Test-Time Training for Graph Neural Networks

Yiqi Wang[1], Chaozhuo Li[2]✉, Jianan Zhao[3], and Rui Li[4]

[1] National University of Defense Technology
[2] Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications
[3] Université de Montréal
[4] Gaoling School of Artificial Intelligence, Renmin University of China
yiq@nudt.edu.cn,lichaozhuo@bupt.edu.cn,jianan.zhao@mila.quebec, lirui121200@ruc.edu.cn

Abstract. Graph Neural Networks (GNNs) have made tremendous progress in the graph classification task. However, a performance gap between the training set and the test set has often been noticed. To bridge such gap, in this work we introduce the first test-time training framework for GNNs to enhance the model generalization capacity for the graph classification task. In particular, we design a novel test-time training strategy with self-supervised learning to adjust the GNN model for each test graph sample. Experiments on the benchmark datasets have demonstrated the effectiveness of the proposed framework, especially when there are distribution shifts between training set and test set. We have also conducted exploratory studies and theoretical analysis to gain deeper understandings on the rationality of the design of the proposed graph test time training framework (GT3).

Keywords: Graph classification · Test-time training

1 Introduction

Many real-world data can be naturally represented as graphs, such as social networks [3] and molecules [24,19,7]. Graph neural networks (GNNs) [22,13,11], a successful generalization of deep neural networks (DNNs) over the graph domain, typically refine node representations via information aggregation among nodes and feature transformation that can be summarized as the graph representation via graph pooling operation. GNNs have been empirically and theoretically proven to be effective in graph representation learning, and have achieved revolutionary progress in various graph-related tasks, including node classification [9,11], graph classification [24,23,14] and link prediction [8]. Graph classification is one of the most important tasks on graphs, which aims to predict some properties of graphs. There are lots of real-world scenarios for graph classification, such as predicting molecular property to help drug discovery [2], forecasting the movie type based on the actor/actress connection graphs [6] and predicting the collaboration topic and research interests of a academic collaboration graph [9].

In the graph classification task, we typically train a graph neural network from the training graph samples and then utilize the trained model to make predictions for the test graph samples. This training strategy works effectively when the training samples and the test samples come from the same distribution. However, this assumption may not hold in many real scenarios. Often, there could exist gaps between the distributions of the training data and the test data, and the model performance will degrade significantly due to such gaps [18]. For example, as reported by the OGB paper [7] , there exist significant gaps between the performance of both the GCN [9] model and the GIN [23] model in the training set and the test set in predicting whether HIV virus replication is inhibited by a molecular. Besides, in graph domain, graph samples can be very diverse in terms of their graph structures. In D&D dataset [15], two graph samples from the same dataset can be very different from each other in terms of structural properties. This also brings in new challenges for the distribution shift problem in graphs. Although recognized by existing works [7], this problem has rarely been explored for GNNs.

Recently, test-time training [18,12], a newly-proposed paradigm, has been successfully proposed to address the distribution shift issue for images [18,1]. Specifically, it adapts the model trained on the training set via solving a self-supervised learning task, such as image rotation, specifically for each single image at the test time. This paradigm has the great potential to solve the graph distribution problem for GNNs, since it not only aims at improving the model generalization, but also can be designed to specifically adapt a GNN model for each single graph sample with unique structure information. However, it is challenging to design a test-time training framework for graph neural networks over graph data. First, what self-supervised learning task to design for test-time training in graph data? Graph data usually consists of not only node attribute information but also abundant structure information. Thus, it is important to blend these two kinds of graph information into the design of the self-supervised learning task. Second, how to mitigate the potential severe feature space distortion during the test time? The representation space can be severally tortured in some directions that are undesirable for the main task in the test-time training [12], since the model is adapted solely based on one sample and a self-supervised learning task. This problem can be much severer for graph data, as graph samples can vary greatly from each other in both the node attributes and the graph structures. Third, as the first work to design a test-time training framework for GNNs, it is desired to understand the rationality of test-time training over GNNs. In order to solve these challenges, we propose Graph Test-Time Training with Constraint (GT3), a test-time training framework with model adaptation constraints and carefully designed self-supervised learning tasks especially for graph neural networks. Our key contributions are summarized as follows:

– We propose to use a two-level self-supervised learning task in the test-time training framework for GNNs, consisting of both the local (node-node level) contrast learning and global (node-graph level) contrast learning, which can fully exploit the attribute and structure information of each graph.

– We have designed an effective adaptation constraint in the test-time training process for GNNs, which can help prevent the excessive feature space distortion and increase the framework robustness and performance.
– We have conducted exploratory studies and theoretical analysis to understand the rationality of the design of the proposed framework.

2 The Proposed Test-time Training Framework for GNNs

In this section, we introduce the proposed framework of Graph Test-Time Training with Constraint (GT3) and detail its key components. We first discuss the overall process, then detail the two-level SSL task and finally describe the adaptation constraint to mitigate the potential issue of feature distortion in GT3, which is detailed at Section 6.1 in Appendix. In addition, we also explore the rationality of test-time training for GNNs from a theoretical perspective in Section 6.6 in Appendix.

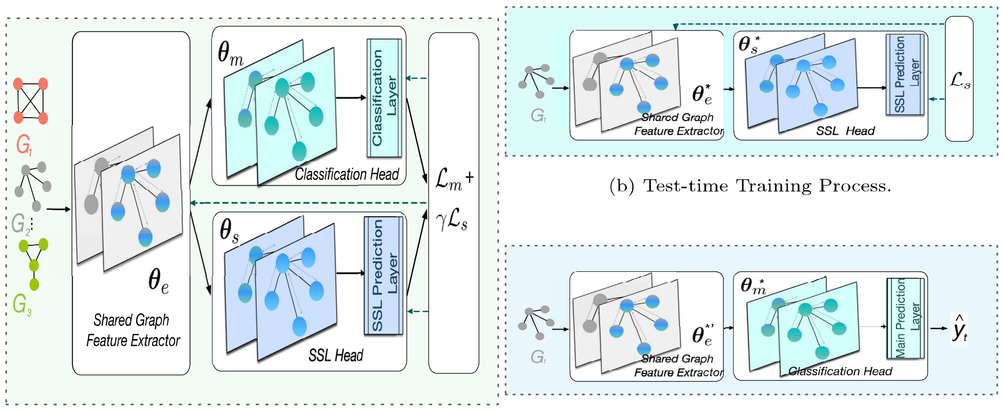
2.1 An Overview

Suppose that the graph neural network for graph classification task consists of L GNN layers, and the learnable parameters in the l-th layer are denoted as $\boldsymbol{\theta}_l$. The overall model parameters can be denoted as $\boldsymbol{\theta}_{main} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_L, \boldsymbol{\delta}_m)$, where $\boldsymbol{\delta}_m$ is the parameters in the prediction layer. In this work, we mainly focus on the graph classification task, which is named as *the main task* in the test-time training framework. Given the training graph set $\mathcal{G} = (G_1, G_2, \dots, G_n)$ and its corresponding graph label set $\mathcal{Y} = (y_1, y_2, \dots, y_n)$. The main task is to solve the following empirical risk minimization problem:

$$\min_{\boldsymbol{\theta}_{main}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_m(\mathbf{A}_i, \mathbf{X}_i, y_i; \boldsymbol{\theta}_{main}), G_i \in \mathcal{G}, \tag{1}$$

where $\mathcal{L}_m(\cdot)$ denotes the loss function for the main task, $\mathbf{A}_i$ is the graph adjacency matrix and $\mathbf{X}_i$ represents the node attributes . In addition to the main task, self-supervised learning (SSL) tasks also play a vital role in the test-time training framework. In this work, we carefully design a two-level graph contrastive learning task as the SSL task (details in Section 2.2). For simplicity, the loss for any self-supervised learning task is denoted as $\mathcal{L}_s(\cdot)$. With the main task and the SSL task, an overall of the proposed framework GT3 is shown in Figure 1. It consists of three key processes: (1) the training process (Figure 1b); (2) the test-time training process (Figure 1b) and (3) the inference process (Figure 1c).

The Training Process. Given the training graphs $\mathcal{G}$, the training process is to train the model parameters with the main task and the SSL task. The key challenge is how to integrate the main task and the SSL task. To tackle this challenge, we conduct an empirical study where we train two GNN models separately by the main task and the SSL task, and we found that these two GNN models will extract similar features by their first a few layers. More details can be found



(a) The Training Process.

(b) Test-time Training Process.

(c) Inference Process.

Fig. 1: An Overview of the Proposed Framework GT3.

at Section 6.3 in Appendix. This empirical finding paves us a way to integrate the main task and the SSL task: we allow them to share the parameters in the first a few layers as shown in Figure 1b. We denote the shared model parameters as $\boldsymbol{\theta}_e = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K)$, where $K \in \{1, 2, \ldots, L\}$. The shared component is named as *shared graph feature extractor* in this work. The model parameters that are specially for the main task are denoted as $\boldsymbol{\theta}_m = (\boldsymbol{\theta}_{K+1}, \ldots, \boldsymbol{\theta}_L, \delta_m)$, described as *the graph classification head*. Correspondingly, the SSL task has its own *self-supervised learning head*, denoted as $\boldsymbol{\theta}_s = (\boldsymbol{\theta}'_{K+1}, \ldots, \boldsymbol{\theta}'_L, \boldsymbol{\delta}_s)$, where $\boldsymbol{\delta}_s$ is the prediction layer for the SSL task. To summarize, the model parameters for the main task can be denoted as $\boldsymbol{\theta}_{main} = (\boldsymbol{\theta}_e, \boldsymbol{\theta}_m)$, the model parameters for the SSL task can be represented as $\boldsymbol{\theta}_{self} = (\boldsymbol{\theta}_e, \boldsymbol{\theta}_s)$, and these for the whole framework can be denoted as $\boldsymbol{\theta}_{overall} = (\boldsymbol{\theta}_e, \boldsymbol{\theta}_m, \boldsymbol{\theta}_s)$. In the training process, we train the whole framework by minimizing a weighted combination loss of $\mathcal{L}_m(\cdot)$ and $\mathcal{L}_s(\cdot)$ as follows:

$$\min_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_m, \boldsymbol{\theta}_s} \frac{1}{n} \sum_{i=1}^{N} \mathcal{L}_m(\mathbf{A}_i, \mathbf{X}_i, y_i; \boldsymbol{\theta}_e, \boldsymbol{\theta}_m) + \gamma \mathcal{L}_s(\mathbf{A}_i, \mathbf{X}_i, y_i; \boldsymbol{\theta}_e, \boldsymbol{\theta}_s), G_i \in \mathcal{G}, \tag{2}$$

where γ is a hyperparameter that balances the main task and the self-supervised task in the training process.

The Test-time Training Process. Given a test graph G_t, the test-time training process will fine-tune the model via the SSL task as illustrated in Figure 1b. Specifically, given a test graph sample $G_t = (\mathbf{A}_t, \mathbf{X}_t)$, the *shared graph feature extractor* and the *self-supervised learning head* are fine-tuned by minimizing the SSL task loss as follows:

$$\min_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_s} \mathcal{L}_s(\mathbf{A}_t, \mathbf{X}_t; \boldsymbol{\theta}_e, \boldsymbol{\theta}_s), \tag{3}$$

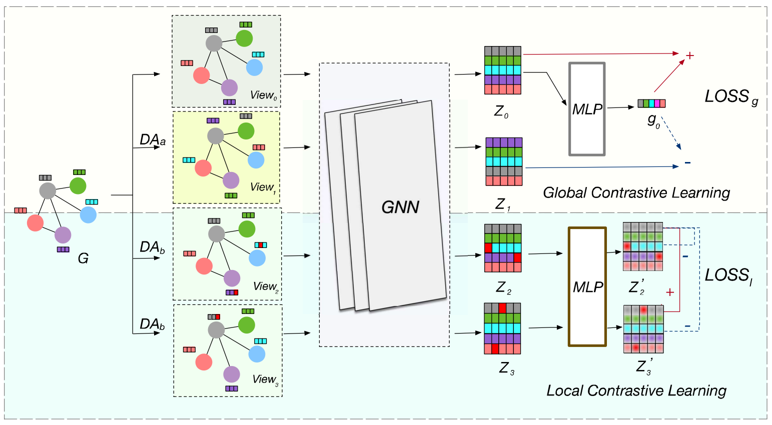


Fig. 2: The Self-supervised Learning Task for GT3.

Suppose that $\boldsymbol{\theta}^*_{overall} = (\boldsymbol{\theta}^*_e, \boldsymbol{\theta}^*_m, \boldsymbol{\theta}^*_s)$ is the optimal parameters from minimizing the overall loss of Eq 2 from the training process. In the test process, given a specific test graph sample G_t, the proposed framework further updates $\boldsymbol{\theta}^*_e$ and $\boldsymbol{\theta}^*_s$ to $\boldsymbol{\theta}^{*'}_e$ and $\boldsymbol{\theta}^{*'}_s$ by minimizing the SSL loss of Eq 3 over the test graph sample G_t.

The Inference Process. The inference process is to predict the label of the test graph G_t based on the finetuned model by the test-time training as demonstrated in Figure 1c. In particular, the label of G_t is predicted via the GNN model $f(\cdot; \boldsymbol{\theta}^{*'}_e, \boldsymbol{\theta}^*_m)$ as:

$$\hat{y}_t = f(\mathbf{X}_t, \mathbf{A}_t; \boldsymbol{\theta}^{*'}_e, \boldsymbol{\theta}^*_m). \tag{4}$$

2.2 Self-supervised Learning Task for GT3

An appropriate and informative SSL task is the key to the success of test-time training [12]. However, it is challenging to design an appropriate SSL task for test-time training for GNNs. First, graph data is intrinsically different from images, some common properties such as rotation invariance do not exist in graph data, thus most SSL tasks commonly adopted by existing test-time training are invalid in graph data. Second, most existing SSL tasks for graph classification are intra-graph contrastive learning based on the difference among diverse graph samples [5]. These are not applicable for the test-time training where we aim at specifically adapting the model for each single graph during the test time. Motivated by the success of constrastive learning in graph domains, we propose to use a two-level SSL task from both local and global perspectives for GT3, which fully exploits the graph information from both node-node level and node-graph level. The proposed self-supervised learning task is based on graph characteristic, instead of the properties in Euclidean data. In addition, the proposed SSL task is not built up based on the differences among distinct graphs, thus it can be naturally applied for a single graph. We empirically validate that the global (node-graph level) contrastive learning task and the local (node-node level) contrastive learning task in the proposed two-level SSL task are necessary for GT3 since they are complementary to each other cross datasets. More details about this empirical study can be found at Section 6.3 in Appendix. Next, we will detail the global contrastive learning task and the local contrastive learning task.

Global Contrastive Learning The global contrastive learning aims at helping node representation capture global information of the whole graph. Overall, the global contrastive learning is based on maximizing the mutual information between the local node representation and the global graph representation. Specifically, as shown in Figure 2, given an input graph sample G, different views can be generated via various types of data augmentations.

Following [20], the positive samples in the global contrastive learning consist of node-graph representation pairs, where both the node representation and graph representation come from the raw view $View_0$. The negative samples consist of node-graph representation pairs where node representations come from $View_1$ (the augmented view) and graph representation comes from $View_0$. A discriminator $\mathcal{D}$ is employed to compute the probability score for each pair, and the score should be higher for the positive pair and lower for the negative pair. In our work, we set $\mathcal{D}(\mathbf{Z}_{si}, \mathbf{g_0}) = \mathbf{Z}_{si} * \mathbf{g}_0$, where $\mathbf{Z_s}_i$ denotes the node representation of node i from $View_s$ and $\mathbf{g_0}$ representes the graph representation from $View_0$.

The objective function for the global contrastive learning is summarized as follows:

$$\mathcal{L}_g = -\frac{1}{2N}(\sum_{i=1}^{N}(log\mathcal{D}(\mathbf{Z}_{0i}, \mathbf{g}_0) + log(1 - \mathcal{D}(\mathbf{Z}_{1i}, \mathbf{g}_0))), \qquad (5)$$

where N denotes the number of nodes in the input graph.

Local Contrastive Learning To fully exploit the structure information of a graph, in addition to the invariant graph representation from a global level, it is also important to learn invariant node representations from a local level. To achieve this, we propose local contrastive learning, which is based on distinguishing different nodes from different augmented views of a graph. As shown in Figure 2, given an input graph G, we can generate two views of the graph via data augmentation. Since the local contrastive learning is to distinguish whether two nodes from different views are the same node of the input graph, the adopted data augmentation should not make large changes on the input graph. To meet this requirement, we adopt two adaptive graph data mechanisms – adaptive edge dropping and adaptive node attribute masking. Specifically, the edges are dropped based on the edge importance in the graph. The more important an edge is, the less likely it would be dropped. Likewise, the node attributes are masked based on the importance of each dimension of node attributes. The more important the attribute dimension is, the less likely it would be masked out.

After the described data augmentation, two views of an input graph are generated and serve as the input of the shared GNN model. The outputs are two node representation matrices $\mathbf{Z}_2$ and $\mathbf{Z}_3$ corresponding for two views. Note that the basic goal of local contrastive learning is to distinguish whether two nodes from augmented views are the same node in the input graph. Therefore, $(\mathbf{Z}_{2i}, \mathbf{Z}_{3i})$, $(i \in \{1, \ldots, N\})$ denotes a positive pair, where N is the number of nodes. $(\mathbf{Z}_{2i}, \mathbf{Z}_{3j})$ and $(\mathbf{Z}_{2i}, \mathbf{Z}_{2j})$ $(i, j \in \{1, \ldots, N\}$ and $i \neq j)$ denote an intra-view negative pair and an inter-view negative pair, respectively. Inspired by InfoNCE [17,25], the objective for a positive node pair $(\mathbf{Z}_{2i}, \mathbf{Z}_{3i})$ is defined as $\ell_c(\mathbf{Z}_{2i}, \mathbf{Z}_{3i}) = log\frac{h(\mathbf{Z}_{2i},\mathbf{Z}_{3i})}{h(\mathbf{Z}_{2i},\mathbf{Z}_{3i})+\sum_{j\neq i} h(\mathbf{Z}_{2i},\mathbf{Z}_{3j})+\sum_{j\neq i} h(\mathbf{Z}_{2i},\mathbf{Z}_{2j})}$, where $h(\mathbf{Z}_{2i}, \mathbf{Z}_{3j}) = e^{cos\left(g(\mathbf{Z}_{2i}), g(\mathbf{Z}_{3j})\right)/\tau}$, $cos()$ denotes the cosine similarity function, τ is a temperature parameter and $g()$ is a two-layer perceptron (MLP) to further enhance the model expression power. The node representations refined by this MLP are denoted as $\mathbf{Z}_2^{'}$ and $\mathbf{Z}_3^{'}$, respectively.

Apart from the contrastive objective described above, a decorrelation regularizer has also been added into the overall objective function of the local contrastive learning, in order to encourage different representation dimensions to capture distinct information. The decorrelation regularizer is applied over the refined node representations $\mathbf{Z}_2^{'}$ and $\mathbf{Z}_3^{'}$ as $\ell_d(\mathbf{Z}_2^{'}) = \left\| \mathbf{Z}_2^{'T} \mathbf{Z}_2^{'} - I \right\|_F^2$.

The overall objective loss function of the local contrastive learning is:

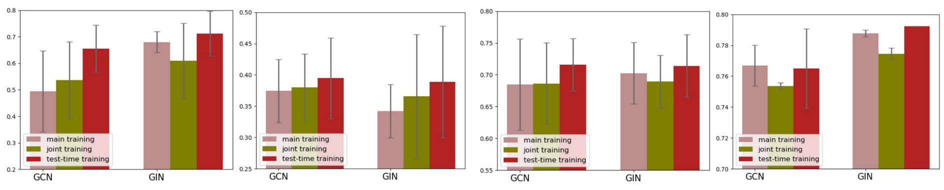
$$\mathcal{L}_l = -\frac{1}{2N} \sum_{i=1}^{N} (\ell_c(\mathbf{Z}_{2i}, \mathbf{Z}_{3i}) + \ell_c(\mathbf{Z}_{3i}, \mathbf{Z}_{2i})) - \frac{\beta}{2}(\ell_d(\mathbf{Z}_2^{'}) + \ell_d(\mathbf{Z}_3^{'})), \tag{6}$$

where β is the balance parameter.

3 Experiment

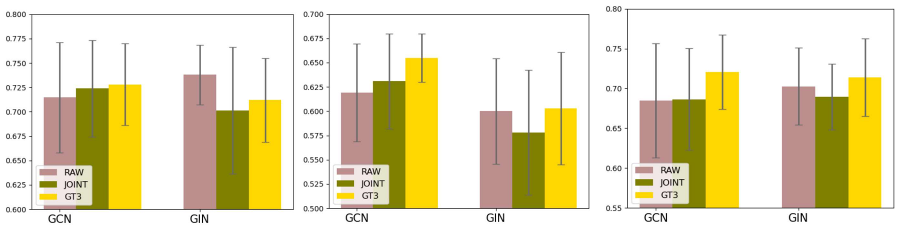
In this section, we validate the effectiveness and explore the rationality of the proposed GT3 through empirical studies on four widely-used graph datasets, detailed experimental settings can be found at Section 6.2 in Appendix. In addition, we also conduct a layer-wise feature space exploration of GNN models trained for different tasks to further validate the rationality of the design of GT3. Furthermore, we perform ablation study and hyper-parameter sensitivity exploration. The corresponding results and analysis are at Section 6.3, 6.4 and 6.5 in Appendix.

We compare the graph classification performance of GCN and GIN in three mechanisms: (1) *RAW* model is trained only by the main task, (2) *JOINT* model is jointly trained by the main task and the proposed two-level SSL task, and (3) *GT3* model is learned by test-time training. The comparison results are shown in Figure 3. Note that the performance of *RAW* in ogbg-molhiv is copied from [7]. From the figure, we can observe that the joint training of the proposed two-level SSL task and the main task can slightly improve the model performance in some cases. However, the proposed *GT3* is able to consistently perform better than both *RAW* and *JOINT*, which demonstrates the effectiveness of the proposed *GT3*. To further demonstrate the performance improvement from *GT3*, we summarize the relative performance improvement of *GT3* over *RAW*. The relative performance improvements of GCN over DD dataset can reach 43.3%. In addition to the performance comparison on the *OOD* data split, we have also conducted a performance comparison on a random data split, where we randomly split the dataset into 80%/10%/10% for the training/validation/test sets, respectively. The results are shown in Figure 4. From it, we can observe that GT3 can also improve the classification performance in most cases, which demonstrates the effectiveness of the proposed GT3 in multiple scenarios.



(a) DD (b) ENZYMES (c) PROTEINS (d) ogbg-molhiv

Fig. 3: Performance Comparison of GCN and GIN on Four Datasets based on *OOD* Data Split. Note that the performance metric for ogbg-molhiv is **ROC-AUC**(%) and that for others is **Accuracy**(%).



(a) DD (b) ENZYMES (c) PROTEINS

Fig. 4: Performance Comparison of GCN and GIN on Three Datasets on the Random Data Split. Note that the performance metric for ogbg-molhiv is **ROC-AUC**(%) and that for others is **Accuracy**(%).

4 Related Work

Test-time training with self-supervision is proposed in [18], which aims at improving the model generalization under distribution shift via solving a self-supervised task for test samples. This framework has empirically demonstrated its effectiveness in bridging the performance gap between training set and test set in image domain [18,1]. There are also works combining the test-time training and meta-learning learning [1], and attempting to apply test-time training in reinforcement learning [4]. Apart from these applied researches, TTT++ tries to explore when test-time training thrives from a theoretical perspective [12]. EATA [16] introduces a strategy to select non-redundant and reliable samples and proposes a Fisher regularizer to avoid the forgetting problem.

5 Conclusion

In this work, we design a test-time training framework for GNNs (GT3) for graph classification task, with the aim to bridge the performance gap between the training set and the test set when there is a data distribution shift. As the first work to combine test-time training and GNNs, we empirically explore the rationality of the test-time training framework over GNNs via a layer-wise representation comparison of GNNs for different tasks. In addition, we validate that the GNN performance on graph classification can be benefited by test-time training from a theoretical perspective. Furthermore, we demonstrate the effectiveness of the proposed GT3 on extensive experiments and understand the impact of the model components via ablation study.

References

1. Bartler, A., Bühler, A., Wiewel, F., Döbler, M., Yang, B.: Mt3: Meta test-time training for self-supervised test-time adaption. arXiv preprint arXiv:2103.16201 (2021)
2. Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: NeurIPS (2015)
3. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, Y.E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019 (2019)
4. Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A.A., Pinto, L., Wang, X.: Self-supervised policy adaptation during deployment. arXiv preprint arXiv:2007.04309 (2020)
5. Hassani, K., Ahmadi, A.H.K.: Contrastive multi-view representation learning on graphs. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Proceedings of Machine Learning Research (2020)
6. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. pp. 639–648 (2020)
7. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. In: NeurIPS (2020)

8. Kipf, T.N., Welling, M.: Variational graph auto-encoders. ArXiv preprint (2016)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
10. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning. pp. 3519–3529. PMLR (2019)
11. Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., Yan, M., Tang, J.: Elastic graph neural networks. In: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. Proceedings of Machine Learning Research (2021)
12. Liu, Y., Kothari, P., van Delft, B., Bellot-Gurlet, B., Mordan, T., Alahi, A.: Ttt++: When does self-supervised test-time training fail or thrive? Advances in Neural Information Processing Systems **34** (2021)
13. Ma, Y., Tang, J.: Deep learning on graphs. Cambridge University Press (2021)
14. Ma, Y., Wang, S., Aggarwal, C.C., Tang, J.: Graph convolutional networks with eigenpooling. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 723–731 (2019)
15. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: A collection of benchmark datasets for learning with graphs. arXiv preprint arXiv:2007.08663 (2020)
16. Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., Tan, M.: Efficient test-time model adaptation without forgetting. In: International conference on machine learning. pp. 16888–16905. PMLR (2022)
17. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. ArXiv preprint (2018)
18. Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A.A., Hardt, M.: Test-time training for out-of-distribution generalization (2019)
19. Tang, W., Wen, H., Liu, R., Ding, J., Jin, W., Xie, Y., Liu, H., Tang, J.: Single-cell multimodal prediction via transformers. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 2422–2431 (2023)
20. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. ICLR (Poster) **2**(3), 4 (2019)
21. Wu, F., Jr., A.H.S., Zhang, T., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. In: ICML. Proceedings of Machine Learning Research (2019)
22. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. ArXiv preprint (2019)
23. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=ryGs6iA5Km`
24. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: NeurIPS (2018)
25. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021. pp. 2069–2080 (2021)