Evolving Graph Generalization Estimation via Self-Supervised Learning

Bin Lu[†], **Tingyan Ma**[†], **Xiaoying Gan**[†], **Luoyi Fu**[†]
Xinbing Wang[†], **Chenghu Zhou**[◇], **Shiyu Liang**[†*]
[†]Shanghai Jiao Tong University,
[◇]Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences
{robinlu1209,xiaokeaiyan,ganxiaoying,yiluofu,xwang8,lsy18602808513}@sjtu.edu.cn
zhouch@lreis.ac.cn

Abstract

Graph Neural Networks are widely deployed in vast fields, but they often struggle to maintain accurate representations as graphs evolve. We theoretically establish a lower bound, proving that under mild conditions, representation distortion inevitably occurs over time. To estimate the temporal representation distortion without human annotation after deployment, one naive approach is to pre-train a recurrent model before deployment and use this model afterwards, but the estimation is far from satisfactory. In this paper, we analyze the representation distortion from an information theory perspective, and attribute it primarily to inaccurate feature extraction during evolution. Consequently, we introduce SMART, a straightforward and effective baseline enhanced by an adaptive feature extractor through self-supervised graph reconstruction. Experimental results on real-world evolving graphs demonstrate our outstanding performance, especially the necessity of self-supervised graph reconstruction. For example, on OGB-arXiv dataset, the estimation metric MAPE deteriorates from 2.19% to 8.00% without reconstruction.

1 Introduction

The rapid rising of Graph Neural Network (GNN) leads to widely deployment in various applications, e.g. social network, smart cities, drug discovery[1, 2, 3]. However, recent studies have uncovered a notable challenge: as the distribution of the graph shifts continuously after deployment, GNNs may suffer from the representation distortion over time, which further leads to continuing performance degradation[4, 5, 6], as shown in Figure 1.

Consequently, a practical and urgent need is to monitor the representation distortion of GNN. An obvious method is to regularly label and test online data. However, constant human annotation is difficult to withstand the rapidly ever-growing evolution of graph after deployment. Therefore, how to proactively estimate the temporal generalization performance without annotation after deployment is a challenging problem.

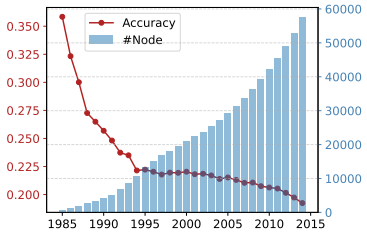


Figure 1: GNN performance continues to decline with the rapid growth of the evolving graph.

To solve this problem, a naive way is to collect the generalization changes through partially-observed labels before deployment, and train a recurrent neural network (RNN) [7] in a supervised manner. However, existing studies [8, 9] have shown that RNN itself have insufficient representation power, thereby usually concatenating a well-designed feature extrac-

[*]Shiyu Liang is the corresponding author.

tor. Unluckily, the representation distortion of static feature extractor still suffers during evolution. Hence, to deal with this problem, we propose SMART (Self-supervised teMporAl geneRalization esTimation). Since it is hard to gather label information in a rapid graph evolution after deployment, our SMART resorts to an adaptive feature extractor through self-supervised graph feature and structure reconstruction, eliminating the information gap due to the evolving distribution drift.

2 Problem Formulation

Before the deployment at time t_{deploy}, we are given with a pre-trained graph neural network G that outputs a label for each node in the graph, based on the graph adjacency matrix $A_k \in \{0,1\}^{n_k \times n_k}$ and feature matrix $X_k \in \mathbb{R}^{n_k \times d}$ at each time k. Additionally, we are given an observation set $\mathcal{D} = \{(A_k, X_k, H_k Y_k)\}_{k=0}^{t}$, consisting of (1) a series of fully observed graph adjacency matrices $A_0, \cdots, A_t$ and node feature matrices $X_0, \cdots, X_t$; (2) a series of partially observed label vectors $H_0 Y_0, \cdots, H_t Y_t$, where each observation matrix $H_k \in \{0,1\}^{n_k \times n_k}$ is diagonal. The diagonal element on the i-th row in matrix H_k is non-zero if and only if the label of the node i is observed.

At each time τ after deployment at time t_{deploy}, we aim to predict the expected temporal generalization performance of the model G on the state $\mathcal{G}_\tau = (A_\tau, X_\tau, Y_\tau)$, given a full observation on the adjacency matrix A_τ and feature matrix X_τ. Obviously, the problem is not hard if we have a sufficient amount of labeled samples. However, it is costly to obtain the human annotation on the constant portion of the whole graph after deployment, especially many real-world graphs grow exponentially fast. Therefore, the problem arises if we try to design a post-deployment testing performance predictor $\mathcal{M}$ without further annotations after deployment. Specifically, we theoretically prove that as graph evolves, the representations of GNN undergo inevitable distortion, making the prediction more challenging.

3 Theoretical Analysis

In this subsection, we show that as the graph evolves, the distortion of representation is unavoidable, which might further lead to potential performance damage. Under Assumption 1 (in Appendix A), we consider the pre-trained graph model as a single-layer GCN with Leaky ReLU activation function parameterized as θ. The optimal parameters for the model are θ^*. However, due to the stochastic gradient descent in optimization and quantization of models, we always obtain a sub-optimal parameter drawn from a uniform distribution $U(\theta^*, \xi)$. We define the expected distortion of the model output on node i at time t as the expected difference between the model output at time t and time zero on the node i, i.e., $\ell_t(i) = \mathbb{E}_{\theta, \mathcal{G}_t}\left[|f_t(i;\theta) - f_0(i;\theta)|^2\right]$.

Theorem 1. *If θ is the vectorization of the parameter set $\{(a_j, W_j, b_j)\}_{j=1}^{N}$ and its i-th coordinate θ_i is drawn from the uniform distribution $U(\theta_i^*, \xi)$ centering at the i-th coordinate of the vector θ_i^*, the expected deviation $\ell_\tau(i)$ of the perturbed GCN model at the time $\tau \geq 0$ on the node $i \in \{1, ..., n\}$ is lower bounded by*

$$\ell_\tau(i) \geq \phi_\tau(i) \triangleq \frac{N\beta^2\xi^4}{9}\mathbb{E}\left[\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)^2 \left|\sum_{k \in \mathcal{N}_0(i)} x_k\right|^2\right]$$

where the set $\mathcal{N}_0(i)$ denotes the neighborhood set of the node i at time 0. Additionally, $\phi_\tau(i)$ is strictly increasing with respect to $\tau \geq 1$.

Remark 1. Proofs of Theorem 1 can be found in Appendix A. This theorem shows that for any node i, the expected distortion of the model output is strictly increasing over time, especially when the width N of the model is quite large in the current era of large models. Therefore, accurately estimating the generalization performance changes of GNNs is a necessary and challenging problem.

4 Methodology

Now, we want to construct a temporal generalization loss estimator that takes the adjacency matrices and feature matrices as its input and predicts the difference between the outputs of the graph neural network G and the observed true labels at time k, i.e., $\ell_k = \ell\left(H_k G(A_k, X_k), H_k Y_k\right)$.

In order to capture the temporal variation, we adopt a recurrent neural network-based model $\mathcal{M}(\cdot; \theta_{RNN})$ to estimate generalization loss in the future. The RNN model sequentially takes

the output of the GNN $G(A_k, X_k)$ as its input and outputs the estimation $\hat{\ell}_k$ at time k. To enhance the representation power of RNN, we usually add a non-linear feature extractor φ to capture the principal features in the inputs and the RNN model becomes

$$\begin{bmatrix} \hat{\ell}_k \\ h_k \end{bmatrix} = \begin{bmatrix} \mathcal{M}_\ell \left(\varphi \circ G(A_k, X_k), h_{k-1}\right) \\ \mathcal{M}_h \left(\varphi \circ G(A_k, X_k), h_{k-1}\right) \end{bmatrix},$$

where h_k denotes the hidden state in the RNN, transferring the historical information. After deployment time t_{deploy}, a naive and straightforward way of using the generalization loss estimator is directly applying the model on the graph sequence $(\mathcal{G}_\tau : \tau > t_{\text{deploy}})$. This results in a population loss prediction error $\mathcal{E}_\tau$ on the optimal trained model $\mathcal{M}^*, \varphi^*$ at time τ given by

$$\mathcal{E}_\tau(\mathcal{M}^*, \varphi^*) = \mathbb{E}\|\mathcal{M}_\ell^* \left(\varphi^* \circ G(A_k, X_k), h_{k-1}\right) - \ell\left(G(A_k, X_k), Y_k\right)\|^2$$

Before deployment, however, we are not able to get sufficient training frames. This indicates φ^* may only have good feature extraction performance on graphs similar to the first several graphs. After deployment, the graphs undergo significant changes, and thereby have unavoidable representation distortion, which makes the generalization estimator perform worse and worse.

To further investigate this problem, let us consider the information loss within the input graph series and the output prediction after the GNN is deployed,

$$\text{Information Loss } \triangleq I(\{(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k) - I(\hat{\ell}_k; \ell_k),$$

where $I(;)$ is the mutual information of two variables. The learning process is equivalent to minimizing the above information loss. Furthermore, it can be divided into two parts:

$$\begin{aligned} \text{Information Loss } &= \underbrace{I(\{\varphi \circ G(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k) - I(\hat{\ell}_k; \ell_k)}_{\text{① Information Loss Induced by RNN}} \\ &+ \underbrace{I(\{G(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k) - I(\{\varphi \circ G(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k)}_{\text{② Information Loss Induced by Representation Distortion}}. \end{aligned} \tag{1}$$

The second part indicates the information loss by the representation distortion of φ. Especially as the graph evolve over time, the information loss correspondingly increases due to the distribution shift. According to the data-processing inequality [10] in information theory, post-processing cannot increase information. Therefore, the second part of Equation 1 holds for any time τ,

$$I(\{G(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k) - I(\{\varphi \circ G(A_\tau, X_\tau)\}_{\tau=t_{\text{deploy}}+1}^{k}, \mathcal{D}; \ell_k) \geq 0.$$

The equality holds if and only if $\varphi \circ G(A_\tau, X_\tau)$ is a one-to-one mapping with $G(A_\tau, X_\tau)$. In other words, there exists an inverse function φ^{-1} to reconstruct the features and structures of graph over the time, such that $\varphi^{-1} \circ \varphi \circ G(A_\tau, X_\tau) = G(A_\tau, X_\tau)$.

Therefore, after deployment, since there are no human-annotated label, we design the *contrastive graph reconstruction* to obtain self-supervised signals to finetune the adaptive feature extractor φ at post-deployment time, thereby reducing the information loss during the dynamic graph evolution. Compared with directly graph reconstruction, contrastive method generates more views through data augmentation, which can learn more principal and robust representations.

Given the pre-trained graph model G and evolving graph (A_k, X_k) at time k, we first obtain the feature embedding matrix $O_k = G(A_k, X_k)$. In order to capture the evolution property of graphs, we define two contrastive loss on augmented feature graph, which is denoted as $(\mathcal{T}(A_k), O_k)$, where $\mathcal{T}(\cdot)$ is the structure transformation function, i.e. randomly add or drop edges [11, 12, 13].

Structure Reconstruction Loss $\mathcal{L}_s$. First of all, we define the structure reconstruction loss $\mathcal{L}_s$. Given the augmented feature graph $(\mathcal{T}(A_k), O_k)$, we perform reconstruction on adjacency matrix A_k. Specifically, it computes the reconstructed adjacency matrix $\hat{A}_k$ by $\hat{A}_k = \sigma(F_k F_k^T), F_k = \varphi(\mathcal{T}(A_k), O_k)$, where $F_k \in \mathbb{R}^{N \times B}$, σ is the sigmoid activation function. Here we utilize one-layer graph attention network (GAT) as model φ [14], and it is optimized by binary cross entropy loss between A_k and $\hat{A}_k$ as a link prediction task, i.e. $\mathcal{L}_s = \mathcal{L}_{\text{BCE}}(A_k, \hat{A}_k)$.

Feature Reconstruction Loss $\mathcal{L}_f$. Moreover, we perform the node-level feature reconstruction on the corrupted adjacency matrix $\mathcal{T}(A_k)$. We utilize a single-layer decoder a to obtain the reconstructed

feature $\hat{O}_k$ by $\hat{O}_k = F_k a^T, F_k = \varphi(\mathcal{T}(A_k), O_k)$, where $a \in \mathbb{R}^{B\times B}$, and it is optimized by mean squared error loss $\mathcal{L}_f = \|O_k - \hat{O}_k\|^2$.

To sum up, the reconstruction loss $\mathcal{L}_g$ is the composition of structure reconstruction loss $\mathcal{L}_s$ and feature reconstruction loss $\mathcal{L}_f$ as $\mathcal{L}_g(\varphi) = \lambda\mathcal{L}_s + (1-\lambda)\mathcal{L}_f$, where λ is the proportional weight ratio to balance two loss functions. By the way, before the deployment, we conduct the same graph reconstruction to improve the performance of feature extraction with the supervised learning. Algorithm 1 outlines the pre-deployment training and post-deployment finetuning of SMART in detail.

5 Experiment

In this section, to validate the effectiveness of our proposed SMART, we conduct experiments on four real-world evolving network datasets. Meanwhile, due to space constraints, we present the experimental results and theoretical analysis on Barabási-Albert random graphs in Appendix C.

Experiment Setting. We use four evolving graph datasets for evaluation: OGB-arXiv [15], DBLP [16], Pharmabio [16], Facebook 100 [17]. Mean Absolute Percentage Error (MAPE) and Standard Error are utilized for evaluations. The detailed experiment setting is shown in Appendix F.

Comparison with Linear Regression. Experimental observations reveal that the degradation of many GNN models exhibits an approximate linearity. Therefore, we compare our model SMART with linear regression on three academic datasets and three different GNN backbones as shown in Table 1. We observe a strikingly prediction improvement. For example, in Pharmabio dataset, our SMART on GCN decreases MAPE from around 32.4% to around 1.3% compared to linear regression. Additional experimental results are presented in Appendix G.

Table 1: Performance comparison on three academic network datasets and three GNN backbones. We use MAPE ± Standard Error to evaluate the estimation on different scenarios.

Dataset	OGB-arXiv (↓)		DBLP (↓)		Pharmabio (↓)	
GNN Backbone	Linear	SMART	Linear	SMART	Linear	SMART
GCN [18]	10.5224	2.1897±0.2211	16.4991	3.4992±0.1502	32.3653	1.3405±0.2674
GAT [14]	12.3652	3.1481±0.4079	17.6388	6.6459±1.3401	29.0404	1.2197±0.2241
GraphSage [19]	19.5480	5.2733±2.2635	23.7363	9.9651±1.4699	31.7033	3.1448±0.6875

Ablation Study. To verify the effectiveness of different modules in SMART, we conducted ablation studies on four datasets with the four variants as shown in Figure 2. (1) Comparing (M1) with our method, contrastive graph reconstruction significantly impacts accurate generalization estimation, particularly evident in the OGB-arXiv and Pharmabio datasets, where the (M1) variant exhibits a substantial performance gap. (2) In the case of (M2), generalization estimation based on historical multi-step information improves prediction accuracy and stability. For instance, in the Cornell dataset, predictions using single-step information result in a larger standard error. (3) As shown by (M3a) and (M3b), removing either of the reconstruction losses leads to a performance decrease in SMART. Since evolving graphs display temporal drift in both structure and features, both graph contrastive reconstruction losses are essential for mitigating information loss over time.

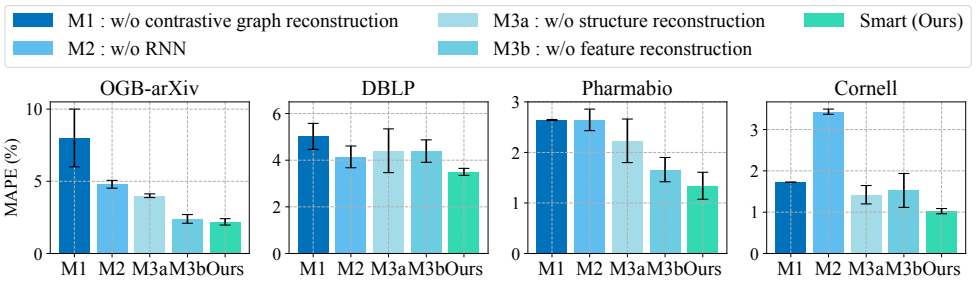


Figure 2: Ablation study on four representative evolving datasets.

6 Conclusions

In this paper, we investigate a practical but underexplored problem of temporal generalization estimation in evolving graph. To this end, we theoretically show that the representation distortion is

unavoidable and further propose a straightforward and effective baseline SMART. Experiments on real-world datasets demonstrate the effectiveness of our methods and verify the importance of self-supervised graph reconstruction. Future work involves exploring our methods in more complicated heterogeneous graphs and spatio-temporal graphs.

Acknowledgement

This work was supported by National Key R&D Program of China (No.2022YFB3904204), NSF China (No.62272301, 42050105, 62020106005, 62061146002, 61960206002), and the Deep-time Digital Earth (DDE) Big Science Program.

References

[1] Carl Yang and Jiawei Han. Revisiting citation prediction with cluster-aware text-enhanced heterogeneous graph neural networks. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*, pages 682–695. IEEE, 2023.

[2] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. Spatio-temporal graph few-shot learning with cross-city knowledge transfer. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 1162–1172. ACM, 2022.

[3] Minkai Xu, Alexander S. Powers, Ron O. Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38592–38610. PMLR, 2023.

[4] 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*, 2022.

[5] 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.

[6] Bin Lu, Xiaoying Gan, Ze Zhao, Shiyu Liang, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. Graph out-of-distribution generalization with controllable data augmentation. *CoRR*, abs/2308.08344, 2023.

[7] Tomás Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA, 2010.

[8] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[9] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5363–5370. AAAI Press, 2020.

[10] Normand J. Beaudry and Renato Renner. An intuitive proof of the data processing inequality. 12(5–6):432–441, may 2012.

[11] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[13] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(6):5879–5900, 2023.

[14] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[15] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[16] Lukas Galke, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. Lifelong learning of graph neural networks for open-world node classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[17] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20887–20902, 2021.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[19] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[20] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[21] Albert-László Barabási. Scale-free networks: a decade and beyond. *Science*, 325(5939):412–413, 2009.

[22] Niema Moshiri. The dual-barabási-albert model. *arXiv preprint arXiv:1810.10538*, 2018.

A Proof for Theorem 1

Before presenting the results, we first present the following several assumptions.

Assumption 1 (Graph Evolution Process)**.** *The initial graph $\mathcal{G}_0 = (A_0, X_0, Y_0)$ has n nodes. (1) We assume that the feature matrix X_0 is drawn from a continuous probability distribution supported on $\mathbb{R}^{n\times d}$. (2) At each time t in the process, a new node indexed by $n+t$ appears in the graph. We assume that this node connects with each node in the existing graph with a positive probability and that edges in the graph do not vanish in the process. (3) We assume that the feature vector x_{n+t} has a zero mean conditioned on all previous graph states, i.e., $\mathbb{E}[x_{n+t}|\mathcal{G}_0, ..., \mathcal{G}_{t-1}] = \mathbf{0}_d$ for all $t \geq 1$.*

Remark 1. Assumption 1 states the following: (1) For any given subspace in the continuous probability distribution, the probability that X_0 appears is zero. (2) The ever-growing graph assumption is common in both random graph analysis, like the Barabási-Albert graph [20], and real-world scenarios. For example, in a citation network, a paper may have a citation relationships with other papers in the network, and this relationships will not disappear over time. Similarly, the purchasing relationships between users and products in e-commerce trading networks are the same. (3) The zero-mean assumption always holds, as it is a convention in deep learning to normalize the features.

Next, we present the proof for Theorem 1 as follows.

Proof. Let $f_\tau(i;\theta)$ denote the output of the GCN on the node i at time $\tau \geq 0$. Therefore, we have

$$f_\tau(i;\theta) = \sum_{j=1}^{N} a_j \sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right). \tag{2}$$

Thus, the expected loss of the parameter θ^* on the node i at time τ is

$$\begin{aligned}
\ell_\tau(i) &= \mathbb{E}\left[\left(f_\tau(i;\theta) - f_0(i;\theta)\right)^2\right] \\
&= \mathbb{E}\left[\left|\sum_{j=1}^{N} a_j \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right].
\end{aligned}$$

Furthermore, recall that each parameter $a_j \sim U(a_j^*, \xi)$ and each element $W_{j,k}$ in the weight vector W_j also satisfies $W_{j,k} \sim U(W_j^*, \xi)$. Therefore, the differences $a_j - a_j^*$ and $W_{j,k} - W_{j,k}^*$ are all i.i.d. random variables drawn from distribution $U(0,\xi)$. Therefore, we have

$$\begin{aligned}
\ell_\tau(i) &= \mathbb{E}\left[\left|\sum_{j=1}^{N} a_j \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right] \\
&= \mathbb{E}\left[\left|\sum_{j=1}^{N} (a_j - a_j^*) \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right.\right. \\
&\qquad \left.\left. + \sum_{j=1}^{N} a_j^* \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right] \\
&= \mathbb{E}\left[\left|\sum_{j=1}^{N} (a_j - a_j^*) \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right] \\
&\quad + \mathbb{E}\left[\left|\sum_{j=1}^{N} a_j^* \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right]
\end{aligned}$$

The third equality holds by the fact that the differences $(a_j - a_j^*)$'s are all i.i.d. random variables drawn from the uniform distribution $U(0,\xi)$. Therefore, we have

$$\ell_\tau(i) \geq \mathbb{E}\left[\left|\sum_{j=1}^{N} (a_j - a_j^*) \left(\sigma\left(\frac{1}{d_\tau(i)} \sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)} \sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right].$$

Furthermore, since the differences $(a_j - a_j^*)$ are i.i.d. random variable drawn from the distribution $U(0, \xi)$, we must further have

$$\begin{aligned}
\ell_\tau(i) &\geq \mathbb{E}\left[\left|\sum_{j=1}^{N}(a_j - a_j^*)\left(\sigma\left(\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)}\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right)\right|^2\right] \\
&= \mathbb{E}\left[\sum_{j=1}^{N}\mathbb{E}[(a_j - a_j^*)^2|A_\tau, X_\tau]\left|\sigma\left(\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)}\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right|^2\right] \\
&= \frac{\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\sigma\left(\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)}\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right|^2\right].
\end{aligned}$$

Furthermore, the leaky ReLU satisfies that $|\sigma(u) - \sigma(v)| \geq \beta|u - v|$. The above inequality further implies

$$\begin{aligned}
\ell_\tau(i) &\geq \frac{\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\sigma\left(\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j\right) - \sigma\left(\frac{1}{d_0(i)}\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j + b_j\right)\right|^2\right] \\
&\geq \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)} x_k^\top W_j + b_j - \frac{1}{d_0(i)}\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j - b_j\right|^2\right] \\
&\geq \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)\setminus\mathcal{N}_0(i)} x_k^\top W_j + \left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j\right|^2\right] \\
&\geq \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\frac{1}{d_\tau(i)}\sum_{k\in\mathcal{N}_\tau(i)\setminus\mathcal{N}_0(i)} x_k^\top W_j\right|^2 + \left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j\right|^2\right] \\
&\geq \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j\right|^2\right]
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\ell_\tau(i) &\geq \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j\right|^2\right] \\
&= \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top (W_j - W_j^* + W_j^*)\right|^2\right] \\
&= \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top (W_j - W_j^*)\right|^2\right] \\
&\quad + \frac{\beta^2\xi^2}{3}\mathbb{E}\left[\sum_{j=1}^{N}\left|\left(\frac{1}{d_\tau(i)} - \frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top W_j^*\right|^2\right],
\end{aligned}$$

where the last equality comes from the fact that random vectors $(W_j - W_j^*)$'s are an i.i.d. random variables drawn from the uniform distribution $U(0, \xi)$ and are also independent of the graph evolution

process. Therefore, we have

$$\ell_\tau(i) \geq \frac{N\beta^2\xi^2}{3}\mathbb{E}\left[\left|\left(\frac{1}{d_\tau(i)}-\frac{1}{d_0(i)}\right)\sum_{k\in\mathcal{N}_0(i)} x_k^\top(W_j - W_j^*)\right|^2\right]$$
$$= \frac{N\beta^2\xi^4}{9}\mathbb{E}\left[\left(\frac{1}{d_\tau(i)}-\frac{1}{d_0(i)}\right)^2\left\|\sum_{k\in\mathcal{N}_0(i)} x_k\right\|^2\right],$$

where the last equality comes from the fact that each element in the random vector $(W_j - W_j^*)$ is i.i.d. random variable drawn from the uniform distribution $U(0,\xi)$. Since the initial feature matrix $X_0 = (x_1, ..., x_n)$ are drawn from a continuous distribution supported on $\mathbb{R}^d$, we must have with probability one,

$$\left\|\sum_{k\in\mathcal{N}_0(i)} x_k\right\|^2 > 0.$$

Furthermore, we have

$$\mathbb{E}_{\mathcal{G}_\tau}\left[\left(\frac{1}{d_\tau(i)}-\frac{1}{d_0(i)}\right)^2\middle|\mathcal{G}_0\right] \geq \left(\mathbb{E}_{\mathcal{G}_\tau}\left[\frac{1}{d_\tau(i)}\middle|\mathcal{G}_0\right]-\frac{1}{d_0(i)}\right)^2$$
$$= \left(\frac{1}{d_0(i)}-\mathbb{E}_{\mathcal{G}_\tau}\left[\frac{1}{d_\tau(i)}\middle|\mathcal{G}_0\right]\right)^2$$

To prove $\phi_\tau(i)$ is strictly increasing, it suffices to prove that $\mathbb{E}_{\mathcal{G}_\tau}\left[\frac{1}{d_\tau(i)}\middle|\mathcal{G}_0\right]$ is decreasing with respect to τ. Since

$$\mathbb{E}_{\mathcal{G}_\tau}\left[\frac{1}{d_\tau(i)}\middle|\mathcal{G}_0\right] = \mathbb{E}_{\mathcal{G}_\tau}\left[\frac{1}{d_\tau(i)}\middle|\mathcal{G}_0\right]$$
$$= \int_0^\infty \mathbb{P}\left(\frac{1}{d_\tau(i)} > r\middle|\mathcal{G}_0\right)dr$$
$$= \int_0^\infty \mathbb{P}\left(d_\tau(i) < \frac{1}{r}\middle|\mathcal{G}_0\right)dr$$
$$< \int_0^\infty \mathbb{P}\left(d_{\tau-1}(i) < \frac{1}{r}\middle|\mathcal{G}_0\right)dr,$$

where the last inequality comes from the fact that

$$\mathbb{P}\left(d_\tau(i) < \frac{1}{r}\middle|\mathcal{G}_0\right) < \mathbb{P}\left(d_{\tau-1}(i) < \frac{1}{r}\middle|\mathcal{G}_0\right).$$

□

B SMART Algorithm

We illustrate the details of our proposed SMART in Algorithm 1. The learning process of SMART is divided into two stage: pre-deployment warmup training and post-deployment finetuning. Before the deployment, we conduct both the supervised learning on the generalization loss prediction with the partially-observed labels and self-supervised learning on the evolving graph structure. After the deployment, since we no longer have the label information over time, we design two contrastive graph reconstruction task as a self-supervised manner to actively finetuning the feature extractor φ.

Algorithm 1 SMART: Self-supervised Temporal Generalization Estimation

Require: Pre-trained graph model G, observation data $\mathcal{D}$, evolving graph (A_k, X_k) at time k.
Ensure: Update generalization performance predictor $\mathcal{M}$ and φ with paramter $\theta = \left[\theta^{\mathcal{M}}, \theta^{\varphi}\right]$.

while not converged or maximum epochs not reached **do** ▷ Pre-deployment Warmup Training
  Compute loss $\mathcal{L}(\theta) = \sum_{\tau=0}^{t_{\text{deploy}}} \| \mathcal{M}_\ell \left(\varphi \circ G(A_\tau, X_\tau), h_{\tau-1}\right) - \ell \left(H_\tau G(A_\tau, X_\tau), H_\tau Y_\tau\right) \|^2$;
  Compute graph self-supervised $\mathcal{L}_g$ via two self-supervised graph reconstruction;
  Update $\theta \leftarrow \theta + \alpha \nabla_\theta (\mathcal{L} + \mathcal{L}_g)$;
end while
for $k = t_{\text{deploy}} + 1, \cdots, T$ **do** ▷ Post-deployment Finetuning
  Get the newly-arrival graph (A_k, X_k);
  while not converged or maximum epochs not reached **do**
    Compute graph self-supervised $\mathcal{L}_g$ via two self-supervised graph reconstruction;
    Update $\theta_k^{\varphi} \leftarrow \theta_k^{\varphi} + \beta \nabla_{\theta_k^{\varphi}} \mathcal{L}_g$;
  end while
end for

C A Closer Look at Barabási–Albert Random Graph

To theoretically verify the effectiveness of SMART, we first take a closer look at a specific scenario of the synthetic random graphs $\mathcal{G} = (\mathcal{G}_t : t \in \mathbb{N})$, which follows preferential attachment growth [21], and is also called Barabási–Albert (BA) graph model.

Assumption 2 (Preferential Attachment Evolving Graphs)**.** *Here we consider a node regression task. The initial graph $\mathcal{G}_0$ has $\mathcal{N}_0$ nodes. (1) We assume the node feature matrix X_k is a Gaussian random variable with $\mathcal{N}(0, \mathbf{I}_B)$, where $\mathbf{I}_B \in \mathbb{R}^B$. (2) The node label of each node i is generated by $y_i = d_i^\alpha X_{im}, \alpha \geq 0$, which is satisfied like node degree, closeness centrality coefficients, etc. (3) A single-layer GCN $f(\mathcal{G}) = LXW$ is given as the pre-trained graph model G.*

Theorem 2. *If at each time-slot t, the Barabási–Albert random graph is grown by attaching one new node with m edges that are preferentially attached to existing nodes with high degree. To quantify the performance of GNN, the mean graph-level generalization relative error is determined by*

$$\mathcal{E}_{\mathcal{G}} = 2m \cdot \frac{t}{\mathcal{N}_0 + t} \cdot (C^2 \sum_{d=1}^{+\infty} d^{-2\alpha-4} - 2C \sum_{d=1}^{+\infty} d^{-\alpha-4} + \sum_{d=1}^{+\infty} d^{-3}),$$

where d is the degree of nodes. $C = (\frac{1}{\beta} \sum_{i=1}^{\mathcal{N}_0} d_i^{\alpha-1}) / (\sum_{i=1}^{\mathcal{N}_0} d_i^{-1})$ and d_i is the degree of node v_i.

Remark 2. Proofs of Theorem 2 can be found in the Appendix D. This theorem shows that when the node scale $\mathcal{N}_0$ of initial graph $\mathcal{G}_0$ is quite large, the graph-level error loss is approximately linearly related to time t and continues to deteriorate.

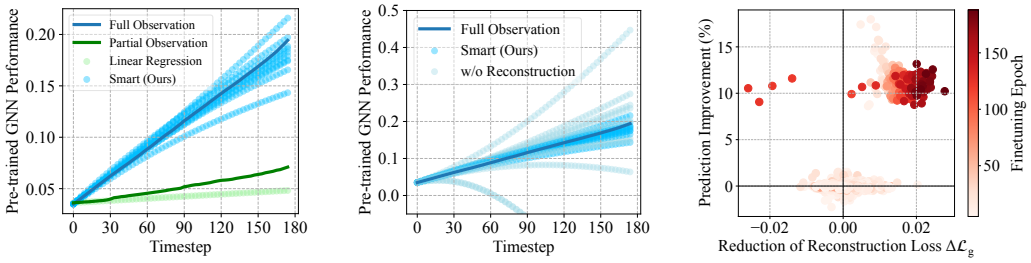
To verify the above propositions, we generate a Barabási–Albert (BA) scale-free model with the following setting: the initial scale of the graph is $\mathcal{N}_0 = 1000$, and the evolution period is 180 timesteps. At each timestep, one vertex is added with $m = 5$ edges. The label of each node is the closeness centrality coefficients. The historical training time period is only 9. As we derived in Theorem 2, the actual graph-level generalization error approximates a linear growth pattern. Therefore, we consider to compare SMART with linear regression model to estimate the generalization performance.

Does Linear Regression Model Work? We conduct experiments with 10 random seeds and present the empirical results in Figure 3a. (1) Generalization error exhibits linear growth, consistent with our theorem (the blue solid line). (2) Our SMART method performs significantly well in a long testing period, with a mean prediction percentage error of 4.2602% and collapses into a roughly linear model. (3) However, the linear regression model, based on the first 9-step partial observations (the green solid line), exhibits extremely poor performance. Due to limited human annotation, partial observations cannot accurately represent the performance degradation of the entire graph. We also adjust parameters in the BA graph model and introduced dual BA graph model [22] for further experiments (see Table 2). Our proposed SMART model effectively captures the evolutionary characteristics across various settings and significantly outperforms the linear regression model.

Effectiveness of Graph Reconstruction. To further validate the effectiveness of graph reconstruction in SMART, we conduct following two experiments. (1) As shown in Figure 3b, we remove the graph

Table 2: Performance comparison on different Barabási–Albert graph setting. We use Mean Absolute Percentage Error (MAPE) ± Standard Error to evaluate the estimation on different scenarios.

	Barabási–Albert ($\mathcal{N}_0 = 1000$)			Dual Barabási–Albert ($\mathcal{N}_0 = 1000, m_1 = 1$)		
	$m = 2$	$m = 5$	$m = 10$	$m_2 = 2$	$m_2 = 5$	$m_2 = 10$
Linear	79.2431	74.1083	82.1677	61.8048	67.6442	38.4884
SMART	**7.1817**±1.2350	**4.2602**±0.5316	**9.1173**±0.1331	**7.9038**±1.8008	**3.8288**±0.1706	**1.9947**±0.1682



(a) Performance comparison with Linear Regression and SMART.

(b) Ablation study of removing graph reconstruction in SMART.

(c) Prediction improvement during the graph reconstruction.

Figure 3: Experimental results of SMART and its variation on BA random graph.

reconstruction module and repeat the experiment with 10 random seeds. Due to the temporal distribution shift caused by the graph evolution, the generalization estimation shows significant deviations and instability. (2) We track the intermediate results during post-deployment fine-tuning, i.e. the reduction of reconstruction loss and prediction improvements. As depicted in Figure 3c, in the early stage of reconstruction (scatter points in light color), the prediction performance optimization is fluctuating. As the optimization continues (scatter points in dark color), the prediction performance is effectively boosted and concentrated in the upper-right corner, with an average performance improvement of 10%.

D Proof for Theorem 2

Proof. Assuming a regression task with a single-layer GCN, we compute mean squared error between prediction and ground truth as the learning objective as follows

$$\min_W \mathbb{E}_X \|LXW - Y\|_2^2 = \min_W \mathbb{E}_X \left((LXW)^T \cdot LXW - 2(LXW)^T Y + \|Y\|_2^2\right) \tag{3}$$

$$= \min_W \mathbb{E}_X \left(W^T X^T L^T LXW - 2W^T X^T L^T Y + \|Y\|_2^2\right). \tag{4}$$

Since D^{-2} is a diagonal matrix, $(L^T L)_{ij} = (A^T D^{-2} A)_{ij} = a_i^T D^{-2} a_j$, where a_i is the i-th row in matrix A. Each node feature is independently Gaussian distributed.

When $i \neq j$,

$$\mathbb{E}_X \left[X^T L^T LX\right]_{ij} = 0 \tag{5}$$

When $i = j$,

$$\mathbb{E}_X \left[X^T L^T LX\right]_{ii} = \mathbb{E}_X \left(x_i^T L^T L x_i\right) = \mathbb{E}_X \left(\sum_{j=1}^{n} \sum_{m=1}^{n} x_{ij} (L^T L)_{jm} x_{mi}\right) \tag{6}$$

Similarly, when and only when $j = m$, $\mathbb{E}_X \left[X^T L^T LX\right]_{ii} \neq 0$

$$\mathbb{E}_X \left[X^T L^T LX\right]_{ii} = \sum_{j=1}^{n} \mathbb{E}_X \left(x_{ij}^2\right) (L^T L)_{jj} = \sum_{j=1}^{n} (L^T L)_{jj} \cdot I \tag{7}$$

where $\sum_{j=1}^{n}(L^TL)_{jj} = \sum_{j=1}^{n}((D^{-1}A)^TD^{-1}A)_{jj} = \sum_{j=1}^{n}(A^TD^{-2}A)_{jj} = \sum_{j=1}^{n}\frac{1}{d_j} \triangleq \beta$

Consequently, the learning objective is equal to

$$\min_W \mathbb{E}_X \|LXW - Y\|_2^2 = \min_W \beta \cdot W^TW - 2W^T\mathbb{E}_X\left(X^TL^TY\right) + \|Y\|^2 \tag{8}$$

Meanwhile, the optimal parameter of GCN equals to $W^* = \frac{1}{\beta}\mathbb{E}_X\left(X^TL^TY\right)$.

$$W_i^* = \frac{1}{\beta}\mathbb{E}_X\left[X^TL^TY\right]_i \tag{9}$$

$$= \frac{1}{\beta}\mathbb{E}_X\sum_{m=1}^{N}(X^TL^T)_{im}Y_m \tag{10}$$

$$= \frac{1}{\beta}\mathbb{E}_X\sum_{m=1}^{N}(LX)_{mi}d_m^{\alpha}X_{mk} \tag{11}$$

$$= \frac{1}{\beta}\mathbb{E}_X\sum_{m=1}^{N}\sum_{s=1}^{N}L_{ms}X_{si}d_m^{\alpha}X_{mk} \tag{12}$$

$$= \frac{1}{\beta}\mathbb{E}_X\sum_{m=1}^{N}L_{mm}X_{mi}d_m^{\alpha}X_{mk} \tag{13}$$

Therefore, only if $i = k$, $W_k^* = \frac{1}{\beta}\sum_{m=1}^{N}L_{mm}d_m^{\alpha} = \frac{1}{\beta}\sum_{m=1}^{N}d_m^{\alpha-1} \triangleq C$, and otherwise $W_i^* = 0$.

For any given node v_i, we have

$$\varepsilon_i = \frac{\mathbb{E}\|(LXW^*)_i - Y_i\|_2^2}{\mathbb{E}\|Y_i\|_2^2} = \frac{\mathbb{E}(LXW^*)_i^2 - 2(LXW^*)_iY_i + \|Y_i\|^2}{\mathbb{E}\|Y_i\|_2^2} \tag{14}$$

To be specific,

$$(LXW^*)_i = \sum_{j=1}^{d}(LX)_{ij}W_j^* = (LX)_{ik}W_k^* = C\sum_{s=1}^{N}L_{is}X_{sk} \tag{15}$$

$$(LXW^*)_iY_i = C\sum_{s=1}^{N}L_{is}X_{sk}d_i^{\alpha}X_{ik} = Cd_i^{\alpha}L_{ii} = Cd_i^{\alpha-1} \tag{16}$$

$$\mathbb{E}(LXW^*)_i^2 = \mathbb{E}[C\sum_{s=1}^{N}L_{is}X_{sk}]^2 = \mathbb{E}[C^2\sum_{s=1}^{N}(L_{is}X_{sk})^2] = C^2\sum_{s=1}^{N}L_{is}^2 = C^2\frac{1}{d_i} \tag{17}$$

$$\mathbb{E}\|Y_i\|^2 = \mathbb{E}d_i^{2\alpha}X_{ik}^2 = d_i^{2\alpha} \tag{18}$$

Therefore, plugging to Eq. 14

$$\varepsilon_i = \frac{C^2d_i^{-1} - 2Cd_i^{\alpha-1} + d_i^{2\alpha}}{d_i^{2\alpha}} \tag{19}$$

$$= C^2d_i^{-2\alpha-1} - 2Cd_i^{-\alpha-1} + 1 \tag{20}$$

Therefore, the error of graph $\mathcal{G}$ is calculated as

$$\mathcal{E}_{\mathcal{G}} = \sum_i \varepsilon_i = \sum_{k=1}^{+\infty}P(k)\varepsilon(k), \tag{21}$$

where $P(k)$ is the probability of node degree equals to k, and $\varepsilon(k)$ is the corresponding node error. Here we assume the inherent graph model follows the Barabási–Albert model [20], where the probability of node degree equals to

$$P(k) = 2m^2 \cdot \frac{t}{\mathcal{N}_0 + t} \cdot \frac{1}{k^3}. \tag{22}$$

$\mathcal{N}_0$ is the initial scale of graphs, m is the newly-arrival number of nodes of each time t. Consequently, the error of graph $\mathcal{G}$ can be further deduced as:

$$\mathcal{E}_{\mathcal{G}} = \sum_{k=1}^{+\infty} 2m^2 \cdot \frac{t}{\mathcal{N}_0 + t} \frac{1}{k^3} (C^2 \cdot k^{-2\alpha-1} - 2C \cdot k^{-\alpha-1} + 1) \tag{23}$$

$$= 2m^2 \frac{t}{\mathcal{N}_0 + t} (C^2 \sum_{k=1}^{+\infty} k^{-2\alpha-4} - 2C \sum_{k=1}^{+\infty} k^{-2\alpha-4} + \sum_{k=1}^{+\infty} k^{-3}) \tag{24}$$

□

E Real-world Datasets

We use two citation datasets, a co-authorship network dataset and a series of social network datasets to evaluate our model's performance. We utilize inductive learning, wherein nodes and edges that emerge during testing remain unobserved during the training phase.

- **OGB-arXiv** [15]: The OGB-arXiv dataset is a citation network where each node represents an arXiv paper, and each edge signifies the citation relationship between these papers. Within this dataset, we conduct node classification tasks, encompassing a total of 40 distinct subject areas. Our experiment spans the years from 2007 to 2020. In its initial state in 2007, OGB-arXiv comprises 4,980 nodes and 6,103 edges. As the graph evolves over time, the citation network boasts 169,343 nodes and 1,166,243 edges. We commence by pretraining graph neural networks on the graph in 2007. Subsequently, we employ data from the years 2008 to 2010 to train our generalization estimation model SMART. Following this training, we predict the generalization performance of the pretrained graph neural network on graphs spanning the years 2011 to 2020.
- **DBLP** [16]: DBLP is also a citation network and this dataset use the conferences and journals as classes. In our experiment, DBLP starts from 1999 to 2015 with 6 classes. Throughout the evolution of DBLP, the number of nodes increase from 6,968 to 45,407, while the number of edges grow from 25,748 to 267,227. We pretrain the graph neural network on the graph in 1999 and train our model on the next three years. We employ the graph spanning from 2004 to 2015 to assess the performance of our model.
- **Pharmabio** [16]: Pharmabio is a co-authorship graph dataset, and each node represents a paper with normalized TF-IDF representations of the publication title as its feature. If two papers share common authors, an edge is established between the corresponding nodes. We conduct node classification tasks on this dataset, comprising a total of seven classes, with each class representing a journal category. The range of Pharmabio is 1985 to 2016. The pretrained graph neural network is based on the graph of the year 1985 with 620 nodes and 57,559 edges. Then we train our estimation model by using graph data from 1986 to 1988. We evaluate our model on consecutive 26 years starting form 1989. At the last year 2016, the graph has evolved to 2,820 nodes with 3,005,421 edges.
- **Facebook 100** [17]: Facebook 100 is a social network which models the friendship of users within five university. We perform binary node classification on this dataset, with the classes representing the gender of the users. Among these datasets, Amherst, Reed and Johns Hopkins are of smaller scale, while Penn and Cornell are larger in size. We sequentially evaluate our model's adaptability to datasets of different scales. All these datasets end in the year of 2010 with the number of nodes varying from 865 to 38,815 and edges from 31,896 to 2,498,498.

F Implementation Details

In this section, we present the implementation details of our model. We adopt the vanilla graph convolution network as the pre-trained graph model G, which is trained on the initial timestep of the

Table 3: Hyperparameter setting in our experiments

Datasets	OGB-arXiv	DBLP	Pharmabio	Facebook 100				
				Penn	Amherst	Reed	Johns Hopkins	Cornell
GNN Layer	3	3	2	2	2	1	2	2
GNN dimension	256	256	256	256	256	32	256	256
RNN Layer	1	1	1	1	1	1	1	1
RNN dimension	64	8	64	64	64	32	64	8
loss lambda	0.5	0.9	0.7	0.3	0.9	0.7	0.1	0.5
learning rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

graph with 10% labeling. During training, we only select the next 3 historical timesteps, where we randomly label 10% of the newly-arrived nodes. The remaining timesteps are reserved for testing, where we have no observations of the label. We run SMART and baselines 20 times with different random seeds. We use Adam optimizer for all the experiments, and the learning rate for all datasets are uniformly set to be 1e-3. In all experiments, the pre-trained graph neural networks are equipped with batch normalization and residual connections, with a dropout rate set to 0.1. Meanwhile, We employed the ReLU activation function. We set hyperparameter for each datasets and specify the details in Table 3.

To simulate real-world human annotation scenarios, we randomly labeled 10% of the samples during the training of the pre-trained graph neural network model. Prior to deployment, at each time step, we labeled 10% of the newly appearing nodes. After deployment, no additional labeling information was available for newly added nodes. For consistency, we use only the first three frames to obtain few labels for all real-world datasets, which is a relatively small sample size. Further enhancing the labeled data can yield additional improvements in temporal generalization estimation.

All the evaluated models are implemented on a server with two CPUs (Intel Xeon Platinum 8336C $\times$ 2) and four GPUs (NVIDIA GeForce RTX 4090 $\times$ 8).

G Additional Experiment Results

In this section, we present additional experiment results as follows.

Estimation on Different Test Time Period. In Table 4, we demonstrate the performance of SMART over time during the evolution of graphs in five social network datasets from Facebook 100. As the evolving pattern gradually deviates from the pre-trained model on the initial graph, generalization estimation becomes more challenging. Consequently, the error in linear estimation increases. However, our SMART method maintains overall stable prediction performance.

Table 4: Performance comparison on five social network datasets in Facebook 100. We divide the test time T_{test} into 3 periods and evaluate the estimation performance separately.

Facebook 100	$[0, T_{\text{test}}/3]$		$(T_{\text{test}}/3, 2T_{\text{test}}/3]$		$(2T_{\text{test}}/3, T_{\text{test}}]$	
	Linear	SMART	Linear	SMART	Linear	SMART
Penn	1.9428	0.0193±0.0041	2.0432	0.6127±0.0307	2.7219	2.2745±0.0553
Amherst	31.1095	1.4489±0.2450	49.2363	2.8280±0.9527	73.5709	4.3320±1.8799
Reed	55.6071	0.0453±0.0020	65.7536	0.0987±0.0078	73.6452	0.0318±0.0085
Johns Hopkins	8.1043	0.5893±0.0491	10.2035	0.8607±0.1661	11.5206	0.9061±0.2795
Cornell	4.5655	0.4663±0.0275	8.6622	1.0467±0.0817	12.3263	1.7311±0.1175

Proportional Ratio of Two Reconstruction Loss. We evaluate performance using different weight ratios $\lambda \in \{0, 1, 0.3, 0.5, 0.7, 0.9\}$, as shown in Figure 4. Our method is generally insensitive to the choice of λ, with $\lambda = 0.5$ being a balanced option in most cases. However, larger λ values can yield better results in specific datasets, such as DBLP and PharmaBio, especially when the node features are simple, like one-hot encoding or TF-IDF representations.

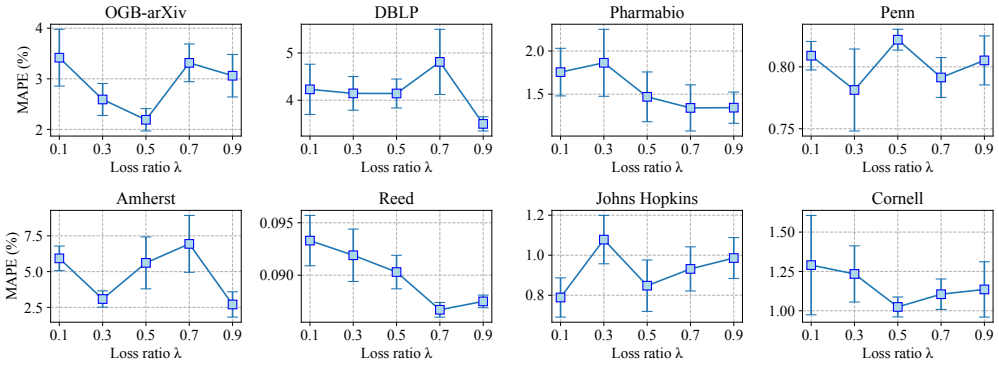


Figure 4: Hyperparameter Study on proportional weight ratio λ of SMART on all datasets.

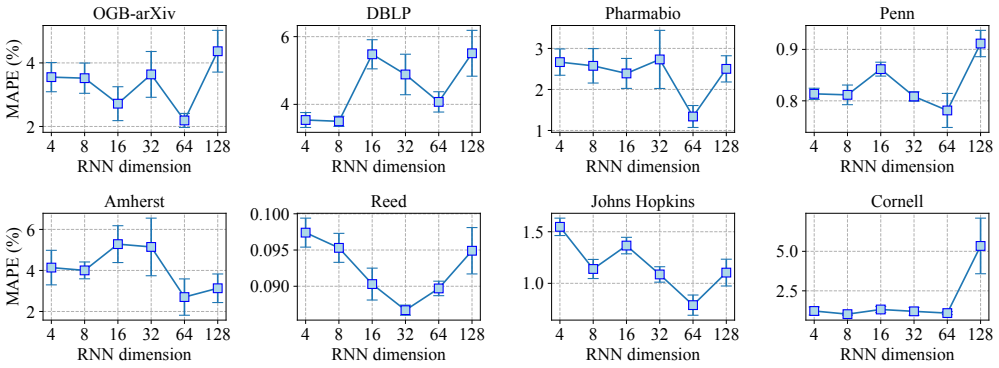


Figure 5: Hyperparameter Study on RNN dimension of SMART on all datasets.

Feature Dimension of RNN Input. We compared RNN feature dimensions ranging from $\{4, 8, 16, 32, 64, 128\}$, as shown in Figure 5. Performance remains stable across four datasets when the feature dimension is set between 4 and 64. However, a significant performance drop occurs on the Cornell dataset when the dimension is set to 128. Setting the RNN feature dimension too high is discouraged for two reasons: (1) As shown in Equation 1, RNN input represents compressed node information over time. To enhance historical information density and effectiveness, the input dimension should be reduced, facilitated by the reconstruction loss. (2) Given limited observation samples during training, reducing the RNN input dimension helps alleviate training pressure.