Catastrophic Forgetting in Deep Graph Networks: an Introductory Benchmark for Graph Classification

Antonio Carta[*]
Department of Computer Science - University of Pisa
Pisa, Italy
antonio.carta@di.unipi.it

Andrea Cossu[*]
Scuola Normale Superiore
Pisa, Italy
andrea.cossu@sns.it

Federico Errica[*]
Department of Computer Science - University of Pisa
Pisa, Italy
federico.errica@phd.unipi.it

Davide Bacciu
Department of Computer Science - University of Pisa
Pisa, Italy
bacciu@di.unipi.it

ABSTRACT

In this work, we study the phenomenon of catastrophic forgetting in the graph representation learning scenario. The primary objective of the analysis is to understand whether classical continual learning techniques for flat and sequential data have a tangible impact on performances when applied to graph data. To do so, we experiment with a structure-agnostic model and a deep graph network in a robust and controlled environment on three different datasets. The benchmark is complemented by an investigation on the effect of structure-preserving regularization techniques on catastrophic forgetting. We find that replay is the most effective strategy in so far, which also benefits the most from the use of regularization. Our findings suggest interesting future research at the intersection of the continual and graph representation learning fields. Finally, we provide researchers with a flexible software framework to reproduce our results and carry out further experiments.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Neural networks**; *Online learning settings.*

KEYWORDS

deep graph networks, continual learning, lifelong learning, benchmarks

ACM Reference Format:
Antonio Carta, Andrea Cossu, Federico Errica, and Davide Bacciu. 2018. Catastrophic Forgetting in Deep Graph Networks: an Introductory Benchmark for Graph Classification. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

[*]Authors contributed equally to this research.

1 INTRODUCTION

Building a robust machine learning model that incrementally learns from different tasks without forgetting requires methodologies that account for drifts in the input distribution. The Continual Learning (CL) research field addresses the catastrophic forgetting problem [15, 16] by devising learning algorithms that improve a model's ability to retain previously gathered information. As of today, CL methods have been studied from the perspective of flat data [24, 28, 39] and, to a lesser extent, sequential data [11, 40].

Graph Representation Learning (GRL) is the study of machine learning models that can make predictions about input data represented as a graph. GRL methods naturally find application in social sciences [33], recommender systems [4], cheminformatics [32], security [21] and natural language processing [29], where the data is arbitrarily structured and cycles may occur [32].

At present, the literature lacks an analysis of catastrophic forgetting in models that deal with graphs. The few existing works focus on new approaches which are not compared to existing CL strategies on challenging benchmarks [44, 47]. This work makes the first step in this direction by carrying out continual learning experiments on graph classification benchmarks in a robust and controlled framework. In this context, we investigate whether specific GRL regularization strategies can mitigate catastrophic forgetting by enforcing structural information preservation.

Our contribution is two-fold. First of all, we study whether CL techniques for flat data still work on the graph domain. If that is not the case, the results will call for different and novel approaches to be developed. Secondly, we provide a robust and reproducible framework to carry out Continual Learning experiments on graph-structured data. Indeed the GRL field has suffered serious reproducibility issues that impacted chemical and social benchmarks [12]. By publicly releasing our code, we foster this trend to prevent common malpractices such as the usage of custom data splits, the absence of a model selection, and incorrect evaluations of the estimated risk on a validation (rather than test) set.

2 RELATED WORKS

This section introduces the high-level notions of both continual learning and deep learning for graphs.

2.1 Continual Learning

The main objective of CL is to learn from a continuous stream of data while mitigating catastrophic forgetting of previously acquired knowledge [34]. Continual learning models can be roughly categorized into three families: regularization strategies, architectural strategies and replay strategies. Though not entirely comprehensive, this taxonomy includes most of the currently used CL strategies.

Regularization strategies add a penalization to the standard loss function to enforce the stability of existing parameters. Elastic Weight Consolidation (EWC) [24] is one of the most used regularization strategies. It is importance-based, in the sense that it computes importance coefficients for each parameter at each step and penalizes drastic changes for important parameters. On the other hand, Learning without Forgetting (LwF) [26] leverages a distillation loss to keep the network's output close to its previous value.

Architectural strategies try to mitigate forgetting by enhancing the model's plasticity. Typically, they expand the network by adding more units [9, 30], an entirely new module [8, 36], or by expanding and then compressing the resulting architecture [20, 42]. These approaches require careful management of resources to avoid high computational costs.

Replay strategies mix input patterns from the current step with patterns from previously encountered steps [22, 35]. Replay memory management is crucial because it is not feasible to store all the patterns from previous steps. Generative replay, instead, overcomes this problem by training a generative model (with fixed space occupancy) that provides on-demand previous patterns [39, 43, 45].

2.2 Deep Learning for Graphs

When it comes to learning from input samples represented as a graph, classical recurrent or recursive approaches cannot deal with the mutual dependencies between the nodes, as these create cycles in the structure. There is a long and consolidated history of works that discuss these problems, with some of them dating back more than twenty years ago [14, 31, 37, 41]. Nowadays, the models that can process a broad spectrum of graphs by means of local and iterative processing of information are called Deep Graph Networks[1] (DGNs) [2]. Generally speaking, DGNs propagate nodes' information across the graph by stacking several graph convolutional layers on top of each other. Each layer works by aggregating each node's neighbouring information, and it ultimately produces node representations that can be used to make predictions about nodes, links, or entire graphs. For the sake of brevity, we refer the reader to recent works that summarize the state of the art [2, 3, 5, 46].

In what follows, we describe the CL strategies and deep graph networks used to evaluate catastrophic forgetting in the domain of graph-structured data; to the best of our knowledge, this is one of the first studies to investigate this particular aspect. To keep the discussion clear, we will focus on regularization and replay strategies applied to simple architectures for graphs, deferring more complex techniques to future studies.

[1]This term disambiguates the more common "Graph Neural Networks" (GNN), which refers to the work of [37].

3 TECHNIQUES

We now describe in more detail the CL techniques that we tested on Deep Graph Networks.

3.1 Elastic Weight Consolidation

Elastic Weight Consolidation [24] is a regularization technique which prevents changes in parameters that are important for previous steps. Formally, EWC adds a squared penalty term $\mathcal{R}$ to the classification loss at training time:

$$\mathcal{R}(\Theta, \Omega) = \lambda \sum_{i=1}^{n-1} \Omega_\mathbf{i} \|\Theta_\mathbf{i} - \Theta_\mathbf{n}\|_2^2, \tag{1}$$

where $\Theta_\mathbf{n}$ is the vector of parameters of current step n, $\Theta_\mathbf{i}$ is the vector of parameters from previous step i and $\Omega_\mathbf{i}$ is the vector of parameter importances for step i. The hyperparameter λ controls the trade-off between classification accuracy on current step and stability of parameters. The importance for step n is computed at the end of training on step n, through a diagonal approximation of the Fisher Information Matrix:

$$\Omega_\mathbf{n} = \mathbb{E}_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \left[(\nabla_{\Theta_\mathbf{n}} \log p_{\Theta_\mathbf{n}}(\mathbf{y}|\mathbf{x}))^2 \right]. \tag{2}$$

The computation of importance values requires an additional pass over the training data $\mathcal{D}$ and the estimation of the log probabilities $\log p_\Theta$ represented by the network outputs. Following [38], we keep a single importance matrix for all steps, by summing the importance on the current step with the previous values. In order to prevent the unbounded growth of importance values we normalize between 0 and 1 when computing importance on the current step.

3.2 Learning without Forgetting

Learning without Forgetting (LwF) [26] is a regularization technique which preserves the knowledge of previous steps by fostering stability at the activation level through knowledge distillation [18]. The method adds a regularization term $\mathcal{R}$ to the loss during step n as follows:

$$\mathcal{R}(\Theta_\mathbf{n}, \Theta_\mathbf{n-1}; \mathbf{x}, \mathbf{y}) = \alpha \, \mathrm{KL}[p_{\Theta_\mathbf{n}}(\mathbf{y}|\mathbf{x}) \, || \, p_{\Theta_\mathbf{n-1}}(\mathbf{y}|\mathbf{x})], \tag{3}$$

where α controls the regularization strength. The KL-divergence term prevents current activations to diverge too much from the ones of the model at previous step.

3.3 Replay

Replay of previous patterns during training is a very effective technique against forgetting of existing knowledge [1, 7, 17, 35]. We leveraged a replay memory which stores a fixed number of patterns for each class. During training on each step, the replay memory is concatenated with the training set. The resulting dataset is shuffled and used for training the model. Therefore, replay patterns are spread uniformly over the training set.

3.4 Naïve

The Naïve strategy trains the model continuously without applying any CL technique. This strategy is heavily subjected to catastrophic forgetting. Therefore, it can be used as a baseline to compare the performance of more effective CL strategies, which should perform significantly better in terms of forgetting.

3.5 Architectural Details

We define a graph as a tuple $g = (\mathcal{V}_g, \mathcal{E}_g, \mathcal{X}_g, \mathcal{A}_g)$ where $\mathcal{V}_g$ is the set of *nodes*, $\mathcal{E}_g$ is the set of oriented *edges* connecting nodes, whereas $\mathcal{X}_g$ (respectively $\mathcal{A}_g$) denotes node (edge) features. The neighbourhood $\mathcal{N}_v$ of a node v is the set of all nodes u for which an edge (u, v) directed towards v exists.

Structure-agnostic Baseline. To assess whether continual learning strategies have an impact when working with graphs, we must first devise a baseline that ignores the structural information and relies only on node features. The most common baseline we find in the literature [10, 12] is a multi-layer perceptron (MLP) that is invariant to the ordering of the nodes. Formally, the baseline compute a node representation $\mathbf{h}_v$ as follows

$$\mathbf{h}_v = \psi(\mathbf{x}_v), \quad x_v \in \mathcal{X}_g, \tag{4}$$

$$\psi(x_v) = \mathbf{W}_L^T(\sigma(\dots(\sigma(\mathbf{W}_1^T x_v + \mathbf{b}_1)\dots) + \mathbf{b}_L), \tag{5}$$

where $\psi(\cdot)$ is an MLP of L layers, the symbol $\mathbf{W}$ denotes a weight matrix and $\mathbf{b}$ is the bias. As the tasks under consideration in this paper deal with graph classification, an additional *readout* phase is necessary, in which we aggregate all node representations into a single graph representation $\mathbf{h}_g$:

$$\mathbf{h}_g = \Psi_g\Big(\{\mathbf{h}_v \mid v \in \mathcal{V}_g\}\Big), \tag{6}$$

where Ψ_g is a permutation invariant function; in this work we will use the *mean* function as the baseline's readout.

Deep Graph Networks. While DGNs usually adopt the same readout scheme as the one of Equation 6, the fundamental difference lies in its graph convolutional layer. If we assume a deep network of L layers, the node representation at layer $\ell < L$, that is, $\mathbf{h}_v^\ell$ is obtained by aggregating the neighbouring information of all nodes using another permutation invariant function Ψ_n:

$$\mathbf{h}_v^{\ell+1} = \phi^{\ell+1}\Big(\mathbf{h}_v^\ell,\ \Psi_n(\{\psi^{\ell+1}(\mathbf{h}_u^\ell) \mid u \in \mathcal{N}_v\})\Big), \tag{7}$$

where ϕ and ψ are usually implemented as linear layers or MLPs.

In our experiments, we define Ψ_n as the *mean* operator for digit classification tasks and sum for the chemical ones.

Structure-preserving Regularization Loss. We believe it is worth investigating whether a structure-preserving regularization loss such as the one of [23] affects catastrophic forgetting when used alongside the various CL strategies. The catch is that regularization will help preserve the output of previously seen classes when similar structural patterns appear in the new training samples. In general, the interplay between GRL and CL regularization strategies opens appealing research directions for the future. In case the chosen regularization does not help, this may indicate that the distribution of neighbour states of patterns belonging to a new class is radically different from those seen before.

4 EXPERIMENTS

This section provides a thorough description of the experimental details necessary to reproduce our experiments. The code is made publicly available to reproduce the results and carry out novel robust evaluations of different continual learning strategies[2].

[2]https://github.com/diningphil/continual_learning_for_graphs

	MNIST	CIFAR10	OGBG-PPA
Size	70000	60000	158100
Node Attrs.	3	5	0
Edge Attrs.	0	0	7
Classes	10	10	37
Avg $\|\mathcal{V}_g\|$	70,57	117,63	243,4
Avg $\|\mathcal{E}_g\|$	564,63	941,07	2266,1
Data Split	55K/5K/15K	45K/5K/15K	49%/29%/22%
Class Split	2+2+2+2+2	2+2+2+2+2	17+5+5+5+5

Table 1: Summary of the datasets' statistics. "Class split" refers to how we group classes in the Split CL experiment.

4.1 Datasets

The evaluation is carried out on three different large graph classification datasets. The former two, namely MNIST and CIFAR10, are the standard digit classification benchmarks used in the CL literature. However, here the digits are represented as graphs of varying dimension and shape [10]. The nodes are "superpixels" obtained through a specific coarsening process, and the adjacency information is constructed using the k-nearest neighbour algorithm. We defer the specifics of this process to the original paper. The third dataset is OGBG-PPA [19], a dataset of undirected protein association neighbourhoods taken from protein-protein interaction graphs. Here the task is to classify each input as one of 37 different taxonomy groups. Here, node features are missing but edges contain information. As such, we treat edges as nodes in the structure-agnostic baseline. We use the same data splits as those provided in the original papers, thus performing standard hold-out model selection and assessment. We also use the readily available version of all datasets provided by the Pytorch Geometric library [13]. Table 1 summarizes some useful dataset statistics.

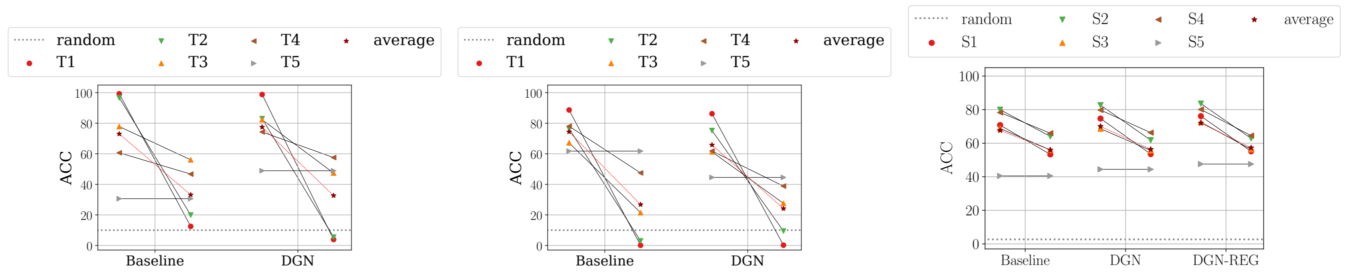
4.2 Experimental Setup

We evaluate each model in the class-incremental scenario, a popular continual learning setting where new classes arrive over time. When a new steps arrives, the model is trained on the new data without using data from the previous steps (except for the replay buffer). We use single-head models, where the entire output layer is used at each step. Table 1 shows the class splits for each dataset. To select the best hyperparameters for each strategy (see Appendix A), we perform the model selection on a separate validation set. The best hyperparameters found during the model selection are used for the model assessment on the test set. We monitor the metric $\text{ACC} = \frac{1}{T}\sum_{t=1}^{T} R_{T,t}$, introduced in [27], where $R_{T,t}$ is the accuracy on step t after training on step T.

We report the average ACC and its standard deviation computed over 5 runs. We evaluate the performance by computing the mean accuracy over all the steps after training on all steps.

5 RESULTS

The empirical results suggest that Deep Graph Networks trained continuously are subjected to catastrophic forgetting of previous knowledge. Table 2 reports the average ACC across all steps. We



(a) MNIST + LWF **(b) CIFAR10 + LWF** **(c) OGBG-PPA + REPLAY**

Figure 1: Paired plots showing the ACC on each step for different models. Each column refers to a model and it is composed by pairs of connected points. Each pair refers to a specific step. The leftmost point in the pair represents ACC after training on that specific step. The rightmost point represents ACC after training on all steps. The more vertical the line connecting the points, the larger the forgetting effect. The dashed horizontal line indicates the performance of a random classifier. The red star represents the average performance over all steps.

also extend the results presented in [25] to Deep Graph Networks: importance-based regularization strategies are not able to prevent forgetting in class-incremental scenarios. In fact, in our experiments EWC always performs comparably to the *Naïve* strategy.
Interestingly, Deep graph networks do not provide significant performance improvements with respect to a structure-agnostic baseline. This is a surprising result, which might have two complementary explanations. The first is that the neighboring states' distribution of different classes varies, thus making the previously trained graph convolutions inadequate for subsequent tasks. The second, instead, relates to the nature of the class-incremental scenario. Since the model sees few classes at a time, each training task becomes so simple that the model ends up relying on node features only to discern between the two classes. This is confirmed by the fact that, when encouraged to retain structural information via the regularization term, DGN shows a slight increase in performance with the replay strategy. We believe that addressing both points in more detail could constitute interesting future work at the intersection of the two research fields.

Not all regularization strategies are, however, subjected to forgetting. In fact, we show that LwF is able to recover part of the original knowledge, outperforming both Naïve and EWC. We also found LwF to be very sensitive to the choice of the hyperparameters (Appendix C). In particular, the softmax temperature and the hyperparameter α, which controls the amount of knowledge distillation heavily influence the final performance. This limits the applicability of LwF in real world applications due to the constraints of model selection in continual learning scenarios [6].

Replay strategy is considered among the strongest CL strategies available. In our experiments, replay consistently outperforms all the other strategies. Appendix B expands on replay by showing the final performance under different replay memory sizes. Deep graph networks and baseline models require a comparable amount of replay to obtain the same level of performance. Therefore, replay seems to behave as a good model-agnostic strategy even in the domain of graphs.

	Model	Strategy			
		Naïve	EWC	Replay	LWF
MNIST	Baseline	$19.56_{\pm 0.1}$	$19.39_{\pm 0.1}$	$86.13_{\pm 4.5}$	$33.16_{\pm 13.1}$
	DGN	$19.19_{\pm 0.1}$	$18.95_{\pm 0.3}$	$79.52_{\pm 1.9}$	$32.64_{\pm 5.0}$
	DGN+reg	$19.31_{\pm 0.1}$	—	$81.42_{\pm 2.4}$	—
CIFAR10	Baseline	$17.49_{\pm 0.1}$	$17.49_{\pm 0.1}$	$42.87_{\pm 3.7}$	$26.77_{\pm 5.1}$
	DGN	$17.11_{\pm 0.2}$	$17.10_{\pm 0.2}$	$39.55_{\pm 2.3}$	$24.13_{\pm 4.1}$
	DGN+reg	$17.13_{\pm 0.1}$	—	$46.61_{\pm 3.5}$	—
OGBG-PPA	Baseline	$14.53_{\pm 0.5}$	$13.90_{\pm 0.8}$	$55.96_{\pm 3.0}$	$20.83_{\pm 6.1}$
	DGN	$14.47_{\pm 0.3}$	$14.15_{\pm 0.5}$	$56.34_{\pm 2.5}$	$18.46_{\pm 5.4}$
	DGN+reg	$15.18_{\pm 0.8}$	—	$57.27_{\pm 3.2}$	—

Table 2: Mean accuracy and mean standard deviation (in parenthesis) among all steps. Replay results are related to memory size of 1000. Results are averaged over 5 final runs. We treat the regularization loss as a separate strategy.

6 CONCLUSIONS

Learning from a data stream in a continual fashion is fundamental for real-world applications. In this paper, we show that deep graph networks suffer from catastrophic forgetting in class-incremental settings. Interestingly, while graph networks outperform feedforward baselines during offline training, our results show that this advantage disappears in continual learning scenarios. While our preliminary results suggest that regularization techniques for DGN may help, the results are still far from the performance achieved in the offline setting. This suggests that more research is needed to explore whether alternative DGN or regularization techniques may be better able to exploit the graph structure and learn robust features. We release our code and baseline models hoping to foster additional research in this direction.

REFERENCES

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. 2019. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 11849–11860.

[2] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. 2020. A Gentle Introduction to Deep Learning for Graphs. *Neural Networks* 129 (9 2020), 203–221.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, and others. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[4] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based systems* 46 (2013), 109–132. Publisher: Elsevier.

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 25. 18–42.

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient Lifelong Learning with A-GEM. In *ICLR*.

[7] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip H S Torr, and Marc'Aurelio Ranzato. 2019. On Tiny Episodic Memories in Continual Learning. *arXiv* (2019).

[8] Andrea Cossu, Antonio Carta, and Davide Bacciu. 2020. Continual Learning with Gated Incremental Memories for sequential data processing. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN 2020)*.

[9] Timothy John Draelos, Nadine E Miner, Christopher Lamb, Jonathan A Cox, Craig Michael Vineyard, Kristofor David Carlson, William Mark Severa, Conrad D James, and James Bradley Aimone. 2017. Neurogenesis Deep Learning. In *IJCNN*.

[10] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982* (2020).

[11] Benjamin Ehret, Christian Henning, Maria R Cervera, Alexander Meulemans, Johannes von Oswald, and Benjamin F Grewe. 2020. Continual Learning in Recurrent Neural Networks with Hypernetworks. *arXiv* (2020).

[12] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

[13] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *Workshop on Representation Learning on Graphs and Manifolds, International Conference on Learning Representations (ICLR)* (2019).

[14] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. 1998. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks* 9, 5 (1998), 768–786. Publisher: IEEE.

[15] Robert French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3, 4 (1999), 128–135.

[16] Stephen Grossberg. 1980. How Does a Brain Build a Cognitive Code? *Psychological Review* 87, 1 (1980), 1–51. https://doi.org/10.1037/0033-295X.87.1.1

[17] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. 2018. Memory Efficient Experience Replay for Streaming Learning. *IEEE International Conference on Robotics and Automation (ICRA)* (2018).

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687* (2020).

[20] Steven C Y Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. 2019. Compacting, Picking and Growing for Unforgetting Continual Learning. In *NeurIPS*. 13669–13679.

[21] Giacomo Iadarola. 2018. *Graph-based classification for detecting instances of bug patterns.* Master's thesis. University of Twente.

[22] David Isele and Akansel Cosgun. 2018. Selective Experience Replay for Lifelong Learning. *Thirty-Second AAAI Conference on Artificial Intelligence* (2018), 3302–3309.

[23] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS* 114, 13 (2017), 3521–3526.

[25] Timothée Lesort, Andrei Stoian, and David Filliat. 2020. Regularization Shortcomings for Continual Learning. *arXiv* (2020).

[26] Zhizhong Li and Derek Hoiem. 2016. Learning without Forgetting. In *European Conference on Computer Vision (Springer)*. 614–629.

[27] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. In *NIPS*.

[28] Davide Maltoni and Vincenzo Lomonaco. 2018. Continuous Learning in Single-Incremental-Task Scenarios. *arXiv* (2018).

[29] Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 2 (Short Papers)*. 486–492.

[30] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. 2002. A self-organising network that grows when required. *Neural Networks* 15, 8-9 (2002), 1041–1058.

[31] Alessio Micheli. 2009. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks* 20, 3 (2009), 498–511. Publisher: IEEE.

[32] Alessio Micheli, Alessandro Sperduti, and Antonina Starita. 2007. An introduction to recursive neural networks and kernel methods for cheminformatics. *Current Pharmaceutical Design* 13, 14 (2007), 1469–1496.

[33] Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. 2018. SocialLink: exploiting graph embeddings to link DBpedia entities to Twitter profiles. *Progress in Artificial Intelligence* 7, 4 (2018), 251–272. Publisher: Springer.

[34] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71.

[35] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2019. Experience Replay for Continual Learning. In *NeurIPS*. 350–360.

[36] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *arXiv* (2016).

[37] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. Publisher: IEEE.

[38] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & Compress: A scalable framework for continual learning. In *International Conference on Machine Learning*. 4528–4537.

[39] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems 30*, I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett (Eds.). Curran Associates, Inc., 2990–2999.

[40] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. 2020. Toward Training Recurrent Neural Networks for Lifelong Learning. *Neural Computation* 32, 1 (2020), 1–35.

[41] Alessandro Sperduti and Antonina Starita. 1997. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks* 8, 3 (1997), 714–735. Publisher: IEEE.

[42] Shivangi Srivastava, Maxim Berman, Matthew B Blaschko, and Devis Tuia. 2019. Adaptive Compression-based Lifelong Learning. In *BMVC*.

[43] Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications* 11 (2020).

[44] Chen Wang, Yuheng Qiu, and Sebastian Scherer. 2020. Lifelong Graph Learning. *arXiv:2009.00647* (2020).

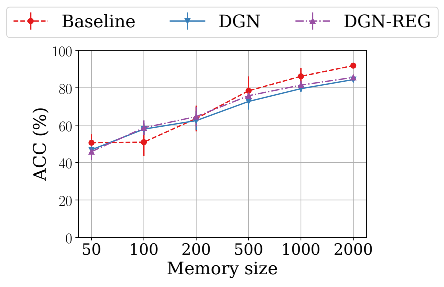
[45] Zhepei Wang, Cem Subakan, Efthymios Tzinis, Paris Smaragdis, and Laurent Charlin. 2019. Continual Learning of New Sound Classes using Generative Replay. *arXiv* (2019).

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

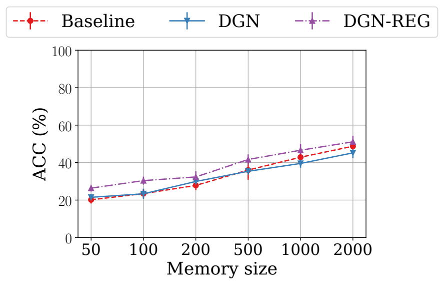
[47] Fan Zhou and Chengtai Cao. 2021. Overcoming Catastrophic Forgetting in Graph Neural Networks with Experience Replay. *arXiv:2003.09908* (2021).

A HYPER-PARAMETERS

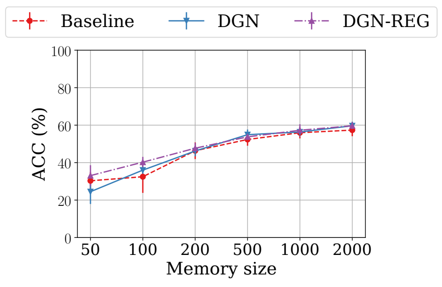
We perform model selection on the validation set using a grid-search strategy for all the implemented models. Regardless of the dataset or continual learning technique used, we selected the number of layers in $\{2, 4\}$ for the DGN and 4 for the baseline. In both cases, the dimension of the hidden layer was chosen in $\{64, 128\}$. The number of epochs was set to 200 (patience = 20) for the Baseline and to 1000 for DGN and DGN+Reg (patience = 50). The learning rate was set to 0.001, and the optimizer chosen was Adam. We used the "sum" version of the EWC combined with normalized importance scores. Being LWF very sensible to the hyper-parameters, we chose $\alpha \in \{0.5, 1., 2.\}$ and the temperature in $\{0.5, 1., 2.\}$.



(a) MNIST



(b) CIFAR10



(c) OGBG-PPA

Figure 2: ACC for increasing replay memory size.

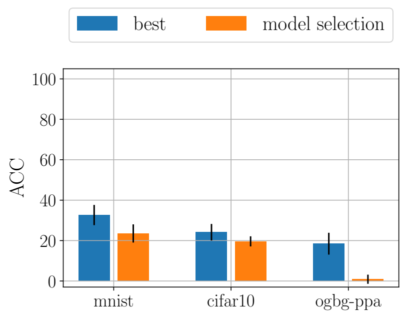


Figure 3: Comparison of performances between model selection (averaged across all configurations) and model assessment (averaged across 5 final training runs). The difference highlights the sensitivity of LwF to the choice of hyperparameters.

B ADDITIONAL REPLAY EXPERIMENTS

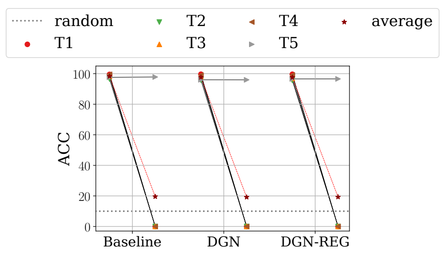
Figure 2 shows ACC values for increasing replay memory sizes.

C SENSITIVITY OF LWF TO HYPERPARAMETERS

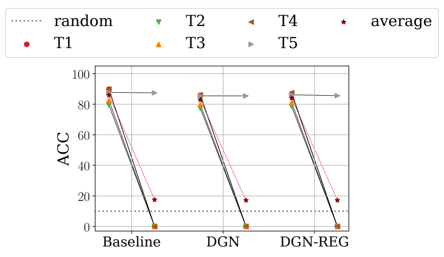
We briefly show the sensitivity of LwF to the choice of hyperparameters (Figure 3). We compute the mean ACC and its standard deviation across all runs of model selection. Then, we compare the results with the best performance we found during model assessment. The difference highlights the sensitivity of Lwf to the hyper-parameters.

D VISUALIZATION OF RESULTS WITH PAIRED PLOTS

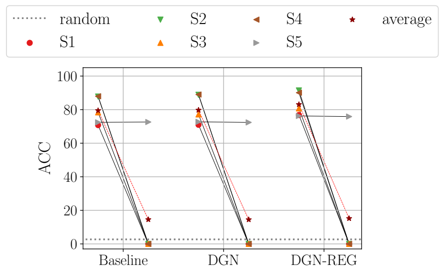
For completeness, Figure 4 reports the paired plots for all the CL techniques and datasets tested in this work.



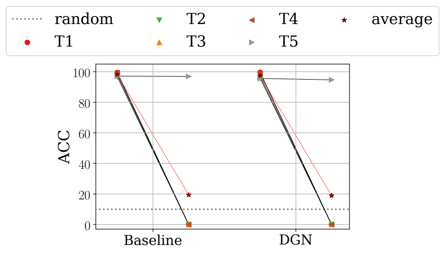
(a) MNIST + Naïve



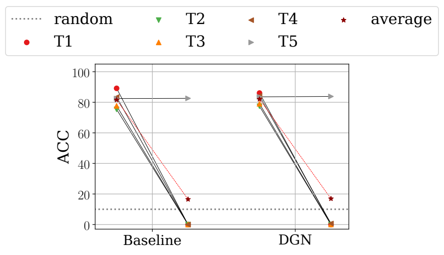
(b) CIFAR10 + Naïve



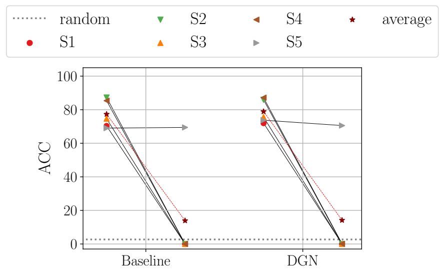
(c) OGBG-PPA + Naïve



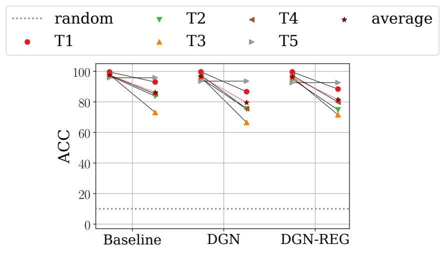
(d) MNIST + EWC



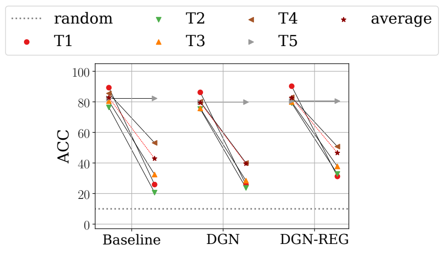
(e) CIFAR10 + EWC



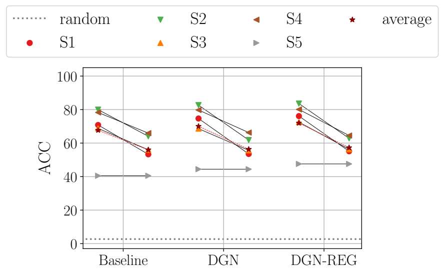
(f) OGBG-PPA + EWC



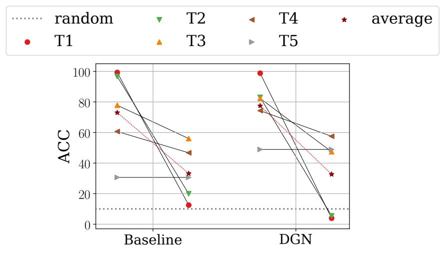
(g) MNIST + REPLAY



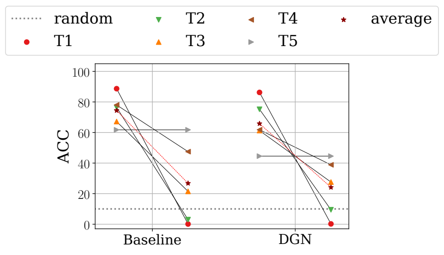
(h) CIFAR10 + REPLAY



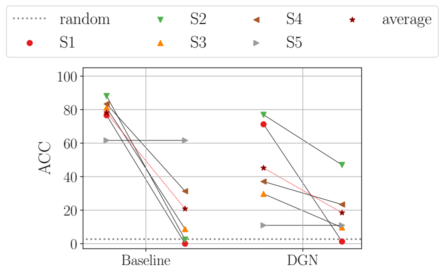
(i) OGBG-PPA + REPLAY



(j) MNIST + LwF



(k) CIFAR10 + LwF



(l) OGBG-PPA + LwF

Figure 4: Additional paired plots for the experiments. Refer to Figure 1 for a description of paired plots.