CURRICULUM DYNAMIC GRAPH INVARIANT LEARNING UNDER DISTRIBUTION SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic graph neural networks have attracted intensive research interests recently but generally suffer from handling distribution shifts that widely exist in dynamic graphs. Although the existing works attempt to disentangle the invariant and variant patterns, they ignore the training status of the graph neural network and the importance of training samples at different times, which are critical to model invariant patterns accurately in dynamic graphs under distribution shifts. In this paper, we study distribution shifts in dynamic graphs with curriculum learning for the first time, which remains unexplored and faces the following challenges: (i) how to design a tailored training status evaluation strategy; and (ii) how to design a tailored sample importance reweighting strategy, so as to handle distribution shifts in dynamic graphs. To address these challenges, we propose a Curriculum Dynamic Graph Invariant Learning (**CDGIL**) model, which can handle distribution shifts in dynamic graphs by capturing and utilizing invariant and variant patterns guided by the proposed curriculum learning strategy. Specifically, we first propose a dual disentangled dynamic attention network to capture the invariant and variant patterns, respectively. Next, we propose a self-paced intervention mechanism based on training status to create adversarial samples by reassembling variant patterns across neighborhoods and time stamps to remove the spurious impacts of variant patterns. Finally, we propose a sample importance reweighting strategy to distinguish invariant and variant patterns better via focusing on the key training samples. Extensive experiments on both synthetic and real-world dynamic graph datasets demonstrate the superiority of our proposed method over state-of-the-art baselines under distribution shifts.

1 INTRODUCTION

Dynamic graphs are ubiquitous in the real world, where some nodes and edges evolve along with time. Thanks to the power to capture both structural and temporal patterns simultaneously, dynamic graph neural networks (DGNNs) have achieved impressive successes in various real-world applications, including financial networks (Nascimento et al., 2021), social networks (Rossi et al., 2020b), traffic networks (Guo et al., 2021), etc. However, existing DGNNs have certain limitations when it comes to addressing distribution shift, which is common in practical scenarios (Holt, 2004; Zhang & Qi, 2005; Jin et al., 2021; Gao et al., 2021; Russell et al., 2019; Berk, 1983).

Existing research on out-of-distribution DGNNs primarily focuses on invariant learning, assuming the existence of invariant patterns across distributions (Zhang et al., 2022; Wu et al.). These studies incorporate the use of an additional loss component called invariant risk minimization (IRM) to capture invariances (Arjovsky et al., 2019). However, the fixed proportion of ERM and IRM in the training loss poses challenges for dynamically adjusting the training strategy in existing works. Moreover, the importance of diversity among samples has been overlooked when attempting to disentangle the invariant patterns, i.e., emphasizing samples that exhibit spurious correlations makes it difficult to capture invariance effectively. Therefore, adjusting the weight of different samples based on their properties and the training stage is crucial for capturing the invariant pattern successfully.

Curriculum learning (Bengio et al., 2009; Wang et al., 2021b) is a powerful approach that enables the dynamic adjustment of the training strategy, such as data reweighting and objection adjustment based on different training statuses. In this paper, we explore the application of curriculum learning to address distribution shifts in dynamic graphs, which is a novel and unexplored area, and it poses the following challenges: (i) devising a customized evaluation strategy for training statuses, and (ii) designing a tailored strategy for reweighting the importance of samples, specifically to handle distribution shifts in dynamic graphs, e.g., the appearance or disappearance of graph nodes and edges.

To address these challenges, this paper presents a Curriculum Dynamic Graph Invariant Learning (**CDGIL**) model. The CDGIL model is introduced as a solution for handling distribution shifts in dynamic graphs by effectively capturing and utilizing both invariant and variant patterns. In particular, we first propose a dual disentangled dynamic attention network to capture the invariant and variant patterns, respectively. Furthermore, we propose a novel self-adaptive intervention method based on training status to create adversarial samples by reassembling variant patterns across neighborhoods and time stamps to remove the spurious impacts of variant patterns. Finally, we propose a novel curriculum-based sample importance reweighting method, which evaluates the importance of various data samples, and dynamically adjusts their weight in the training process. In summary, we made the following contributions:

- We investigate curriculum dynamic graph invariant learning for the first time, and propose a exquisitely designed curriculum learning method for dynamic graph generalization, which can automatically adjust both the training procedure and data weight according to data sample importance and different training stages.
- We design a self-adaptive intervention method to combine the invariant and variant parts, aiming to strengthen the generalization ability of our model and enhance our performance while facing data from unknown environments.
- We propose a novel curriculum-based sample importance reweighting method, by increasing the weight of data samples that are important to the current model and decreasing the weight of data samples that are not important to the current model, our models can get more suitable data inputs and get better performance optimization.
- Extensive experiments show that our proposed CDGIL method has the ability to outperform all of the state-of-the-art baselines on all of the dynamic graph datasets with distribution shifts.

2 Related Works

2.1 Dynamic Graph Neural Networks

Dynamic Graph Neural Networks (DGNN) (Skarding et al., 2021; Zhu et al., 2022) have attracted intensive research attention as a powerful approach to handling the complex structural and temporal information in dynamic graphs. It has been applied to a wide range of real-world sceneries, such as action recognition (Yan et al., 2018), epidemic forecasting (Panagopoulos et al., 2020; Rozemberczki et al., 2021), social networks (Rossi et al., 2020b; Cai et al., 2022; Goyal et al., 2020), recommendation (Song et al., 2019), traffic prediction (Guo et al., 2021; Diao et al., 2019), and anomaly detection (Liu et al., 2021; Weber et al., 2019; Wang et al., 2021a). There are two mainstream DGNN methods, whose main difference between them is the order of dealing with time series and structural information. One of them utilizes a graph neural network (GNN) to aggregate structural information for the graph at each time at first and then adopts sequence models such as recurrent neural networks (RNN) (Yang et al., 2021; Sun et al., 2021; Hajiramezanali et al., 2019; Seo et al., 2018b; Pareja et al., 2020) or self-attention modules (Sankar et al., 2020) to process the temporal information. While the other method proposes to transform temporal links into a time-dependent function. using time-encoding techniques, then use GNN (Wang et al., 2021c; Cong et al., 2021; Xu et al., 2020; Rossi et al., 2020a) to process the graph with the time series information and get structural information. DIDA (Zhang et al., 2022) is the only work for dynamic graph distribution shift, it proposes a disentangled spatio-temporal attention network and a spatio-temporal intervention mechanism in order to handle spatio-temporal distribution shifts in dynamic graphs. But its training strategy is not related to the training status or sample importance.

2.2 Out-of-Distribution Generalization for Graph

Traditional machine learning method has the assumption that the training and test sets are guaranteed to be independently and identically distributed (i.i.d.), but this assumption does not hold in many real-world scenarios (Shen et al., 2021). Ignoring the fact that this assumption does not always hold may lead to the degradation of model performance (Fang et al., 2020). Various work has been done on this important question, such as IRM (Arjovsky et al., 2019), DRO (Rahimian & Mehrotra, 2019), REx (Krueger et al., 2021), and so on. In particular, designing models able to generalize in out-of-distribution (OOD) scenarios has attracted remarkable interest in graph representation learning. GIL (Li et al., 2022b) is proposed to capture the invariant relationships between predictive graph structural information and labels in a mixture of latent environments. OOD-GNN (Li et al., 2022a) proposes a nonlinear graph representation decorrelation method and a scalable global-local weight estimator to learn out-of-distribution (OOD) generalized graph representation under complex distribution shifts. However, most of the previous generalization work is only conducted on the static graph and failed to capture the feature of dynamic graphs such as timestamps.

2.3 Curriculum Learning

Curriculum learning (CL) (Bengio et al., 2009; Wang et al., 2021b; Soviany et al., 2022) is a popular method to train machine learning models in a meaningful order, such as from easier data to harder data, which is able to improve the performance of machine learning models, as well as bring faster convergence. Thanks to its powerful ability, curriculum learning have been widely applied to numerous branches of machine learning, including computer vision (Huang et al., 2020b), natural language processing (Cirik et al., 2016), speech (Braun et al., 2017), medical (Lotter et al., 2017), robotics (Florensa et al., 2017) and etc. Curriculum learning mainly includes two parts, difficulty measurer and training scheduler. The difficulty measurer aims to evaluate the difficulty of different input data, and the training scheduler can adjust the sequence according to the difficulty. Typically, there are two kinds of curriculum learning. One is predefined curriculum learning, or vanilla curriculum learning, where both the difficulty measurer and training scheduler is designed by human prior and expert domain knowledge. The other is automatic curriculum learning, where one or both difficulty measurer and training scheduler is learned from data-driven algorithms, which can change with different training procedures, rather than a prior decision. And there are many kinds of automatic curriculum learning, such as self-paced learning (Kumar et al., 2010), transfer learning (Weinshall et al., 2018), reinforcement learning (Florensa et al., 2017) and teacher-student method (Kim & Choi, 2018). However, the current curriculum learning methods failed to capture the dynamic feature, and thus can't be applied to dynamic graph out-of-distribution generalization.

3 Preliminary

In this section, we introduce the notations of dynamic graphs and their spatio-temporal distribution shift.

3.1 Dynamic Graph

Typically, a graph consists of nodes and edges: $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of vertices and $\mathbf{E}$ is the set of edges. And in a dynamic graph, some graph nodes or graph edges will appear or disappear with the passing of time. So a dynamic graph can be formalized into the following form: $\mathbf{G} = \{\mathbf{G}_t : t = 1, 2, \cdots, T\}$, where $\mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t)$, t is the different time stamp, and T is the total number of time stamps. $\mathbf{V} = \bigcup_{t=1}^{T} \mathbf{V}_t$ is the node set of dynamic graph, and $\mathbf{E} = \bigcup_{t=1}^{T} \mathbf{E}_t$ is the edge set of dynamic graph. The prediction task of the dynamic graph is predicting future labels using history graphs: $p(\mathbf{Y}_{t+1}|\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_t) = p(\mathbf{Y}_{t+1}|\mathbf{G}_{1:t})$. In this paper, we mainly study node-level tasks, so $\mathbf{Y}_{t+1}$ typically represents the node property or edge status. And the prediction task can be formulated as a minimization task:

$$\min_{\theta} E_{y_{t+1} \sim p_{\text{train}}(\mathbf{y}_{t+1})} \mathbf{L}(f_\theta(\mathbf{G}_{1:t}), y_{t+1}), \tag{1}$$

where y_{t+1} represents the instance of label, and $\mathbf{y}_{t+1}$ represents the abstract of label.

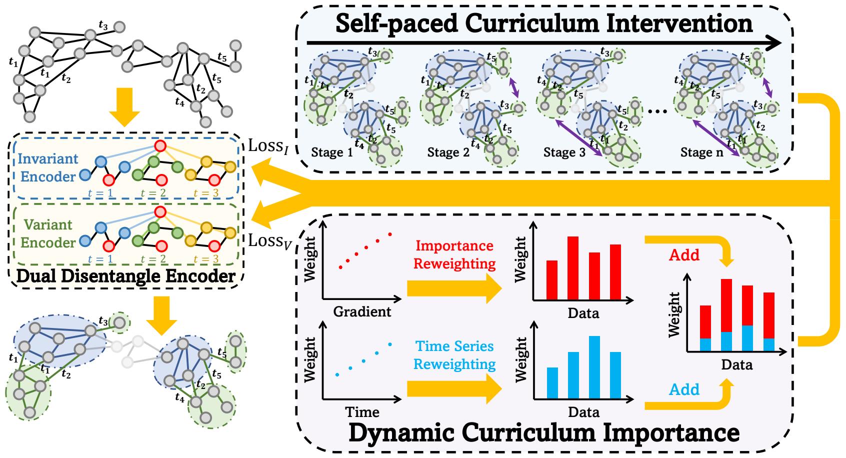

Figure 1: The framework of our proposed method **CDGIL**. Firstly, we adopt the dual disentangled dynamic attention network to capture the invariant patterns and variant patterns separately. Secondly, we conduct the self-paced intervention on dynamic graphs, in order to further enhance the generalization ability of the invariant encoder using variant patterns. Thirdly, we apply the dynamic curriculum method for importance reweighting, calculating the weights of data by evaluating its time as well as importance. Finally, we calculate the training loss, updating the invariant encoder and variant encoder, separately.

3.2 Distribution Shift

However, learning a model for training distribution may suffer from a distribution shift between training distribution and testing distribution, which is still a key challenge. In our paper, we adopt the assumption that the relationship between dynamic graph data and the label remains the same in training distribution and testing distribution: $p_{train}(\mathbf{Y}_{t+1}|\mathbf{G}_{1:t}) = p_{test}(\mathbf{Y}_{t+1}|\mathbf{G}_{1:t})$ following (Wang et al., 2021c; Qiu et al., 2020; Huang et al., 2020a; Zhou et al., 2018; Trivedi et al., 2019). However, the distribution of the dynamic graph is different between training distribution and testing distribution: $p_{train}(\mathbf{G}_{1:t}) \neq p_{test}(\mathbf{G}_{1:t})$, i.e. our model is suffering covariate shift problem.

4 Method

To tackle the problem of spatio-temporal distribution shift in dynamic graphs, we propose our novel curriculum dynamic graph invariant learning in three parts. Firstly, in Section 4.1, we introduce our base model: dual disentangled dynamic attention network. Secondly, in Section 4.2, we introduce our proposed self-paced curriculum intervention method. Finally, in Section 4.3, we introduce our dynamic curriculum method for importance reweighting method.

4.1 Dual Disentangled Dynamic Graph Attention Networks

In this subsection, we introduce our base model: dual disentangled dynamic graph attention networks (D^{3}GAT). In previous work (Zhang et al., 2022), disentangled dynamic graph attention networks (DDGAT) compute the invariant patterns and variant patterns at the same time, limiting the ability of the variant patterns to adjust with the invariant patterns. Here we introduce it first.

Disentangled dynamic graph attention networks (DDGAT) apply the self-attention mechanism to integrate the node information of its neighborhood. The neighborhood of node v at timestamp t can be defined as: $N_t(v) = \{u : (v, u) \in \mathbf{E}_t\}$. And the history neighborhood of node v at timestamp t is defined as: $\widetilde{N}_t(v) = \bigcup_{\tau=1}^{t} N_\tau(v)$. To fully explore the neighborhood relationship and bring together all the information, the spatio-temporal graph attention mechanism aggregates all of the history neighbors of node v. And to express the different edges of different times, DDGAT also adopts $\mathbf{TE}(t)$ as time encoding of timestamp t. Thus the self-attention mechanism is carried out between the concatenation of node embedding as well as time encoding of node v and all of its historic neighbors. Thus, we have query, key, and value as below, where $\mathbf{z}_v^t$ for the embedding of node v at time t:

$$\mathbf{q}_v^t = \mathbf{W}_q(\text{concat}(\mathbf{z}_v^t, \mathbf{TE}(t))), \mathbf{k}_u^{t'} = \mathbf{W}_k(\text{concat}(\mathbf{z}_u^{t'}, \mathbf{TE}(t'))), \mathbf{v}_u^{t'} = \mathbf{W}_v(\text{concat}(\mathbf{z}_u^{t'}, \mathbf{TE}(t'))). \tag{2}$$

And the query, key, and value are used to furthermore calculate the invariant and variant structural pattern masks. To further capture invariant patterns, a learnable mask $\mathbf{m}_f = \text{Softmax}(\mathbf{w}_f)$ is proposed to mask the variant feature. Then is the result of invariant and variant patterns:

$$\mathbf{z}_I^t = \text{Softmax}(\frac{\mathbf{q} \cdot \mathbf{k}^T}{\sqrt{d}})\mathbf{v} \cdot \mathbf{m}_f, \tag{3}$$

$$\mathbf{z}_V^t = \text{Softmax}(-\frac{\mathbf{q} \cdot \mathbf{k}^T}{\sqrt{d}})\mathbf{v}. \tag{4}$$

And after calculating the invariant and variant patterns, they are added up as the node feature input of the next layer:

$$\mathbf{z}_v^t = \mathbf{z}_I^t(v) + \mathbf{z}_V^t(v), \tag{5}$$

where the output vector $\mathbf{z}_v^t$ is for the next layer: layer $l+1$, while the input vectors $\mathbf{z}_I^t(v)$ and $\mathbf{z}_V^t(v)$ are from the current layer: layer l. With the aggregation and disentanglement of neighborhood information and invariant and variant pattern, the proposed disentangled dynamic graph attention networks with L layers can represent the invariant patterns as well as the variant patterns in the L-th dynamic history neighborhood.

To summarize, the disentangled dynamic graph attention networks (DDGAT) get the input of the whole dynamic graph, and the output of DDGAT is the invariant pattern of the node feature as well as the variant pattern of the node feature. But as mentioned above, it jointly calculates the invariant pattern and the variant pattern of the node feature, which limits pattern learning, because they are decided jointly, and can not adjust with regard to the other.

In our method, we adopt dual DDGAT, utilizing two modules for recognizing the invariant pattern and the variant pattern, separately. We call the two module invariant encoder and variant encoder for short. Invariant encoder and variant encoder have different learning objectives. The invariant encoder aims to precisely discover the invariant pattern of graphs, while the variant encoder captures the variant pattern of graphs in order to improve the generalization ability of the invariant encoder. The pattern captured by the invariant encoder and variant encoder is presented as $\mathbf{z}_I$ and $\mathbf{z}_V$, respectively.

4.2 Self-paced Curriculum Intervention

The objective of the invariant encoder is to capture the invariant patterns precisely, and the variant patterns are a good source to let the invariant encoder generalize better. The adversarial variant patterns are reassembled with the invariant pattern to conduct intervention and the intensity of intervention is decided by the training stage.

The reassembled result is presented as $\mathbf{z}_r$, and the result is controlled by a parameter reassemble λ_r, where the larger the λ_r is, the greater the intensity of the intervention result is:

$$\mathbf{z}_r = \text{Assemble}(\mathbf{z}_I, \mathbf{z}_V, \lambda_r), \tag{6}$$

where the assemble function is calculated from the intervention result of the variant part and invariant part. Following (Zhang et al., 2022), we adopt the approximate intervention method, randomly replacing a specific variant pattern with another random variant pattern:

$$\mathbf{z}_I^{t_1}(u), \mathbf{z}_V^{t_1}(u) \leftarrow \mathbf{z}_I^{t_1}(u), \mathbf{z}_V^{t_2}(v). \tag{7}$$

The λ_r is the parameter controlling the intervention intensity. In the early training stage, the invariant and variant encoder are not well-trained, thus the λ_r should be relatively smaller. While in the late training stage, the invariant and variant encoder are well-trained in training distribution and eager to generalize to testing distribution or other unknown distributions. So this time, the λ_r should be relatively larger. In our proposed self-paced curriculum intervention method, we adjust the λ_r with our training stage and evaluate the training stage through the functional loss ℓ. We set a threshold for the minimum loss ℓ_{min} for each different task, and calculate the λ_r for each epoch:

$$\lambda_r = \lambda_0 * \min(\ell, \ell_{min})^{-1}. \tag{8}$$

To summarize, for the invariant encoder's loss, we add the training loss ℓ_{trainI} to the loss of reassembled with adversarial variant patterns, $\ell_{assemble}$. Thus, we have our invariant loss for our invariant encoder.

$$\ell_I = \ell_{trainI} + \ell_{assemble} * \lambda_{assemble}, \tag{9}$$

where $\lambda_{assemble}$ is the weight assigned to $\ell_{assemble}$, because it's not so important as the loss of training.

The objective of the variant encoder is to capture the variant patterns in order to enhance the generalization ability of the invariant encoder. And we maximize the difference between variant patterns reassembled with the invariant patterns and the invariant patterns themselves, further enhancing the variation between variant patterns and invariant patterns, formatting adversarial variant patterns.

To capture adversarial variant patterns compared to invariant patterns, we evaluate the difference between the reassembled patterns and the invariant patterns. In this paper, we utilize the mean of the square of the difference between them ($\mathbf{z}_r$ and $\mathbf{z}_I$):

$$\text{Diff}_{var} = \text{mean}((\mathbf{z}_r - \mathbf{z}_I)^2). \tag{10}$$

On the basis of the training loss of the variant decoder, we subtract our calculated difference variant patterns made from the training loss. So the final loss of the variant encoder is presented below:

$$\ell_V = \ell_{trainV} - \text{Diff}_{var} * \lambda_{diff}, \tag{11}$$

where λ_{diff} is the weight assigned to Diff_{var}.

In our method, the capture of invariant patterns should adjust with different training stages, and the capture of variant patterns should adjust with different invariant patterns. So the optimization of our proposed dual disentangled dynamic attention network is of two steps. In the first step, we calculate the loss of the invariant encoder and optimize its weight while freezing the weight of the variant encoder. And in the second step, we calculate the loss of the variant encoder and optimize its weight while freezing the weight of the invariant encoder. Our two-step optimization can capture invariant patterns more precisely and capture proper variant patterns.

4.3 Dynamic Curriculum Importance

4.3.1 Curriculum Time Serial Reweighting

In dynamic graphs, all the nodes and edges are related to a certain timestamp. And the data of different timestamps tend to have different features. With the smaller gap in time, the graph data tend to have more similar properties. To further consider the time information of the graph, we propose the curriculum time serial reweighting method for dynamic graphs.

More specifically, we change the weight of the data sample of the graph with timestamp t: $\mathbf{w}_t = (1+\lambda)^t$. And for all of the data, we adjust their weight to maintain that the average weight of all the data remains 1, in order to avoid further introducing bias:

$$\mathbf{w}_{time} = \mathbf{w}_t + (1 - \text{average}(\mathbf{W}_t)), \tag{12}$$

where $\mathbf{W}_t$ is the vector of all the dynamic graph data according to weights, consisting of each w_t of data samples.

4.3.2 Curriculum Sample Importance Reweighting

In curriculum learning, one of the basic assumptions is that different data should not be treated equally, i.e. different data could have different weights or order. However, the traditional method for dynamic graph out-of-distribution generalization treats all of the data in the same way. Thus here we discuss how to distinguish the importance of different data, and how to change their weights in order to gain better results and generalization ability.

During experiments, we discover that the gradient is an important signal of the data's importance to the model now. We made the trial that observed the pattern of the gradient of the prediction results, and we discover that with the entanglement of the variant features and invariant features, the gradient descent failed to optimize the model to fit the training set or enhance the generalization ability. So we propose our novel curriculum learning for sample reweighting through the gradient measure.

For data samples $D = \{d_1, d_2, \cdots, d_n\}$, we first calculate their loss $\ell(\mathbf{y}_{pred}, \mathbf{y}_{true})$, and perform gradient backward, thus we have the gradient of all of the prediction results: $\text{grad}(d_1)$, $\text{grad}(d_2)$, $\cdots$, $\text{grad}(d_n)$. Using the gradient backward result of the prediction results, we can reweight our data's weight via a function f mapping gradients to weights:

$$\mathbf{w}^i_{imp} = f(\text{grad}(d_i)). \tag{13}$$

And the mapping function is a two-stage segmentation function. For the gradient that is smaller than zero, the function is $f(x) = 1$, which will increase the weight of their data samples. And for the gradient that is larger than zero, the function is $f(x) = -\exp(\text{sigmoid}(\log(x) * A + B) * C + D)$, where the A, B, C and D are hyper-parameters. This mapping function will increase the weight of the data samples that have a negative gradient, slightly decrease the data samples that have a large positive gradient, and largely decrease the data samples that have small positive samples. This mapping function will reweight their weight by evaluating the importance and further have better optimization and generalization ability.

After calculating the weights of the time series as well as sample importance, we add them together for the final output for each data sample $\mathbf{w}_{time} + \mathbf{w}_{imp}$. However, there appear some cases while applying these weights that the weights drop below zero and resulting in loss ℓ_{train} drop below zero. Thus we adopt a soft positive function on the weights in order to avoid optimization of the contradictory direction:

$$\mathbf{w} = \max(\mathbf{w}_{time} + \mathbf{w}_{imp}, 0). \tag{14}$$

The final reweighting weight w is used to calculate the training loss ℓ_{train}. And the overall algorithm of our proposed CDGIL method is shown in Table1.

Algorithm 1 Our proposed curriculum dynamic graph invariant learning method

```
1: Initialize. Number of training epochs $E$, dynamic graph dataset $G$, invariant decoder $D_I$, variant decoder
   $D_V$.
2: for $e = 1 : E$ do
3:     Calculate $z_I$ and $z_V$: $z_I^t = \text{Softmax}(\frac{q \cdot k^T}{\sqrt{d}})v \cdot m_f$,$z_V^t = \text{Softmax}(-\frac{q \cdot k^T}{\sqrt{d}})v$.
4:     Calculate $w_{time}$ and $w_{imp}$: $w_{time} = w_t + (1 - \text{average}(W_t))$,,$w^i_{imp} = f(\text{grad}(d_i))$.
5:     Calculate $w$: $w = \max(w_{time} + w_{imp}, 0)$.
6:     Calculate invariant loss $\ell_I = \ell_{trainI} + \ell_{assemble} * \lambda_{assemble}$.
7:     Perform gradient descent on the invariant encoder.
8:     Calculate variant loss $\ell_V = \ell_{trainV} - \text{Diff}_{var} * \lambda_{diff}$.
9:     Perform gradient descent on the variant encoder.
10: end for
11: Output. The well-trained model.
```

5 EXPERIMENTS

In this section, we present the experiments to verify the effectiveness and wide applicability of our proposed method, including experimental setup, results, and ablation studies. Please refer to Appendix for more details.

5.1 EXPERIMENTAL SETUP

Baselines. In the experiments, we consider three kinds of representative baselines, including (1) static GNNs: **GAE** (Kipf & Welling, 2016b) and **VGAE** (Kipf & Welling, 2016b), (2) dynamic GNNs: **GCRN** (Seo et al., 2018a), **EGCN** (Pareja et al., 2020), **DySAT** (Sankar et al., 2020), and (3) OOD generalization methods: **IRM** (Arjovsky et al., 2019), **GroupDRO** (Sagawa et al., 2019), **V-REx** (Krueger et al., 2021), **DIDA** (Zhang et al., 2022).

Datasets. The experiments are conducted on both real-world and synthetic dynamic graph datasets.

- **COLLAB** (Tang et al., 2012) is a dataset of cross-domain collaboration recommendations, including the authors and publications from 1990 to 2005. The nodes of the dynamic graph are authors, and the edges between nodes are coauthorship. Its edges contain five

Table 1: The experiment results (ROCAUC%) of different methods on real-world link prediction datasets. The best results are in bold. ‘w/o DS’ and ‘w/ DS’ denote test data with and without distribution shift.

Model	**COLLAB**		**Yelp**	
Test Data	w/o DS	w/ DS	w/o DS	w/ DS
GAE	77.15±0.50	74.04±0.75	70.67±1.11	64.45±5.02
VGAE	86.47±0.04	74.95±1.25	76.54±0.50	65.33±1.43
GCRN	82.78±0.54	69.72±0.45	68.59±1.05	54.68±7.59
EGCN	86.62±0.95	76.15±0.91	78.21±0.03	53.82±2.06
DySAT	88.77±0.23	76.59±0.20	78.87±0.57	66.09±1.42
IRM	87.96±0.90	75.42±0.87	66.49±10.78	56.02±16.08
VREx	88.31±0.32	76.24±0.77	79.04±0.16	66.41±1.87
GroupDRO	88.76±0.12	76.33±0.29	**79.38±0.42**	66.97±0.61
DIDA	91.97±0.05	81.87±0.40	78.22±0.40	75.92±0.90
Ours	**93.60±0.11**	**84.39±0.54**	77.15±1.54	**76.44±1.79**

Table 2: The experiment results (ROCAUC%) of different methods on synthetic link prediction datasets. The best results are in bold. ‘w/o DS’ and ‘w/ DS’ denote test data with and without distribution shift.

Model	**synthetic-0.4**		**synthetic-0.6**		**synthetic-0.8**	
Split	Train	Test	Train	Test	Train	Test
GCRN	69.60±1.14	72.57±0.72	74.71±0.17	72.29±0.47	75.69±0.07	67.26±0.22
EGCN	78.82±1.40	69.00±0.53	79.47±1.68	62.70±1.14	81.07±4.10	60.13±0.89
DySAT	84.71±0.80	70.24±1.26	89.77±0.32	64.01±0.19	94.02±1.29	62.19±0.39
IRM	85.20±0.07	69.40±0.09	89.48±0.22	63.97±0.37	**95.02±0.09**	62.66±0.33
VREx	84.77±0.84	70.44±1.08	89.81±0.21	63.99±0.21	94.06±1.30	62.21±0.40
GroupDRO	84.78±0.85	70.30±1.23	89.90±0.11	64.05±0.21	94.08±1.33	62.13±0.35
DIDA	87.92±0.92	85.20±0.84	91.22±0.59	82.89±0.23	92.72±2.16	72.59±3.31
Ours	**89.77±0.15**	**87.60±0.18**	**91.88±0.26**	**84.67±0.27**	94.84±0.29	**79.54±0.87**

sub-domains: data mining, medical informatics, theory, visualization, and database. Here we choose data mining as the testing domain, and the other four sub-domains as the training domains.

- **Yelp** (Sankar et al., 2020) is a dataset of businesses and reviews, including the customers and their reviews on businesses. The nodes of the dynamic graph are customers or businesses, and the edges between them are reviews. Its edges contain five sub-domains: pizza, American food, coffee & tea, sushi bars, and fast food. Here we choose pizza as the testing domain, and the other four sub-domains as the training domains.
- **Synthetic Dataset** (Zhang et al., 2022) is one manually designed dataset based on collab dataset, introducing external node features to create distribution shifts. The external node features are related to the average positive sample rate $\overline{p}$, wherein the testing dataset, $\overline{p}$ is set to 0.1, and in the training dataset, $\overline{p}$ is set to 0.4, 0.6 or 0.8, formatting the three datasets: synthetic-0.4, synthetic-0.6, and synthetic-0.8, where we can see that the larger $\overline{p}$ is, the larger the distribution shift between the training part and the testing part of the dataset.

5.2 RESULTS

- The performances of the baseline methods drop significantly under distribution shifts, although some of them show relatively competitive results on test data without distribution shifts. For example, when existing distribution shifts the performance of DySAT, which is one representative dynamic method, drops more than 10% in COLLAB and Yelp datasets. It shows that the existing GyGNNs fail to handle distribution shifts and focus on variant patterns to make predictions, leading to poor OOD generalization. As one dynamic graph OOD generalization method, DIDA shows strong performances to handle distribution shifts but the performance drop is also significant. It means that although this method attempts to capture invariant patterns and only uses them to make predictions, the result is not promising since it ignores the training status and sample importance when learning invariant patterns in dynamic graphs.

- Our method can accurately capture invariant patterns in dynamic graphs to consistently remove the impact of variant patterns under distribution shifts. For the real-world datasets, we can find consistent performance improvements. And for the synthetic datasets, we can also observe that our method achieves the most stable performances as the shift level increases, while almost all baselines increase in train results and decline in test results. It verifies that the existing baselines can easily exploit variant patterns for predictions and suffer from their harmful effects on OOD generalization.

5.3 Ablation Studies

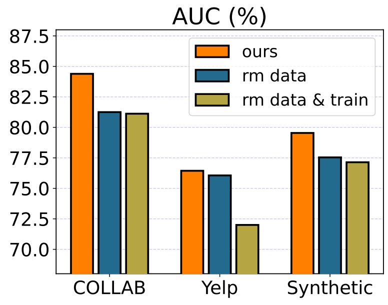

Figure 2: The ablation study result of our method compared to our method without dynamic curriculum importance sample reweighting, and our method without dynamic curriculum importance sample reweighting as well as self-paced curriculum intervention.

We first remove the dynamic curriculum importance sample reweighting module. And we can observe that the performance gained a sharp drop without our novel sample reweighting method, demonstrating its effectiveness.

Next, we also consider removing the self-paced curriculum intervention as well as the dynamic curriculum importance sample reweighting. And we can observe that the performance drop without our novel training scheduler method, which indicates the effectiveness of our designs.

5.4 Complexity Analysis

Here we will analyze our proposed CDGIL method's computational complexity. Let $|V|$ and $|E|$ denote the number of total nodes and edges, respectively. Let d denote the dimension of the hidden representation. From (Zhang et al., 2022), we learn that the computational complexity of the disentangled DGNN is $O(|E|\, d + |V|\, d^2)$. Further, let $|E_t|$ denote the number of edges to train and test, for each training step, the computation cost of curriculum learning for data is only several times of $|E_t|$, or $O(|E_t|)$. And the computation cost of self-adaptive curriculum learning for intervention is $|E|\, d$, where we conduct intervention operation on the whole graph. What's more, we can also find that $O(|E_t|)$ is dominated by $O(|E|)$. Finally, we get our result: the computational complexity of our proposed CDGIL is $O(|E|\, d + |V|\, d^2)$, less than DIDA's, which is $O(|E|\, d + |V|\, d^2 + |E_t|\, |S|\, d)$, and our method outperforms DIDA largely in all of the datasets.

6 Conclusion

In this paper, we propose Curriculum Dynamic Graph Invariant Learning (CDGIL) to handle the dynamic graph distribution problem. More specifically, we present to capture invariant and variant patterns directed by our proposed curriculum learning method considering both training status and sample importance. Firstly, we utilize dual disentangled dynamic attention networks to capture invariant and variant patterns and optimize them separately. Then, we conduct self-paced curriculum intervention to generalize better. Finally, we propose to compute the weight of data samples by evaluating their time series weight as well as sample importance weight. Extensive experiments and ablation studies further demonstrate the effectiveness of our proposed method.

REFERENCES

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Richard A Berk. An introduction to sample selection bias in sociological data. *American sociological review*, pp. 386–398, 1983.

Stefan Braun, Daniel Neil, and Shih-Chii Liu. A curriculum learning method for improved noise robustness in automatic speech recognition. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 548–552. IEEE, 2017.

Taotao Cai, Shuiqiao Yang, Jianxin Li, Quan Z Sheng, Jian Yang, Xin Wang, Wei Emma Zhang, and Longxiang Gao. Incremental graph computation: Anchored vertex tracking in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*, 2016.

Weilin Cong, Yanhong Wu, Yuandong Tian, Mengting Gu, Yinglong Xia, Mehrdad Mahdavi, and Chun-cheng Jason Chen. Dynamic graph representation learning via graph transformer networks. *arXiv preprint arXiv:2111.10447*, 2021.

Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 890–897, 2019.

Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. Rethinking importance weighting for deep learning under distribution shift. *Advances in neural information processing systems*, 33: 11996–12007, 2020.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. Ics-gnn: lightweight interactive community search via graph neural network. *Proceedings of the VLDB Endowment*, 14(6):1006–1018, 2021.

Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2021.

Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

Hong Huang, Zixuan Fang, Xiao Wang, Youshan Miao, and Hai Jin. Motif-preserving temporal network embedding. In *IJCAI*, pp. 1237–1243, 2020a.

Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: adaptive curriculum learning loss for deep face recognition. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5901–5910, 2020b.

Tian Jin, Qiong Wu, Xuan Ou, and Jianjun Yu. Community detection and co-author recommendation in co-author networks. *International Journal of Machine Learning and Cybernetics*, 12:597–609, 2021.

Tae-Hoon Kim and Jonghyun Choi. Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904*, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 2022a.

Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning invariant graph representations for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, 2022b.

Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

William Lotter, Greg Sorensen, and David Cox. A multi-scale cnn and curriculum learning strategy for mammogram classification. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pp. 169–177. Springer, 2017.

Diego C Nascimento, Bruno A Pimentel, Renata MCR Souza, Lilia Costa, Sandro Gonçalves, and Francisco Louzada. Dynamic graph in a symbolic data framework: An account of the causal relation using covid-19 reports and some reflections on the financial world. *Chaos, Solitons & Fractals*, 153:111440, 2021.

George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. United we stand: Transfer graph neural networks for pandemic forecasting. 2020.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5363–5370, 2020.

Zhenyu Qiu, Wenbin Hu, Jia Wu, Weiwei Liu, Bo Du, and Xiaohua Jia. Temporal network embedding with high-order nonlinear information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5436–5443, 2020.

Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020a.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020b.

Benedek Rozemberczki, Paul Scherer, Oliver Kiss, Rik Sarkar, and Tamas Ferenci. Chickenpox cases in hungary: a benchmark dataset for spatiotemporal signal processing with graph neural networks. *arXiv preprint arXiv:2102.08100*, 2021.

Ginny Russell, William Mandy, Daisy Elliott, Rhianna White, Tom Pittwood, and Tamsin Ford. Selection bias on intellectual ability in autism research: A cross-sectional review and meta-analysis. *Molecular autism*, 10(1):1–10, 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*, pp. 519–527, 2020.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, pp. 362–373. Springer, 2018a.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, pp. 362–373. Springer, 2018b.

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.

Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*, pp. 555–563, 2019.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Hao Peng, Sen Su, and S Yu Philip. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4375–4383, 2021.

Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1285–1293, 2012.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Andrew Z Wang, Rex Ying, Pan Li, Nikhil Rao, Karthik Subbian, and Jure Leskovec. Bipartite dynamic representations for abuse detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3638–3648, 2021a.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2021b.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*, 2021c.

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, pp. 5238–5246. PMLR, 2018.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1975–1985, 2021.

G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Zhou Qin, and Wenwu Zhu. Dynamic graph neural networks under spatio-temporal distribution shift. In *Advances in Neural Information Processing Systems*, 2022.

Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34:27965–27977, 2021.

Yuecai Zhu, Fuyuan Lyu, Chengming Hu, Xi Chen, and Xue Liu. Learnable encoder-decoder architecture for dynamic graph: A survey. *arXiv preprint arXiv:2203.10480*, 2022.

A Notations

Notations	Descriptions
$\mathbf{G} = (\mathbf{V}, \mathbf{E})$	A graph consists of its node set as well as its edge set
$\mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t)$	Graph at time t
$\mathbf{Y}_{t+1}$	The feature labels at time $t+1$
$\mathbf{G}_{1:t}$	The set of graph from time 1 to time t
p_{train}	The distribution in training set
p_{test}	The distribution in testing set
y_{t+1}	A feature label at time $t+1$
$\mathbf{y}_{t+1}$	Feature labels at time $t+1$
$f_\theta(\cdot)$	The prediction function
$N_t(v)$	The neighborhood of node v at time t
$\widetilde{N}_t(v)$	The neighborhood of node v from time 1 to time t
$TE(\cdot)$	The time encoding
$\mathbf{z}_v^t$	The embedding of node v at time t
$\mathbf{q}, \mathbf{k}, \mathbf{v}$	The query, key and value vector
$\mathbf{z}_I^t(v), \mathbf{z}_V^t(v)$	The invariant and variant patterns of node v at time t
$\mathbf{m}_f$	The learnable mask for invariant pattern
ℓ_I, ℓ_V	Invariant loss and variant loss
$\mathbf{w}_{time}, \mathbf{w}_{imp}$	Weight of time and weight of importance

B Reproducibility Details

B.1 Dataset Details

We provide the statistic of our datasets in Table 3.

Table 3: Statistics of dynamic graph datasets.

Dataset	#Timestamps	#Nodes	#Edges	Time Interval	Node Feature Dimension
COLLAB	16	23035	151790	year	32
Yelp	24	13095	65375	month	32
Synthetic	16	23035	151790	year	64

For Collab and Yelp datasets, we adopt word2vec**?** to extract a feature vector with a length of 32 from each paper abstract or business review and use the mean pooling to get the feature of an author or a user as the node feature of dynamic graphs. And for the Synthetic dataset, we add the node feature computed in Section 5.1 to concat with the original node feature while inheriting the topology of the original dynamic graph, thus the length of the node feature is 64. And for Collab and Synthetic datasets, we split the 16 timestamps into 3 groups: 10, 1, and 5 for training, validation, and testing separately. For the Yelp dataset, we split the 24 timestamps into 3 groups: 20, 2, and 8 for training, validation, and testing separately.

B.2 **CDGIL** Details

Basic Details. We set the dimension of the hidden layer to 16, and the number of layers to 2. As for the optimizer, we adopt Adam**?** with a learning rate 0.01 and weight decay 5×10^{-7}. The number of total epochs is set to 1000, and we also adopt an early stop method. And for evaluation, we adopt random negative sampling to get negative edges between nodes without edges, where we set the same random seed to guarantee fairness. We predict the link by multiplying the two learned nodes embedding together. And the evaluation metric is the area under the roc curve (AUCROC).

Curriculum Details. For self-paced curriculum intervention, the intensity parameter λ_r is decided by the λ_0 as well as the loss ℓ. For all datasets, ℓ_{min} is set to 0.1, λ, and λ_0 is set to 0.015 for

COLLAB and 0.005 for others. And for the variant decoder, λ_{diff} is also set to 0.05. For time serial reweighting, λ is set to 0.001. While for sample importance reweighting, we set the parameter $A = 0.8$, $B = 0$, $C = -0.0004$, and $D = -9 + \lambda_D * epoch - \log(\lambda_{imp})$, where epoch refers to the corresponding epoch, $\lambda_D = 0.004$, and λ_{imp} is the parameter multiplied to $\mathbf{w}_{imp}$, which is set to 0.15 for COLLAB dataset, and 0.05 for Yelp and Synthetic datasets.

B.3 Configurations

All of our experiments are conducted in the following environments:

- **Operating System.** Ubuntu 20.04.5 LTS
- **CPU.** Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz
- **GPU.** NVIDIA GeForce RTX 3090
- **Software.** Python 3.9.13, Cuda 11.7, PyTorch 1.11.0, PyTorch Geometric 2.0.4

C Additional Experiments

C.1 Hyperparameter Sensitivity

We carry out the hyperparameter sensitivity experiments. We can see that, either the hyperparameter is too big or too small will bring a performance drop, which indicates that too small will lose its effect, while too big will break the normal data weight.

Dataset	0.03	0.06	0.3	0.6	Best baseline
COLLAB	81.49 ± 0.77	83.83 ± 0.40	84.25 ± 0.30	83.69 ± 0.44	81.87 ± 0.40

Table 4: Hyperparameter sensitivity experiments on λ_{imp}.

Dataset	0.01	0.02	0.1	0.2	Best baseline
Yelp	76.84 ± 1.05	75.13 ± 1.95	76.41 ± 1.23	76.50 ± 2.09	75.92 ± 0.90
Synthetic	86.00 ± 0.32	85.64 ± 0.43	86.60 ± 0.48	86.98 ± 0.29	85.20 ± 0.84

Table 5: Hyperparameter sensitivity experiments on λ_{imp}.

Dataset	0.6	0.7	0.8	0.9	Best baseline
COLLAB	84.42 ± 0.26	84.60 ± 0.16	84.40 ± 0.43	83.24 ± 0.78	81.87 ± 0.40

Table 6: Hyperparameter sensitivity experiments on A.

C.2 Extensive Experiments

To validate the effectiveness of dual disentangled dynamic attention network, we compare its experimental result with disentangled dynamic attention network. Results are shown in Table 9.

To validate the effectiveness of CDGIL, we further add several representative baselines: GCN (Kipf & Welling, 2016a), GIN (Xu et al., 2018), GAT (Veličković et al., 2017), GraphSage (Hamilton et al., 2017), EERM (Wu et al., 2022), SR-GNN (Zhu et al., 2021), and results are shown in Table 10

Dataset	-0.0003	-0.0004	-0.0005	Best baseline
COLLAB	84.54 ± 0.27	84.35 ± 0.30	84.53 ± 0.39	81.87 ± 0.40

Table 7: Hyperparameter sensitivity experiments on C.

Dataset	0.003	0.004	0.005	Best baseline
COLLAB	84.54 ± 0.27	84.35 ± 0.30	84.53 ± 0.39	81.87 ± 0.40

Table 8: Hyperparameter sensitivity experiments on λ_D.

Dataset	D3GAT	DDGAT
COLLAB	78.28 ± 0.40	77.25 ± 0.36
Yelp	66.51 ± 8.06	64.66 ± 5.28

Table 9: Comparison experiment between D3GAT and DDGAT.

Dataset	COLLAB	COLLAB	Yelp	Yelp
Test Data	w/o DS	w/ DS	w/o DS	w/ DS
GCN	79.20 ± 0.08	69.24 ± 0.20	56.99 ± 0.18	64.18 ± 0.61
GIN	66.55 ± 0.32	60.65 ± 0.47	68.97 ± 0.95	67.11 ± 5.21
GAT	71.66 ± 0.02	65.79 ± 0.11	60.16 ± 0.78	65.14 ± 2.38
Sage	76.49 ± 0.30	66.19 ± 0.56	69.30 ± 0.18	68.67 ± 0.31
EERM	OOM	OOM	73.01 ± 3.99	69.92 ± 9.59
SR-GNN	79.36 ± 0.13	69.65 ± 0.47	70.41 ± 0.52	69.31 ± 0.30
Ours	93.60 ± 0.11	84.39 ± 0.54	77.15 ± 1.54	76.44 ± 1.79

Table 10: Comparison experiment between D3GAT and DDGAT.