Overcoming Catastrophic Forgetting in Graph Neural Networks

Huihui Liu, Yiding Yang, Xinchao Wang*

Stevens Institute of Technology
{hliu79, yyang99, xinchao.wang}@stevens.edu

Abstract

Catastrophic forgetting refers to the tendency that a neural network "forgets" the previous learned knowledge upon learning new tasks. Prior methods have been focused on overcoming this problem on convolutional neural networks (CNNs), where the input samples like images lie in a grid domain, but have largely overlooked graph neural networks (GNNs) that handle non-grid data. In this paper, we propose a novel scheme dedicated to overcoming catastrophic forgetting problem and hence strengthen continual learning in GNNs. At the heart of our approach is a generic module, termed as topology-aware weight preserving (TWP), applicable to arbitrary form of GNNs in a plug-and-play fashion. Unlike the main stream of CNN-based continual learning methods that rely on solely slowing down the updates of parameters important to the downstream task, TWP explicitly explores the local structures of the input graph, and attempts to stabilize the parameters playing pivotal roles in the topological aggregation. We evaluate TWP on different GNN backbones over several datasets, and demonstrate that it yields performances superior to the state of the art. Code is publicly available at https://github.com/hhliu79/TWP.

Introduction

Deep neural networks have demonstrated unprecedentedly gratifying results in many artificial intelligence tasks. Despite the encouraging progress, in the *continual learning* scenario where a model is expected to learn from a sequence of tasks and data, deep models are prone to the *catastrophic forgetting* problem (McCloskey and Cohen 1989; Ratcliff 1990; McClelland, McNaughton, and O'Reilly 1995; French 1999).

Many recent endeavours have been made towards alleviating catastrophic forgetting or preserving the performance on the older task upon learning newer ones. Earlier approaches rely on a rehearsal scheme, in which a part of samples from previous tasks are stored and replayed in learning the new task (Lopez-Paz and Ranzato 2017; Rebuffi et al. 2017). Regularization-based methods, as another popular line of work, precludes the key parameters of the previous tasks from undergoing dramatic changes (Kirkpatrick et al. 2017; Li and Hoiem 2017; Aljundi et al. 2018). Parameter-isolation methods, on the other hand, focus on assigning different subsets of parameters for different tasks and reusing knowledge from previous tasks (Rusu et al. 2016; Fernando et al. 2017; Yoon et al. 2017).

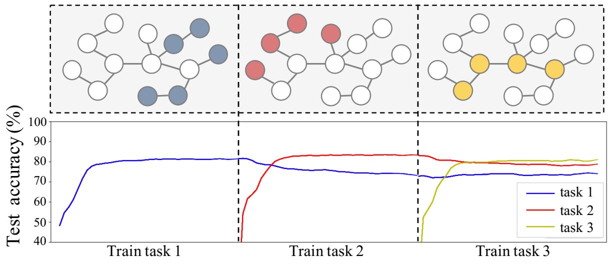

Figure 1: Illustration of catastrophic forgetting on graph attention networks (GATs) models on the Corafull dataset based on a CNN-based method, EWC (Kirkpatrick et al. 2017). Top: Blue, red, and yellow nodes are used for task 1, 2, and 3, respectively. Bottom: The test performance of each task when learning new tasks gradually.

Existing deep-learning-based continual learning methods, in spite of their impressive results, have been focusing on convolutional neural networks (CNNs) with the grid data like images as input. However, many real-world data lie in the non-grid domain and take the form of graphs. Examples include but are not limited to such as citation networks, social networks (Newman and Girvan 2004), and biological reaction networks (Pavlopoulos et al. 2011). To handle data in the irregular domain, various graph neural networks (GNNs) have been proposed (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017) to explicitly account for the topological interactions between nodes in the graph.

Yet unfortunately, GNNs suffer from catastrophic forgetting as well. As demonstrated in Figure 1, the state-of-the-art graph attention networks (GATs), when applied to a sequence of three node-classification tasks, significantly deteriorate as new tasks are learned. A straightforward solution towards handling this issue is to directly apply continual learning methods designed for CNNs on the graph data, by treating each node as a single sample like an image for CNNs. However, the CNN-based methods merely focus on individual nodes of the graph but neglect the topological structure and the interconnections between nodes, which

*Corresponding Author.

play pivotal roles in propagating and aggregating information in GNNs. In the case of Figure 1, this approach leads to a performance drop of 7.48% on the accuracy of task 1 upon learning task 3. In this paper, we propose a novel approach dedicated to overcoming catastrophic forgetting problem in GNNs. Specifically, we introduce a generic module, dubbed as topology-aware weight preserving (TWP). As opposed to conventional CNN-based methods that strive to retain only the performances of the old task by decelerating changes of the crucial parameters, TWP also explicitly looks into topological aggregation, and endeavors to preserve the attention and aggregation strategy of GNNs on the old tasks. In other words, TWP expressly accounts for not only the node-level learning in terms of parameter updates, but also the propagation between nodes for the continual learning process.

The workflow of the proposed GNN-based continual learning approach is shown in Figure 2. Given an input graph and embedded feature of nodes, the TWP module estimates an importance score for each parameter of the network based on its contribution to both the task-related performance and the topological structure. This is achieved by computing the gradients of the task-wise objective and the topological-preserving one with respect to each parameter, and treating such gradient as an index for the parameter importance. Upon learning a new task, we penalize the changes to the significant parameters with respect to all the old tasks, and hence enables us to "remember" the knowledge learned from the previous tasks. When employing our proposed method to solve the problem in Figure 1, the test accuracy of task 1 dropped by only 3.77%, which significantly enhances over the CNN-based method and showcases the power of utilizing the topological information of graphs. Since TWP does not impose assumptions on the form of a GNN model, it can be readily applicable to arbitrary GNN architectures.

We demonstrate the effectiveness of the proposed method on three GNN architectures, namely graph attention networks (GATs) (Veličković et al. 2017), graph convolutional networks (GCNs) (Kipf and Welling 2016), and graph isomorphism networks (GINs) (Xu et al. 2018) over four node classification datasets including Corafull, Amazon Computers, PPI, and Reddit, and one graph classification dataset, Tox21. Results show that our method consistently achieves the best performance among all the compared methods.

Our contribution is therefore introducing a novel continual learning scheme tailored for GNNs, dedicated to overcoming catastrophic forgetting. The proposed topology-aware weight preserving (TWP) module explicitly looks into the topological aggregation mechanism of the learned tasks to strength continual learning. Experiments on three GNN backbones over five datasets demonstrate that TWP consistently yields gratifying performances.

Related work

Graph Neural Networks. A pioneering graph-based convolutional network approach was proposed in (Bruna et al. 2013), which generalized CNNs to signals defined on more general domains, and the work (Defferrard, Bresson, and Vandergheynst 2016) improved it by using Chebyshev polynomials. (Kipf and Welling 2016) showed that GNNs can be built by stacking layers of first-order Chebyshev polynomial filters. (Hamilton, Ying, and Leskovec 2017), on the other hand, proposed a framework based on sampling and aggregation instead of using all nodes, while (Veličković et al. 2017) introduced an attention mechanism to GNNs by specifying different weights to different neighboring nodes. Many other GNN models (Yang et al. 2019, 2020a; You, Ying, and Leskovec 2019) have been explored recently, demonstrating promising performances on graph applications (Qiu et al. 2020; Wang et al. 2014)

Continual Learning. This line of work has focused on the memory modules (Lopez-Paz and Ranzato 2017; Rebuffi et al. 2017; Chaudhry et al. 2018), and thus falls within the domain of model reuse (Yang et al. 2020b; Ye et al. 2019; Shen et al. 2019). However, the computation and memory costs increase rapidly as the number of tasks increases, which facilitated the emergence of pseudo-rehearsal methods (Robins 1995) and generative network based methods (Shin et al. 2017; Nguyen et al. 2017). Recently, several methods have been proposed based on meta-learning algorithms (Riemer et al. 2019; Javed and White 2019) and online sample selection (Aljundi et al. 2019). Another popular strategy for overcoming catastrophic forgetting focuses on preserving the parameters inferred in one task while training on another. One milestone is Elastic Weight Consolidation (EWC) (Kirkpatrick et al. 2017) which slows down learning on important parameters for previous task. Subsequently, many other regularization based methods have been proposed (Zenke, Poole, and Ganguli 2017; Lee et al. 2017; Li and Hoiem 2017; Aljundi et al. 2018; Mallya and Lazebnik 2018) to help remember the old knowledge by penalizing changes to important parameters for previous tasks. The last major family seeks to prevent forgetting of old tasks by assigning different parameter subsets for different tasks and reusing as much previous knowledge as possible (French 1994; Chen, Goodfellow, and Shlens 2015; Rusu et al. 2016; Schwarz et al. 2018; Serrà et al. 2018). Our method can be cast as a regularization-based method, as it attempts to preserve parameters inferred in previous tasks to overcome catastrophic forgetting. Unlike previous methods, however, we explicitly integrate topological information of graphs into continual learning.

Preliminaries

Before introducing the proposed approach, we briefly review graph neural networks (GNNs) and formulate the problem of continual learning on GNNs.

Graph Neural Networks. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with N nodes, specified as a set of node feature $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ and an adjacency matrix $\boldsymbol{A}$ that represents the adjacency relations among nodes. The hidden representation of node v_i at the l-th layer, denoted as $\boldsymbol{h}_i^{(l)}$, is computed by:

$$\boldsymbol{h}_i^{(l)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \mathcal{A}_{ij} \boldsymbol{h}_j^{(l-1)} W^{(l)}\Big), \tag{1}$$

where $\mathcal{N}(i)$ denotes the neighbors of node v_i, $\sigma(\cdot)$ is an activation function, and $W^{(l)}$ is the transformation matrix of the

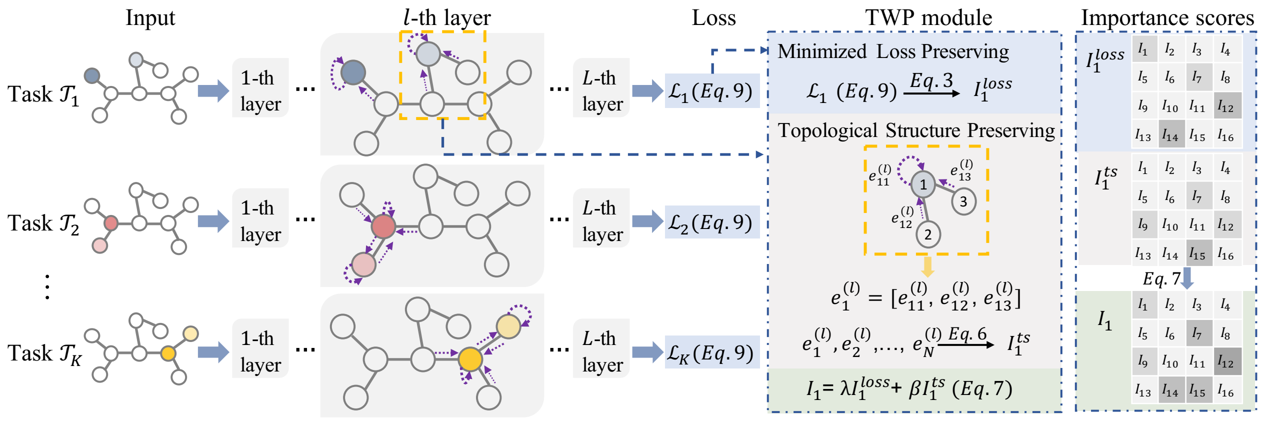
Input
l-th layer
Loss
TWP module
Importance scores
Task $\mathcal{T}_1$
Task $\mathcal{T}_2$
Task $\mathcal{T}_K$
1-th layer
L-th layer
$\mathcal{L}_1(Eq.9)$
$\mathcal{L}_2(Eq.9)$
$\mathcal{L}_K(Eq.9)$
Minimized Loss Preserving
$\mathcal{L}_1\ (Eq.9) \xrightarrow{Eq.3} I_1^{loss}$
Topological Structure Preserving
$e_{11}^{(l)}$
$e_{12}^{(l)}$
$e_{13}^{(l)}$
$e_1^{(l)} = [e_{11}^{(l)}, e_{12}^{(l)}, e_{13}^{(l)}]$
$e_1^{(l)}, e_2^{(l)}, ..., e_N^{(l)} \xrightarrow{Eq.6} I_1^{ts}$
$I_1 = \lambda I_1^{loss} + \beta I_1^{ts}\ (Eq.7)$
I_1^{loss}
I_1^{ts}
Eq. 7
I_1
I_1 I_2 I_3 I_4
I_5 I_6 I_7 I_8
I_9 I_{10} I_{11} I_{12}
I_{13} I_{14} I_{15} I_{16}

Figure 2: Overview of the proposed method. For simplicity, we assume that there are only two classes in each task, and take task $\mathcal{T}_1$ as an example to illustrate the TWP module. I_1 contains the importance scores of all network parameters for task $\mathcal{T}_1$, where the darker gray means higher importance score. Best viewed in color.

l-th layer. $\boldsymbol{h}_i^{(0)}$ represents the input feature of node v_i. $\mathcal{A}$ is a matrix that defines the aggregation strategy from neighbors, which is one of the cores of GNNs.

The graph convolutional networks (GCNs) (Kipf and Welling 2016) computes $\mathcal{A}$ based on a first-order approximate of the spectral of graph, which is fixed given $\boldsymbol{A}$. For attention-based GNNs, like graph attention networks (GATs) (Veličković et al. 2017), $\mathcal{A}$ is computed based on the pair-wise attention, which is defined as

$$e_{ij}^{(l)} = S_{j\in\mathcal{N}(i)} a(\boldsymbol{h}_i^{(l-1)} W^{(l)}, \boldsymbol{h}_j^{(l-1)} W^{(l)}), \tag{2}$$

where a is a neural network, S represents softmax normalization. Authors in (Xu et al. 2018) further invest the expression power of GNNs and proposes graph isomorphism networks (GINs).

Problem Formulation. In a learning sequence, the model receives a sequence of disjoint tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$ which will be learned sequentially. Each task $\mathcal{T}_k$ contains a training node set $\mathcal{V}_k^{tr}$ and a testing node set $\mathcal{V}_k^{te}$, with corresponding feature sets X_k^{tr} and X_k^{te}. Each node $v_i \in \{\mathcal{V}_k^{tr} \cup \mathcal{V}_k^{te}\}$ corresponds a category label $y^l \in \mathcal{Y}_k$ where $\mathcal{Y}_k = \{y^1, y^2, ..., y^{c_k}\}$ is the label set and c_k is the number of classes in task $\mathcal{T}_k$. In the continual learning settings, different tasks correspond to different splits of a dataset without overlap in category labels. Once the learning of a task is completed, the data related to this task is no longer available. In this paper, we aim to learn a shared GNN model f_W parameterized by $W = \{\boldsymbol{w}_m\}$ over a sequence of graph related tasks $\mathcal{T}$, such that this model can not only perform well on the new task but also remember old tasks.

Method

In this section, we provide the details of the topology-aware weight preserving (TWP) module and how the module boosts the continual learning performance of GNNs. Moreover, we impose sparse regularization to balance the plasticity and stability of GNNs. We also provide the scheme to extend the proposed method to arbitrary GNNs.

Topology-aware Weight Preserving

The TWP module captures the topology information of graphs and finds the crucial parameters that are both important to the task-related objective and the topology-related one. It contains two sub-modules: minimized loss preserving and topological structure preserving.

Minimized Loss Preserving. After training task $\mathcal{T}_k$, the model has learned a set of optimal parameters W_k^* that minimize the loss of this task. However, not all network parameters contribute equally. We thus try to find the crucial parameters for preserving the minimized loss.

Given the training set $\mathcal{D}_k^{tr}$ for task $\mathcal{T}_k$, we denote the loss as $\mathcal{L}(X_k^{tr}; W)$, where X_k^{tr} contains the feature of nodes in $\mathcal{D}_k^{tr}$ and $W = \{\boldsymbol{w}_m\}$ contains all network parameters. Like (Zenke, Poole, and Ganguli 2017), the change in loss for an infinitesimal parameter perturbation $\Delta W = \{\Delta\boldsymbol{w}_m\}$ can be approximated by

$$\mathcal{L}(X_k^{tr}; W + \Delta W) - \mathcal{L}(X_k^{tr}; W) \approx \sum_m f_m(X_k^{tr})\Delta\boldsymbol{w}_m, \tag{3}$$

where $\Delta\boldsymbol{w}_m$ is the infinitesimal change in parameter $\boldsymbol{w}_m$. $f_m(X_k^{tr}) = \frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_m}$ is the gradient of the loss with respect to the parameter $\boldsymbol{w}_m$, and it can be used to approximate the contribution of the parameter $\boldsymbol{w}_m$ to the loss.

To remember the task $\mathcal{T}_k$ while learning future tasks, we expect to preserve the minimized loss as much as possible, which can be achieved by maintaining the stability of parameters that are important for the minimized loss while learning new tasks. Similar to EWC, we represent the importance of a parameter $\boldsymbol{w}_m$ as the magnitude of the gradient f_m based on Equation 3. All parameters W's importance to the loss of task $\mathcal{T}_k$ can be denoted as $I_k^{loss} = [\|f_m(X_k^{tr})\|]$, where I_k^{loss} is a matrix contains the importance scores of all parameters.

Topological Structure Preserving. We seek to find parameters that are important to the learned topological information. We first adopt GATs as the base model and model the attention coefficients between the center node and its neighbors as the topological information around the center node. We rewrite the attention coefficient $e_{ij}^{(l)}$ between node v_i and its neighbor v_j at the l-th layer, as shown in Equation 2, in the form of matrix:

$$e_{ij}^{(l)} = a(\boldsymbol{H}_{i,j}^{(l-1)}; W^{(l)}), \tag{4}$$

where a is a neural network and $W^{(l)}$ is the weight matrix of the l-th layer. $\boldsymbol{H}_{i,j}^{(l-1)}$ contains the embedding feature of node v_i and v_j computed from the $(l-1)$-th layer. Similar to Equation 3, the change in $e_{ij}^{(l)}$ caused by the infinitesimal change $\Delta W = \{\Delta \boldsymbol{w}_m\}$ can be approximated by:

$$\begin{aligned} a(\boldsymbol{H}_{i,j}^{(l-1)}; W^{(l)} + \Delta W^{(l)}) - a(\boldsymbol{H}_{i,j}^{(l-1)}; W^{(l)}) \\ \approx \sum_m g_m(\boldsymbol{H}_{i,j}^{(l-1)}) \Delta \boldsymbol{w}_m, \end{aligned} \tag{5}$$

where $g_m(\boldsymbol{H}_{i,j}^{(l-1)}) = \frac{\partial a}{\partial \boldsymbol{w}_m}$ is the gradient of the attention coefficient $e_{ij}^{(l)}$ with respect to the parameter $\boldsymbol{w}_m$.

Unlike the task-related loss that is already defined and is a single value, there is no objective function for the attention layer and the attention coefficients centered on node v_i at the l-th layer form a multi-dimensional vector $\boldsymbol{e}_i^{(l)} = [e_{i1}^{(l)}, ..., e_{i|\mathcal{N}_i|}^{(l)}]$. We define the topological loss for all nodes in $\mathcal{D}_k^{tr}$ as the squared l_2 norm of the multi-dimensional vector $\boldsymbol{e}_i^{(l)}$ at the l-th layer and compute the gradients as

$$g_m(\boldsymbol{H}^{(l-1)}) = \frac{\partial \left(\left| [e_1^{(l)}, ..., e_{|\mathcal{D}_k^{tr}|}^{(l)}] \right|_2^2 \right)}{\partial \boldsymbol{w}_m}, \tag{6}$$

where $\boldsymbol{H}^{(l-1)}$ is the output of the $(l-1)$-th layer. The importance scores of all parameters to the topological structure in task $\mathcal{T}_k$ can be represented as $I_k^{ts} = [\|g_m(\boldsymbol{H}_k^{(l-1)})\|]$, where I_k^{ts} is a matrix contains the importance scores of all parameters. In the experiments, we always adopt the attention coefficients computed in the middle layer of the GNN model as the topological information of graphs.

The final importance scores of all parameters W to the task $\mathcal{T}_k$ can be computed by

$$I_k = \lambda_l I_k^{loss} + \lambda_t I_k^{ts}, \tag{7}$$

where λ_l and λ_t are two hyper-parameters which together determine the importance of parameters. Changing of parameters with small importance scores does not affect the performance of the task much, and can minimize the loss for subsequent tasks.

Continual Learning on GNNs

After learning each task, we can measure the importance of all the parameters by using the proposed TWP module. When learning a new task $\mathcal{T}_{k+1}$, in addition to improve the performance of the new task, we also need to remember old ones by maintaining the stability of the important parameters for them, which can be achieved by penalizing changes to important parameters for old tasks. The loss function for the new task $\mathcal{T}_{k+1}$ is formulated as

$$\mathcal{L}_{k+1}^{'}(W) = \mathcal{L}_{k+1}^{new}(W) + \sum_{n=1}^{k} I_n \otimes (W - W_n^*)^2, \tag{8}$$

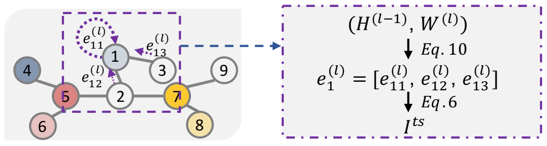

Figure 3: Extension to general GNN models at the l-th layer.

where $\otimes$ means element-wise multiplication. $\mathcal{L}_{k+1}^{new}(W)$ is the task-related loss function. I_n indicates how important the network parameters for the old task, and W_n^* contains the optimal parameters for the task $\mathcal{T}_n$. Such learning strategy ensures that parameters with low importance scores are free to change to adapt to the new task, while changing the parameters with high importance scores is penalized.

Promoting Minimized Importance Scores

Although maintaining the stability of the important parameters can make the model remember the learned tasks, there is a risk that the model will have less plasticity and cannot learn well on the subsequent tasks. In order to have sufficient model capacity reserved for the future tasks, we promote minimized on the computed importance scores of all parameters by adding a l_1 norm as a regularization to the importance scores computed from the current task. Then the total loss while training the current task $\mathcal{T}_{k+1}$ will become

$$\mathcal{L}_{k+1}(W) = \mathcal{L}_{k+1}^{'}(W) + \beta \left\| I_{k+1} \right\|_1, \tag{9}$$

where β is a hyper-parameter that controls the capacity preserved for future tasks. I_{k+1} contains the importance scores of all parameters for current task, which is computed based on the proposed TWP module. β affects the preserved capacity for future tasks. Higher β will preserve larger learning capacity for future tasks.

Extension to General GNNs

We show in this section that the proposed method can be easily extended to GNNs that may not hold an attention mechanism, such as GCNs and GINs, which makes it ready for arbitrary GNNs. As shown in Figure 3, we construct the topological structure by adding a non-parametric attention mechanism, which is formulated as

$$e_{ij}^{(l)} = (\boldsymbol{h}_i^{(l-1)} W^{(l)})^{\mathrm{T}} \tanh(\boldsymbol{h}_j^{(l-1)} W^{(l)}), \tag{10}$$

This makes the attention weights dependent on the distance between the node v_i and its neighbor v_j. Then, we normalize these distances across all neighbors of node v_i as the final coefficients. Given the constructed topological structure, we can directly employ the proposed TWP module to an arbitrary GNNs. Notice that although we construct the attention coefficients between nodes, we only use them for the TWP module. The models still use their original graph to aggregate and update the feature of nodes.

Experiments

We have performed comparative evaluation of our method against a wide variety of strong baselines, on four node classification datasets (transductive as well as inductive) and one graph classification dataset, on which our method achieves

Datasets	Methods	GATs		GCNs		GINs	
		AP (↑)	AF (↓)	AP (↑)	AF (↓)	AP (↑)	AF (↓)
Corafull	Fine-tune	51.6 ± 6.4%	46.1 ± 7.0%	34.0 ± 3.1%	63.3 ± 2.8%	30.9 ± 4.2%	46.1 ± 7.0%
	LWF	57.3 ± 2.3%	39.5 ± 3.1%	43.1 ± 5.8%	53.5 ± 7.2%	34.1 ±1.8 %	39.5 ± 3.1%
	GEM	84.4 ± 1.1%	4.2 ± 1.0%	80.5 ± 2.1%	5.5 ± 2.4%	78.0 ± 2.8%	3.5 ± 0.6%
	EWC	86.9 ± 1.7%	6.4 ± 1.8%	84.5 ± 3.0%	7.1 ± 2.3%	68.9 ± 11.7%	6.4 ± 1.8%
	MAS	84.1 ± 1.8%	8.6 ± 2.2%	86.1 ± 1.9%	4.8 ± 1.3%	73.7 ± 4.3%	8.6 ± 2.2%
	Ours	**89.0** ± 0.8%	**3.3** ± 0.3%	**87.8** ± 1.5%	**2.9** ± 1.1%	**78.4** ± 1.6%	**3.3** ± 0.3%
	Joint train	91.9 ± 0.8%	0.1 ± 0.2%	88.6 ± 1.5%	1.6 ± 1.0%	83.5 ± 1.6%	0.1 ± 0.2%
Amazon Computers	Fine-tune	86.5 ± 8.0%	12.3 ± 12.3%	78.0 ± 6.3%	24.7 ± 6.8%	66.2 ± 7.1%	34.9 ± 8.6%
	LWF	90.3 ± 6.4%	9.9 ± 7.0%	86.2 ± 5.6%	14.3 ± 5.4%	71.3 ± 3.3%	31.4 ± 7.8%
	GEM	97.1 ± 0.9%	0.7 ± 0.5%	97.1 ± 1.2%	0.9 ± 0.3%	91.9 ± 5.0%	4.2 ± 2.7%
	EWC	94.5 ± 3.3%	4.6 ± 4.5%	94.4 ± 2.5%	4.0 ± 3.4%	83.7 ± 6.2%	10.2 ± 6.1%
	MAS	94.0 ± 5.5%	5.0 ± 6.9%	96.8 ± 1.2%	1.3 ± 0.8%	92.4 ± 3.2%	**1.7** ± 3.4%
	Ours	**97.3** ± 0.6%	**0.6** ± 0.2%	**97.5** ± 0.6%	**0.6** ± 0.9%	**92.6** ± 1.0%	3.8 ± 1.1%
	Joint train	98.2 ± 0.6%	0.02 ± 0.1%	98.3 ± 0.4%	0.2 ± 0.3%	95.5 ± 0.6%	0.7 ± 1.3%
PPI	Fine-tune	0.365 ± 0.024	0.178 ± 0.019	0.366 ± 0.039	0.294 ± 0.010	0.598 ± 0.033	0.302 ± 0.002
	LWF	0.382 ± 0.024	0.185 ± 0.060	0.540 ± 0.026	0.338 ± 0.026	0.606 ± 0.011	0.332 ± 0.014
	GEM	0.741 ± 0.016	0.112 ± 0.030	0.628 ± 0.114	0.052 ± 0.086	0.710 ± 0.073	0.132 ± 0.002
	EWC	0.826 ± 0.027	0.142 ± 0.028	0.653 ± 0.009	0.050 ± 0.012	0.644 ± 0.144	0.038 ± 0.008
	MAS	0.749 ± 0.007	0.092 ± 0.008	0.656 ± 0.006	**0.014** ± 0.010	0.686 ± 0.008	0.082 ± 0.008
	Ours	**0.853** ± 0.004	**0.086** ± 0.005	**0.661** ± 0.010	0.038 ± 0.012	**0.715** ± 0.005	**0.013** ± 0.001
	Joint train	0.931 ± 0.040	0.035 ± 0.026	0.768 ± 0.003	0.003 ± 0.006	0.809 ± 0.037	0.010 ± 0.002
Reddit	Fine-tune	0.474 ± 0.006	0.580 ± 0.007	0.397 ± 0.064	0.670 ± 0.072	0.277 ± 0.068	0.382 ± 0.058
	LWF	0.500 ± 0.033	0.550 ± 0.034	0.441 ± 0.100	0.615 ± 0.113	0.313 ± 0.059	0.250 ± 0.119
	GEM	0.947 ± 0.001	0.030 ± 0.008	0.970 ± 0.004	0.014 ± 0.005	0.443 ± 0.045	0.135 ± 0.030
	EWC	0.944 ± 0.019	0.032 ± 0.021	0.917 ± 0.029	0.074 ± 0.007	0.394 ± 0.058	0.168 ± 0.132
	MAS	0.865 ± 0.031	0.085 ± 0.024	0.975 ± 0.002	0.002 ± 0.060	0.333 ± 0.063	0.184 ± 0.097
	Ours	**0.954** ± 0.014	**0.014** ± 0.015	**0.976** ± 0.002	**0.001** ± 0.062	**0.451** ± 0.043	**0.130** ± 0.036
	Joint train	0.978 ± 0.001	0.001 ± 0.001	0.978 ± 0.003	0.001 ± 0.002	0.582 ± 0.037	0.012 ± 0.028

Table 1: Performance comparison with different GNN backbones on different datasets. For the task performance, we use classification accuracy on Corafull and Amazon Computers datasets, and micro-averaged F1 score for PPI and Reddit datasets. The symbol ↑ (↓) indicates higher (lower) is better.

consistently better or on-par performances. In the following sections, we provide the details of datasets, baselines, experimental setup, the quantitative results and analysis.

Datasets

To provide a thorough evaluation, we adopt datasets related to both node-level and graph-level tasks. More information of the datasets and settings of graph continual learning can be found in the supplementary material.

Node Classification. For transductive learning, we utilize two widely used datasets named Corafull (Bojchevski and Günnemann 2017) and Amazon Computers (McAuley et al. 2015). We construct nine tasks with five classes per task on the Corafull dataset. Each task is a five-way node classification task. The model is trained on all tasks one-by-one and tested on all learned tasks. We conduct five tasks on Amazon Computers dataset and each task has two classes. For inductive learning, we use two datasets: a protein-protein interaction (PPI) dataset (Zitnik and Leskovec 2017) and a large graph dataset of Reddit posts (Hamilton, Ying, and Leskovec 2017). We follow the same dataset splitting protocol as (Hamilton, Ying, and Leskovec 2017). We perform 12 tasks for PPI and each task has ten classes; we generate eight tasks on Reddit and each one has five classes.

Graph Classification. We conduct experiment on a graph classification dataset, Tox21 (Huang et al. 2014), which contains qualitative toxicity measurements for 8014 compounds on 12 different targets. Each target results in a binary label. Each target in this dataset will be used for one task, leading to 12 tasks and each one is a binary classification task.

Baselines

Since there is no continual learning method designed for GNN models, we implement several continual learning methods designed for CNN models including Fine-tune (Girshick et al. 2014), LWF (Li and Hoiem 2017), EWC (Kirkpatrick et al. 2017), MAS (Aljundi et al. 2018), and GEM (Lopez-Paz and Ranzato 2017), and apply them on graph domain. Among these methods, Fine-tune can be seen as the lower bound without any continual learning mechanism. We also add Joint train method (Caruana 1997) as the approximate *upper bound* of continual learning since this method allows the access of data from all learned tasks during training. LWF utilizes the idea of knowledge distillation to remember the old tasks. EWC and MAS remember previous tasks by constraining changes to the important parameters. GEM alleviates forgetting based on an experience replay buffer which is used to avoid the increasing of losses associated to old tasks.

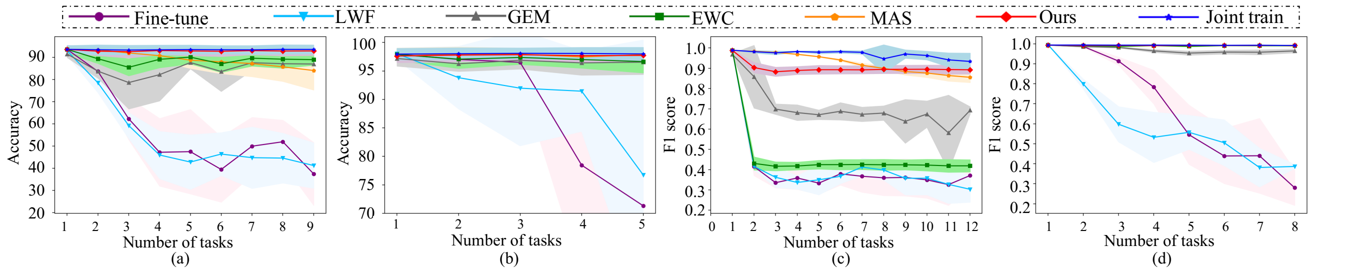

Figure 4: Performance of the first task on Corafull (a), Amazon Computers (b), PPI (c) and Reddit (d), as more tasks are learned.

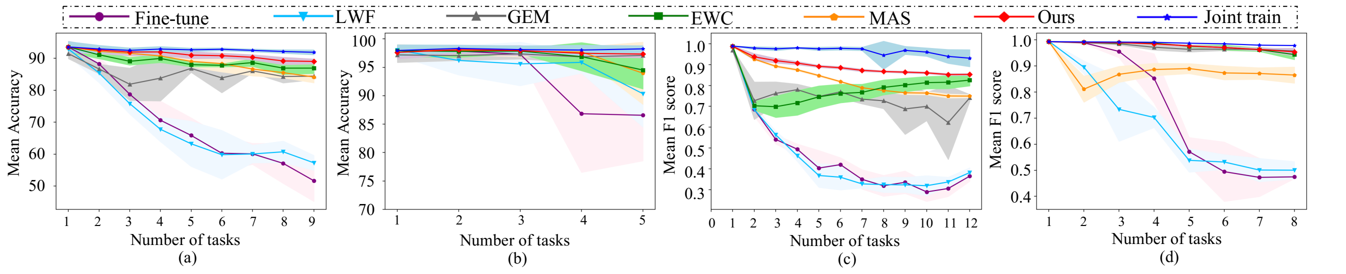

Figure 5: Average performance across all the learned tasks on Corafull (a), Amazon Computers (b), PPI (c), and Reddit (d), as more tasks are learned.

Metrics

The performance of graph continual learning can be evaluated from two different aspects named average performance (AP) and average forgetting (AF) (Lopez-Paz and Ranzato 2017), corresponding to the plasticity and stability of the model. More information about AP and AF could be found in the supplementary material. Basically, AP measures the average test performance across all learned tasks while AF measures the performance difference between after learning the particular task and after learning subsequent tasks. For the task performance, we use accuracy on Corafull and Amazon Computers datasets, micro F1 score on PPI and Reddit datasets, and AUC score on Tox21 dataset.

Experimental Setup

To validate the generalization ability, we conduct experiments based on three GNN backbones: GATs (Veličković et al. 2017), GCNs (Kipf and Welling 2016), and GINs (Xu et al. 2018). Since GATs uses attention mechanism to weight the neighbor for each node, we can directly employ the TWP module on it. For GCNs, we first compute the attention coefficients between each node and its neighbors according to Equation 10 and then use the TWP module to estimate the important scores of parameters. Considering GINs was proposed for the graph-level classification task, we first take the embedding feature before the pooling operation and feed them into a linear layer whose output acts as the prediction for the whole graph. Then we can compute the importance of parameters in the same way as GCNs. The detailed architecture of these three GNN backbones can be found in the supplementary material.

Adam SGD optimizer is used and the initial learning rate is set to 0.005 for all the datasets. The training epochs are set to 200, 200, 400, 30, and 100 for Corafull, Amazon Computers, PPI, Reddit, and Tox21 respectively. Early stopping is adopted for PPI and Tox21 where the patience is 10 for both of them. The regularizer hyper-parameter for EWC and MAS is always set to 10,000. The episodic memory for GEM contains all training nodes for Corafull and PPI datasets, and 100, 1,000, 100 training nodes for Amazon Computers, Reddit, and Tox21 datasets, respectively. For our method, λ_l is always set to 10,000, λ_t is selected from 100 and 10,000 for different datasets, and β is selected from 0.1 and 0.01. We set β globally for all tasks. We run five time with random seed and report the mean and stan-

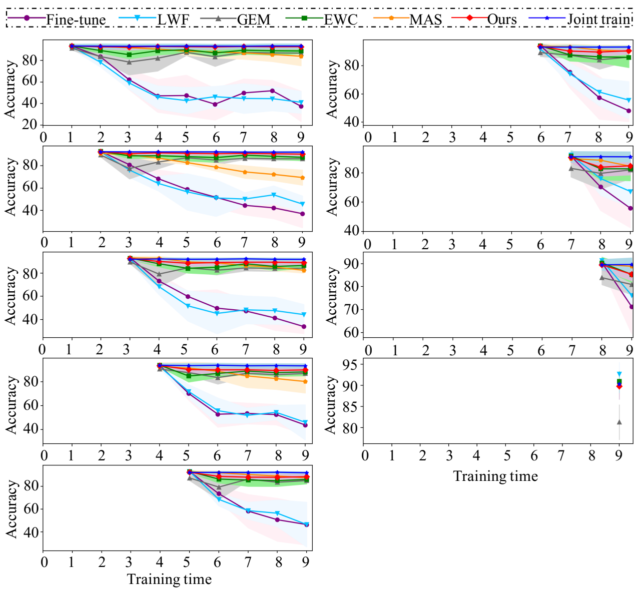

Figure 6: Evolution of performance for each task on GATs when gradually adding new tasks to the model. Different tasks are shown in different sub-graphs. The x-axis denotes the new task added each time and the y-axis denotes the performance of each task.

Methods	Corafull		Amazon Computers	
	AP (↑)	AF (↓)	AP (↑)	AF (↓)
W/_Loss	86.9 ± 1.7%	6.4 ± 1.8%	94.5 ± 3.3%	4.6 ± 4.5%
W/_TWP	88.6 ± 1.0%	3.7 ± 2.0%	96.2 ± 2.2%	0.8 ± 1.1%
Full	89.0 ± 0.8%	3.3 ± 0.3%	97.3 ± 0.6%	0.6 ± 0.2%

Table 2: Ablation study with GATs as the base model. W/_Loss and W/_TWP measures importance of parameters based on task-related loss and the proposed TWP module respectively. Full is the proposed method with all parts.

dard deviation for all methods and datasets.

Node Classification Task

Table 1 shows the results of performance comparison with baselines in terms of AP and AF for node classification task. We can clearly observe catastrophic forgetting in GNNs from the higher AF for Fine-tune. Our proposed method achieves best or second best performance across all datasets and GNN backbones. It is worth noting that, in some cases (e.g., PPI), some methods (e.g., MAS) perform better than other methods (e.g., EWC) in terms of AF, but become worse regards AP. These methods scarify much the performance of the new tasks in order to remember the old ones.

Figure 4 shows the evolution of performance for the first task as more tasks are learned. Figure 5 shows the average performance across all the learned tasks as more tasks are learned. Our method performs minimal forgetting among all the methods, and is very close to upper-bound. LWF, which adds new nodes to the network layer for each new task, performs inferior to other baselines even similar to the lower-bound in all the node classification datasets, indicating that LWF is not adapted well to GNNs. GEM, which prevents the loss for the previous tasks from increasing by storing a subset of the data of old tasks, holds unstable performances across different datasets and GNN backbones. Although GEM achieves comparable results on Amazon Computers and Reddit dataset, it performs inferior to other baselines on Corafull and PPI even though we put all training nodes of each task into the episodic memory for the GATs backbone. This can be partially explained by the fact that the stored data cannot play fully the role of remembering old knowledge. It is also worth noting that EWC and MAS, which aim to protect the important parameters when training the new task, work well on all these four datasets, which means that slowing down learning on important parameters is indeed applicable to GNNs. Our method further boosts the performance over EWC and MAS, which demonstrates the superior of the proposed TWP module in term of overcoming catastrophic forgetting on graph domain.

Figure 6 shows the training curves for nine tasks on Corafull dataset with GATs as the base model. The performance of our method degrees slower over time than the baselines throughout the training process, and is generally similar to that of the joint training. This is thanks to the capability of the proposed method to maintain the learned topological information of previous tasks.

Graph Classification Task

The experiment of the graph-level task is shown in Table 3, where Tox21 dataset is adopted. We use GATs as the base model and AUC as the metric to compute the continual learning performance. Our method boosts the forgetting performance by a significant margin, which demonstrates its effectiveness. We also conduct experiments on the Tox21 dataset with GCNs and GINs as the base models. More results can be found in the supplementary material.

Methods	GATs	
	AP (↑)	AF (↓)
Fine-tune	0.753 ± 0.024	0.084 ± 0.021
LWF	0.768 ± 0.041	0.058 ± 0.043
GEM	0.789 ± 0.010	0.036 ± 0.017
EWC	0.774 ± 0.025	0.038 ± 0.027
MAS	0.790 ± 0.023	0.032 ± 0.024
Ours	**0.801** ± 0.019	**0.022** ± 0.012
Joint train	0.831 ± 0.009	0.006 ± 0.009

Table 3: Performance comparison on the Tox21 dataset.

Ablation Study

We conduct ablation study to validate the effectiveness of each part of the proposed continual learning method. First, we conduct experiment based purely on the original loss of the task, namely, W/_Loss in Table 2. In this case, our method would degrade into EWC, which measures the importance of parameters only based on the task related loss function. Next, we verify the effectiveness of the topological information of graphs in graph-based continual learning by conducting experiments with the proposed TWP module. According to Table 2, TWP achieves consistently better performance on these two datasets, showing the effectiveness of preserving topological structure information to remember old tasks. Finally, we promote minimized importance scores for all parameters to preserve model capacity for future tasks, which further improves overall performance.

Conclusion

In this paper, we propose a dedicated continual learning method for graph neural networks, which is to our best knowledge the first attempt along this line. Specifically, we design a topology-aware weight preserving module which explicitly captures the topological information of graphs and measures the importance of the network's parameters based on the task-related loss function and the topological information. When learning a new task, changes to the important parameters will be penalized to remember old tasks. Moreover, the proposed approach can be readily extended to arbitrary GNNs. The extensive experiments on both node-level tasks and graph-level one demonstrates the effectiveness and applicability of the proposed continual learning method on the graph domain.

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Aljundi, R.; Lin, M.; Goujaud, B.; and Bengio, Y. 2019. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, 11816–11825.

Bojchevski, A.; and Günnemann, S. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via rankingfmccloskey1989catastrophic. *arXiv preprint arXiv:1707.03815* .

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* .

Caruana, R. 1997. Multitask learning. *Machine learning* 28(1): 41–75.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420* .

Chen, T.; Goodfellow, I.; and Shlens, J. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641* .

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, 3844–3852.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* .

French, R. M. 1994. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. *network* 1111: 00001.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3(4): 128–135.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Huang, R.; Sakamuru, S.; Martin, M. T.; Reif, D. M.; Judson, R. S.; Houck, K. A.; Casey, W.; Hsieh, J.-H.; Shockley, K. R.; Ceger, P.; et al. 2014. Profiling of the Tox21 10K compound library for agonists and antagonists of the estrogen receptor alpha signaling pathway. *Scientific reports* 4: 5664.

Javed, K.; and White, M. 2019. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, 1820–1830.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114(13): 3521–3526.

Lee, S.-W.; Kim, J.-H.; Jun, J.; Ha, J.-W.; and Zhang, B.-T. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, 4652–4662.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40(12): 2935–2947.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 6467–6476.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7765–7773.

McAuley, J.; Targett, C.; Shi, Q.; and Van Den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 43–52.

McClelland, J. L.; McNaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* 102(3): 419.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, 109–165. Elsevier.

Newman, M. E.; and Girvan, M. 2004. Finding and evaluating community structure in networks. *Physical review E* 69(2): 026113.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2017. Variational continual learning. *arXiv preprint arXiv:1710.10628* .

Pavlopoulos, G. A.; Secrier, M.; Moschopoulos, C. N.; Soldatos, T. G.; Kossida, S.; Aerts, J.; Schneider, R.; and Bagos, P. G. 2011. Using graph theory to analyze biological networks. *BioData mining* 4(1): 10.

Qiu, J.; Yang, Y.; Wang, X.; and Tao, D. 2020. Hallucinating visual instances in total absentia. In *European Conference on Computer Vision*.

Ratcliff, R. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review* 97(2): 285.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2001–2010.

Riemer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2019. Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. In *In International Conference on Learning Representations (ICLR)*.

Robins, A. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science* 7(2): 123–146.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* .

Schwarz, J.; Luketina, J.; Czarnecki, W. M.; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R.; and Hadsell, R. 2018. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370* .

Serrà, J.; Suris, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423* .

Shen, C.; Wang, X.; Song, J.; Sun, L.; and Song, M. 2019. Amalgamating Knowledge towards Comprehensive Classification. In *AAAI Conference on Artificial Intelligence*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, 2990–2999.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* .

Wang, X.; Türetken, E.; Fleuret, F.; and Fua, P. 2014. Tracking Interacting Objects Optimally Using Integer Programming. In *European Conference on Computer Vision*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* .

Yang, Y.; Feng, Z.; Song, M.; and Wang, X. 2020a. Factorizable Graph Convolutional Networks. *Advances in Neural Information Processing Systems* 33.

Yang, Y.; Qiu, J.; Song, M.; Tao, D.; and Wang, X. 2020b. Distilling Knowledge From Graph Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 7074–7083.

Yang, Y.; Wang, X.; Song, M.; Yuan, J.; and Tao, D. 2019. SPAGAN: Shortest Path Graph Attention Network. In *International Joint Conference on Artificial Intelligence*, 4099–4105.

Ye, J.; Ji, Y.; Wang, X.; Ou, K.; Tao, D.; and Song, M. 2019. Student Becoming the Master: Knowledge Amalgamation for Joint Scene Parsing, Depth Estimation, and More. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* .

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. In *International Conference on Machine Learning*, 7134–7143. PMLR.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995. JMLR. org.

Zitnik, M.; and Leskovec, J. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33(14): i190–i198.