu, "Iterative deep structural graph contrast clustering for multiview raw data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2023.

[30] P. Zhang, S. Wang, L. Li, C. Zhang, X. Liu, E. Zhu, Z. Liu, L. Zhou, and L. Luo, "Let the data choose: flexible and diverse anchor graph fusion for scalable multi-view clustering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 262–11 269.

[31] M. Liu, K. Liang, B. Xiao, S. Zhou, W. Tu, Y. Liu, X. Yang, and X. Liu, "Self-supervised temporal graph learning with temporal and structural intensity alignment," *arXiv preprint arXiv:2302.07491*, 2023.

[32] J. Gao and B. Ribeiro, "On the equivalence between temporal and static graph representations for observational predictions," *arXiv preprint arXiv:2103.07016*, 2021.

[33] M. Liu, Y. Liu, K. Liang, S. Wang, S. Zhou, and X. Liu, "Deep temporal graph clustering," *arXiv preprint arXiv:2305.10738*, 2023.

[34] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion proceedings of the the web conference 2018*, 2018.

[35] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *ICLR*, 2019.

[36] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," *arXiv preprint arXiv:2101.05974*, 2021.

[37] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *KDD*, 2018.

[38] A. G. Hawkes, "Point spectra of some mutually exciting point processes," *Journal of the Royal Statistical Society: Series B (Methodological)*, 1971.

[39] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *KDD*, 2022.

[40] M. G. Preti, T. A. Bolton, and D. Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *Neuroimage*, 2017.

[41] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *ICDM*, 2016.

[42] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *EMNLP-IJCNLP*, 2019.

[43] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, 2006.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014.

[45] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[46] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proceedings of the web conference 2020*, 2020.

[47] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *KDD*, 2019.

[48] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[49] M. Liu and Y. Liu, "Inductive representation learning in temporal networks via mining neighborhood and community influences," in *SIGIR*, 2021.

[50] Z. Wen and Y. Fang, "Trend: Temporal event and node dynamics for graph representation learning," in *Proceedings of the ACM Web Conference 2022*, 2022.