Topological Continual Learning with Wasserstein Distance and Barycenter

Tananun Songdechakraiwut
University of Wisconsin–Madison
songdechakra@wisc.edu

Xiaoshuang Yin
Google

Barry D. Van Veen
University of Wisconsin–Madison

Abstract

Continual learning in neural networks suffers from a phenomenon called catastrophic forgetting, in which a network quickly forgets what was learned in a previous task. The human brain, however, is able to continually learn new tasks and accumulate knowledge throughout life. Neuroscience findings suggest that continual learning success in the human brain is potentially associated with its modular structure and memory consolidation mechanisms. In this paper we propose a novel topological regularization that penalizes cycle structure in a neural network during training using principled theory from persistent homology and optimal transport. The penalty encourages the network to learn modular structure during training. The penalization is based on the closed-form expressions of the Wasserstein distance and barycenter for the topological features of a 1-skeleton representation for the network. Our topological continual learning method combines the proposed regularization with a tiny episodic memory to mitigate forgetting. We demonstrate that our method is effective in both shallow and deep network architectures for multiple image classification datasets.

1 Introduction

Neural networks can be trained to achieve impressive performance on a variety of learning tasks. However, when an already trained network is further trained on a new task, a phenomenon called *catastrophic forgetting* [16] occurs, in which previously learned tasks are quickly forgotten with additional training. The human brain, however, is able to continually learn new tasks and accumulate knowledge throughout life without significant loss of previously learned skills. Neuroscience findings suggest that the principle of modularity [7] may play an important role. Modular structures [23] are aggregates of modules that perform specific functions without perturbing one another. Human brains are characterized by modular structures during learning [5]. Such structures are hypothesized to reduce the interdependence of components, enhance robustness, and facilitate learning [1].

Persistent homology has emerged as a tool for understanding, characterizing and quantifying the topology of brain networks by interpreting brain networks as 1-skeletons of a simplicial complex [24, 25]. The topology of a 1-skeleton is *completely* characterized by connected components and cycles. In this paper we show that persistent homology can be used to improve performance of neural networks in continual learning tasks. In particular, we interpret a network as a 1-skeleton and propose a novel *topological regularization* of the 1-skeleton's cycle structure to avoid catastrophic forgetting of previously learned tasks. Regularizing the cycle structure allows the network to explicitly learn its complement, i.e., the modular structure, through gradient optimization. Our approach is made computationally efficient by use of the closed form expressions for the Wasserstein barycenter and the gradient of Wasserstein distance between network cycle structures. We evaluate our approach using image classification across multiple datasets and show it generally improves continual learning performance compared to competing approaches in the challenging case of both shallow and deep networks of limited width.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

2 Efficient Topological Representation and Measure for Neural Networks

Birth and Death Decomposition Represent a neural network as an undirected weighted graph $G = (V, \mathbb{W})$ with a set of nodes V, and a set of edge weights $\mathbb{W} = \{w_{i,j}\}$. Create a binary graph G_ϵ with the identical node set V by thresholding the edge weights so that an edge between nodes i and j exists if $w_{i,j} > \epsilon$. The binary graph is considered as a *1-skeleton* [17]. The only non-trivial topological features in a 1-skeleton are *connected components* and *cycles*. As ϵ increases, more and more edges are removed from the network G, resulting in a *graph filtration* [12]: $G_{\epsilon_0} \supseteq G_{\epsilon_1} \supseteq \cdots \supseteq G_{\epsilon_k}$, where $\epsilon_0 \leq \epsilon_1 \leq \cdots \leq \epsilon_k$ are called filtration values. If a topological feature is born at a filtration value b_l and persists up to a filtration value d_l over ϵ, then this feature is represented as a 2D *persistence* point (b_l, d_l). The set of all points $\{(b_l, d_l)\}_l$ is called *persistence barcode* [6]. The graph filtration simplifies the barcodes to 1D descriptors [24]. Specifically, the representation of the connected components can be simplified to a collection of *birth values* $\mathbb{B}(G) = \{b_l\}$ and that of cycles to a collection of *death values* $\mathbb{D}(G) = \{d_l\}$. In addition, neural networks of the same architecture have a birth set $\mathbb{B}$ and a death set $\mathbb{D}$ of the *same* cardinality as $|V| - 1$ and $|\mathbb{W}| - (|V| - 1)$, respectively, which resolve the problem of point mismatch in barcodes for same-architecture networks.

Closed Form Wasserstein Distance and Gradient Wasserstein distance for cycle structure depends solely on the death sets. Let G, H be two given networks based on the same architecture. Their *Wasserstein distance for cycles* has a closed-form expression that allows for very efficient computation [24] as $W^2_{cycle}(G, H) = \sum_{d_l \in \mathbb{D}(G)} \big[d_l - \phi^*(d_l)\big]^2$, where ϕ^* maps the l-th smallest death value in $\mathbb{D}(G)$ to the l-th smallest death value in $\mathbb{D}(H)$ for all l. In addition, the gradient of Wasserstein distance for cycles $\nabla_G W^2_{cycle}(G, H)$ also has a closed-form expression as [25] $\partial W^2_{cycle}(G, H)/\partial d_l = 2\big[d_l - \phi^*(d_l)\big]$.

Closed Form Wasserstein Barycenter Wasserstein barycenter is the mean of a collection of networks under Wasserstein distance and represents a topological centroid. Consider same-architecture networks $G^{(1)}, ..., G^{(N)}$. Let $\mathbb{D}(G^{(i)}) : d^{(i)}_1 \leq \cdots \leq d^{(i)}_{|\mathbb{D}|}$ be the death set of network $G^{(i)}$. *Wasserstein barycenter for cycles* $\mathcal{G}_{cycle}$ has a closed form expression as $\mathcal{G}_{cycle} : \overline{d}_1 \leq \cdots \leq \overline{d}_{|\mathbb{D}|}$, where $\overline{d}_l = \sum_{i=1}^N \nu_i d^{(i)}_l \Big/ \sum_{i=1}^N \nu_i$. The complete proof is given in Appendix A.

3 Topological Continual Learning

Consider a *continual learning* scenario in which T supervised learning tasks are learned sequentially. Each task has a task descriptor $\tau \in \mathbb{T} = \{1, 2, ..., T\}$ with a corresponding dataset $\mathbb{P}_\tau = \{(\boldsymbol{x}_{i,\tau}, \boldsymbol{y}_{i,\tau})_{i=1}^{N_\tau}\}$ containing N_τ labeled training examples consisting of a feature vector $\boldsymbol{x}_{i,\tau} \in \mathcal{X}$ and a target vector $\boldsymbol{y}_{i,\tau} \in \mathcal{Y}$. We further consider the continuum of training examples that are experienced *only once*, and assume that the continuum is *locally independent and identically distributed* (iid), i.e., $(\boldsymbol{x}_{i,\tau}, \boldsymbol{y}_{i,\tau}) \overset{iid}{\sim} P_\tau$ following the prior work of Lopez-Paz & Ranzato [15]. The goal is to train a model $f : \mathcal{X} \times \mathbb{T} \to \mathcal{Y}$ that predicts a target vector $\boldsymbol{y}$ corresponding to a test pair $(\boldsymbol{x}, \tau)$, where $(\boldsymbol{x}, \boldsymbol{y}) \sim P_\tau$.

Our approach to addressing this problem is to topologically penalize training with future tasks based on the underlying 1-skeleton of the neural network. Define a neural network $G^{(\tau)}$ for learning task τ with nodes given by neurons, and edge weights defined by the weight/parameter set $\mathbb{W}$. All past-task networks $G^{(j)}$, for $j = 1, ..., \tau - 1$, have the identical node sets with the trained weight set $\mathbb{W}^*_j$ denoting the weights after training through the entire sequence up to task j. Since these graphs have the same architecture, their death sets have the same cardinality denoted by $|\mathbb{D}|$. Then the birth-death decomposition of the weight set $\mathbb{W}^*_j$ results in the death set $\mathbb{D}(G^{(j)}) : d^{(j)}_1 \leq \cdots \leq d^{(j)}_{|\mathbb{D}|}$. The Wasserstein barycenter for cycles of the first $\tau - 1$ training tasks associated with networks $G^{(1)}, ..., G^{(\tau-1)}$ is $\mathcal{G}^{(\tau-1)}_{cycle} : \overline{d}^{(\tau-1)}_1 \leq \cdots \leq \overline{d}^{(\tau-1)}_{|\mathbb{D}|}$, where $\overline{d}^{(\tau-1)}_l = \sum_{j=1}^{\tau-1} \nu_j d^{(j)}_l \Big/ \sum_{j=1}^{\tau-1} \nu_j$. Our approach to learning task τ minimizes the empirical risk minimization loss (ERM) with the Wasserstein distance and barycenter penalty $\mathcal{L}_\tau(\mathbb{W}) = \mathcal{L}_{ERM,\tau}(\mathbb{W}) + \frac{\lambda}{2} W^2_{cycle}(G^{(\tau)}, \mathcal{G}^{(\tau-1)}_{cycle})$ for all task $\tau > 1$, where λ controls relative importance between past- and current-task cycle structure. Intuitively, we penalize changes of cycle structure in a neural network while allowing the network to explicitly learn the modular structure represented by births of connected components.

Table 1: ACC and BWT performance for P-MNIST, R-MNIST, split CIFAR and split miniImageNet datasets. The mean and standard deviation over five different task sequences are shown.

		P-MNIST		R-MNIST	
Methods	Memory	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Finetune	N	34.44 ± 2.07	-58.89 ± 2.23	41.43 ± 2.09	-55.42 ± 2.09
EWC	N	48.05 ± 1.27	-44.55 ± 1.46	39.80 ± 2.01	-55.26 ± 2.03
RGO	N	73.18 ± 0.57	-19.86 ± 0.61	63.34 ± 0.96	-29.30 ± 0.95
A-GEM	Y	59.50 ± 1.20	-33.63 ± 1.24	53.38 ± 0.90	-43.35 ± 0.93
ORTHOG-SUB	Y	42.83 ± 1.22	-9.94 ± 1.05	23.85 ± 0.84	-3.45 ± 1.08
ER-Res	Y	66.38 ± 1.29	-25.02 ± 1.51	72.54 ± 0.36	-23.48 ± 0.39
ER-Ring	Y	70.10 ± 0.89	-23.19 ± 1.01	70.52 ± 0.51	-25.72 ± 0.51
TOP-Res	Y	68.13 ± 0.66	-24.50 ± 0.66	74.33 ± 0.66	-20.80 ± 0.63
TOP-Ring	Y	71.05 ± 0.80	-22.18 ± 0.86	72.12 ± 0.81	-23.85 ± 0.84
Multitask	–	90.31	–	93.43	–
		Split CIFAR		Split miniImageNet	
Methods	Memory	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Finetune	N	40.62 ± 5.09	-23.80 ± 5.31	33.13 ± 2.72	-24.95 ± 2.30
EWC	N	38.26 ± 3.71	-25.30 ± 4.57	33.48 ± 1.79	-19.56 ± 2.24
RGO	N	38.93 ± 1.03	-18.98 ± 0.89	42.03 ± 1.22	-14.19 ± 1.56
A-GEM	Y	43.54 ± 6.23	-23.25 ± 5.65	39.52 ± 4.10	-18.48 ± 4.39
ORTHOG-SUB	Y	37.93 ± 1.59	-5.44 ± 1.37	32.36 ± 1.44	-5.52 ± 1.07
ER-Res	Y	43.28 ± 1.26	-23.08 ± 1.51	38.51 ± 2.40	-13.58 ± 3.49
ER-Ring	Y	52.75 ± 1.18	-14.51 ± 1.79	44.67 ± 1.81	-12.23 ± 1.79
TOP-Res	Y	45.92 ± 1.50	-20.10 ± 1.00	39.95 ± 1.91	-14.34 ± 2.06
TOP-Ring	Y	54.27 ± 1.54	-11.70 ± 1.27	49.08 ± 1.71	-8.42 ± 1.48
Multitask	–	61.08	–	57.99	–

The minimization is accomplished via gradient descent over all training samples in learning task τ using the closed form Wasserstein gradient. In other words, the gradient computation is achieved in two steps: 1) compute birth-death decomposition of $\mathbb{W}$ to find $\mathbb{D}(G^{(\tau)})$; 2) sort $\mathbb{D}(G^{(\tau)})$ to find the optimal matching ϕ^* between $\mathbb{D}(G^{(\tau)})$ and $\mathcal{G}_{cycle}^{(\tau-1)}$. Furthermore, we execute the first step every m iterations. When $m = 1$, we compute birth-death decomposition every iteration to determine which edge belongs to $\mathbb{D}(G^{(\tau)})$ before sorting, while $m > 1$ allows to utilize multiple gradient descent updates to consolidate changes to previously determined edges in $\mathbb{D}(G^{(\tau)})$ by directly sorting updated weights in $\mathbb{D}(G^{(\tau)})$ from previous iterations. This approach may mimic human learning and memory consolidation over multiple temporal scales from days to decades [1].

At any time, we only need to store *one* Wasserstein barycenter from the previous task $\mathcal{G}_{cycle}^{(\tau-1)} = \{\overline{d}_l^{(\tau-1)}\}_l$. The barycenter for the current task $\mathcal{G}_{cycle}^{(\tau)} = \{\overline{d}_l^{(\tau)}\}_l$ can be computed as: $\overline{d}_l^{(\tau)} = (p\overline{d}_l^{(\tau-1)} + qd_l^{(\tau)})/(p+q)$, where $d_l^{(\tau)} \in \mathbb{D}(G^{(\tau)}(\mathbb{W}_\tau^*))$ and $p, q > 0$ used to emphasize the contribution of $\mathcal{G}_{cycle}^{(\tau-1)}$ and $\mathbb{D}(G^{(\tau)})$ to $\mathcal{G}_{cycle}^{(\tau)}$. This online update satisfies the closed-form Wasserstein barycenter. A proof by induction is provided in Appendix A.

Pseudo-code and related work are provided in Appendices B and C, respectively.

4 Image Classification Experiments

Datasets Experiments are performed on four datasets: (1) *Permuted MNIST* (P-MNIST) [9], (2) *Rotated MNIST* (R-MNIST) [15], (3) *Split CIFAR* [10] and (4) *split miniImageNet* [20, 26].

Network Architecture The P-MNIST and R-MNIST use a fully-connected neural network with two hidden layers each with 128 neurons. The split CIFAR and split miniImageNet use a downsized version of ResNet18 [8] with eight times fewer feature maps across all layers, similar to [15].

Method Comparison We evaluate our method performance in relative to eight baseline approaches that learn a sequence of tasks in a fixed-size network architecture: (1) *finetune*, i.e., a model trained sequentially without any regularization and past-task episodic memory, (2) *elastic weight consolidation* (EWC) [9], (3) *recursive gradient optimization* (RGO) [14], (4) *averaged gradient episodic memory* (A-GEM) [2], (5) ORTHOG-SUB [4], (6) *experience replay* [3] with *reservoir sampling*

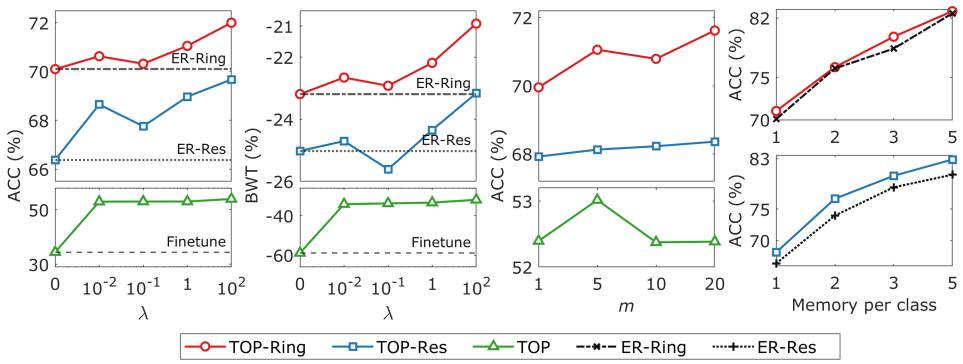

Figure 1: ACC and BWT scores as a function of the hyper-parameters of TOP-Ring, TOP-Res and TOP for P-MNIST dataset. $\lambda = 0$ for TOP-Ring, TOP-Res and TOP describe ER-Ring, ER-Res and *finetune*, respectively. TOP is our method without any memory buffer and uses a neural network graph constructed with the whole neural network.

[27] (ER-Res), (7) ER with *ring buffer* (ER-Ring) [15] and (8) *multitask* method that jointly learns the entire dataset in one training round and thus is not a continual learning strategy but serves as an upper bound reference for other methods. We employ our proposed topological regularization with reservoir sampling and ring buffer strategies, termed TOP-Res and TOP-Ring.

Continual Learning Evaluation We evaluate the algorithms based on two performance measures: average accuracy (ACC) and backward transfer (BWT) as proposed by Lopez-Paz & Ranzato [15]. Formally, ACC and BWT are defined as $\text{ACC} = \frac{1}{T}\sum_{j=1}^{T} R_{T,j}$ and $\text{BWT} = \frac{1}{T-1}\sum_{j=1}^{T-1} R_{T,j} - R_{j,j}$, where T is the total number of sequantial tasks, and $R_{i,j}$ is the accuracy of the model on the j^{th} task after learning the i^{th} task in sequence. We report ACC and BWT averaged over five different task sequences for each dataset.

Further experimental details are provided in Appendix D

Experimental Results Table 1 shows method performance on all four datasets. *Finetune* without any continual learning strategy produces lowest ACC and BWT performance, while the oracle *multitask* is trained across all tasks and sets the upper bound ACC performance for all datasets. Gradient-based RGO and ORTHOG-SUB rely on over-parameterization in a neural network to reduce interference between tasks. Given the long sequence of tasks and small network architecture in our experiments, it is likely that over-parameterization is insufficient for strong performance. Although ORTHOG-SUB achieves highest BWT, it also has the lowest ACC scores, in some cases worse than *Finetune*. ER-Res and ER-Ring achieve high ACC and BWT scores relative to the other baseline methods across all experiments. Our TOP-Res and TOP-Ring methods in turn demonstrate clear performance improvement over ER-Res and ER-Ring based on both ACC and BWT, suggesting that our topological continual learning strategy facilitates the consolidation of past-task knowledge beyond that provided by memory replay alone.

Figure 1 illustrates the impact of parameters λ and m of the proposed method on the ACC and BWT measures for the P-MNIST dataset. The two leftmost column plots depict the increase in ACC and BWT scores provided by TOP-Ring and TOP-Res relative to ER-Ring and ER-Res baselines for all λ, the topological regularizer weight. In addition, we demonstrate classification performance of TOP, our method without any episodic memory buffer. The results show that TOP outperforms *finetune* baseline and demonstrates clear past-task knowledge retention. TOP without any memory buffer performs better than the regularization-based EWC and gradient-based ORTHOG-SUB with memory buffers, as displayed in Table 1. The third column plot displays ACC scores as a function of m, the number of iterations between birth-death decomposition updates. We observe stable upward trend as m increases for TOP-Ring and TOP-Res. Less frequent birth-death decomposition update ($m = 10, 20$) improves performance and reduces run time. The last column displays ACC scores as a function of memories per class of 1, 2, 3 and 5. We observe that ACC scores increases as more past-task examples are stored in a tiny memory buffer. TOP-Ring and TOP-Res improves the performance over ER-Ring and ER-Res for all memory sizes. Finally, we explored ACC as a function of p/q of $9/1, 7/3, 5/5, 3/7$ and $1/9$, and found that the performance of both TOP-Ring and TOP-Res are not dependent on p/q.

References

[1] Danielle S Bassett, Nicholas F Wymbs, Mason A Porter, Peter J Mucha, Jean M Carlson, and Scott T Grafton. Dynamic reconfiguration of human brain networks during learning. *Proceedings of the National Academy of Sciences*, 108(18):7641–7646, 2011. doi: 10.1073/pnas.1018985108.

[2] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019.

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[4] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9900–9911. Curran Associates, Inc., 2020.

[5] Karolina Finc, Kamil Bonna, Xiaosong He, David M Lydon-Staley, Simone Kühn, Włodzisław Duch, and Danielle S Bassett. Dynamic reconfiguration of functional brain networks during working memory training. *Nature Communications*, 11(1):1–15, 2020.

[6] Robert Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[7] Corey B Hart and Simon F Giszter. A neural basis for motor primitives in the spinal cord. *Journal of Neuroscience*, 30(4):1322–1336, 2010. doi: 10.1523/JNEUROSCI.5894-08.2010.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[9] 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. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[12] Hyekyoung Lee, Hyejin Kang, Moo K. Chung, Bung-Nyun Kim, and Dong Soo Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE Transactions on Medical Imaging*, 31(12):2267–2277, 2012. doi: 10.1109/TMI.2012.2219590.

[13] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pp. 3925–3934. PMLR, 2019.

[14] Hao Liu and Huaping Liu. Continual learning with recursive gradient optimization. In *International Conference on Learning Representations*, 2022.

[15] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[16] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.

[17] James R Munkres. *Elements Of Algebraic Topology*. Avalon Publishing, 1996. ISBN 9780201627282.

[18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

[19] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

[22] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[23] Herbert A Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106, 1962.

[24] Tananun Songdechakraiwut, Li Shen, and Moo Chung. Topological learning and its application to multimodal brain network integration. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pp. 166–176, Cham, 2021. Springer International Publishing. ISBN 978-3-030-87196-3.

[25] Tananun Songdechakraiwut, Bryan M Krause, Matthew I Banks, Kirill V Nourski, and Barry D Van Veen. Fast topological clustering with Wasserstein distance. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=0kPL3xO4R5`.

[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[27] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. doi: 10.1145/3147.3165.

[28] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[29] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[30] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations*, 2020.

[31] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

A Proofs

A.1 Closed Form Wasserstein Barycenter

Let $\mathbb{D}(G^{(i)}) : d_1^{(i)} \leq \cdots \leq d_{|\mathbb{D}|}^{(i)}$ be the death set of network $G^{(i)}$. It follows that the l-th smallest death value of the barycenter $\mathcal{G}_{cycle}$ of the N networks is given by the weighted mean of all the l-th smallest death values of such networks, i.e., $\mathcal{G}_{cycle} : \overline{d}_1 \leq \cdots \leq \overline{d}_{|\mathbb{D}|}$, where

$$\overline{d}_l = \sum_{i=1}^{N} \nu_i d_l^{(i)} \Big/ \sum_{i=1}^{N} \nu_i.$$

Proof. Recall that the *Wasserstein barycenter for cycles* $\mathcal{G}_{cycle}$ is defined as the death set that minimizes the *weighted* sum of the Wasserstein distances for cycles, i.e.,

$$\begin{aligned}\mathcal{G}_{cycle} &= \arg\min_{\mathcal{G}} \sum_{i=1}^{N} \nu_i W_{cycle}^2(\mathcal{G}, G^{(i)}) \\ &= \arg\min_{\mathcal{G}} \sum_{i=1}^{N} \nu_i \sum_{\overline{d}_l \in \mathcal{G}} \left[\overline{d}_l - \phi_i^*(\overline{d}_l)\right]^2,\end{aligned}$$

where ν_i is a non-negative weight, and ϕ_i^* maps the l-th smallest death value in $\mathcal{G}$ to the l-th smallest death value in $\mathbb{D}(G^{(i)})$ for all l. The sum can be expanded as

$$\sum_{i=1}^{N} \nu_i \sum_{\overline{d}_l \in \mathcal{G}} \left[\overline{d}_l - \phi_i^*(\overline{d}_l)\right]^2 = \sum_{i=1}^{N} \nu_i \Big([\overline{d}_1 - d_1^{(i)}]^2 + \cdots + [\overline{d}_{|\mathbb{D}|} - d_{|\mathbb{D}|}^{(i)}]^2 \Big),$$

which is quadratic. By setting its derivative equal to zero, we find the minimum at $\overline{d}_l = \sum_{i=1}^{N} \nu_i d_l^{(i)} \Big/ \sum_{i=1}^{N} \nu_i$. □

A.2 Online Computation for Closed Form Wasserstein Barycenter

Given the Wasserstein barycenter from the previous task $\mathcal{G}_{cycle}^{(\tau-1)} = \{\overline{d}_l^{(\tau-1)}\}_l$. The barycenter for the current task $\mathcal{G}_{cycle}^{(\tau)} = \{\overline{d}_l^{(\tau)}\}_l$ can be computed as:

$$\overline{d}_l^{(\tau)} = \frac{p\overline{d}_l^{(\tau-1)} + q d_l^{(\tau)}}{p+q},$$

where $d_l^{(\tau)} \in \mathbb{D}(G^{(\tau)})$ and $p, q > 0$.

Proof. Let $\rho = q/(p+q)$. Then $1-\rho = p/(p+q)$. Recall $\mathbb{D}(G^{(j)}) = \{d_l^{(j)}\}_l$ be the death set of weights after training through the entire sequence up to task j.

When $\tau = 2$, we have the barycenter after training through the entire sequence up to the second task $\mathcal{G}_{cycle}^{(2)} = \{\overline{d}_l^{(2)}\}_l$, where $\overline{d}_l^{(2)} = (1-\rho) d_l^{(1)} + \rho d_l^{(2)}$. That is, $\mathcal{G}_{cycle}^{(2)}$ is the barycenter of $G^{(1)}$ and $G^{(2)}$ with weights associated with $G^{(1)}$ and $G^{(2)}$ as $(1-\rho)$ and ρ, respectively.

For $\tau > 2$, suppose $\mathcal{G}_{cycle}^{(\tau-1)} = \{\overline{d}_l^{(\tau-1)}\}_l$ is the barycenter after training through the sequence up to task $\tau - 1$. Then the online computation for barycenter for the next task results in $\{\overline{d}_l^{(\tau)}\}_l$, where

$$\begin{aligned}\overline{d}_l^{(\tau)} &= (1-\rho)\overline{d}_l^{(\tau-1)} + \rho d_l^{(\tau)} \\ &= (1-\rho) \sum_{i=1}^{\tau-1} \nu_i d_l^{(i)} \Big/ \sum_{i=1}^{\tau-1} \nu_i + \rho d_l^{(\tau)} \quad \nu_i > 0, \forall i.\end{aligned}$$

It follows that the sum of weights associated with $G^{(1)}, ..., G^{(\tau)}$

$$\begin{aligned}(1-\rho) \sum_{i=1}^{\tau-1} \nu_i \Big/ \sum_{i=1}^{\tau-1} \nu_i + \rho &= (1-\rho) \cdot 1 + \rho \\ &= 1,\end{aligned}$$

indicating that the l^{th} smallest death value $\overline{d}_l^{(\tau)}$ is given by the weighted mean of all the l^{th} smallest death values of networks $G^{(1)}, ..., G^{(\tau)}$. Thus, $\mathcal{G}_{cycle}^{(\tau)} = \{\overline{d}_l^{(\tau)}\}_l$ is the barycenter after additional training of the τ^{th} task. □

B Algorithm

Algorithm 1 Topological continual learning algorithm

procedure TOP($\mathbb{P}, \mathbb{W} = \{w\}, \lambda, p, q, \gamma$)
  $\mathbb{M} \leftarrow \{\}$ ▷ Allocate a tiny memory buffer
  for $B_{\mathbb{P}} \sim \mathbb{P}_1$ **do** ▷ Sample without replacement a mini-batch from 1^{st} dataset
    $g_{sgd} \leftarrow \nabla_w \mathcal{L}_{ERM}(B_{\mathbb{P}})$ ▷ Compute gradient using the mini-batch
    $w \leftarrow w - \gamma \cdot g_{sgd}$
    $\mathbb{M} \leftarrow$ mem_update($\mathbb{M}, B_{\mathbb{P}}$) ▷ Update memory
  end for
  $\mathcal{G}^{(1)} \leftarrow \mathbb{D}(\mathbb{W})$ ▷ Initial barycenter
  for $\tau \in \{2, .., T\}$ **do**
    $iter \leftarrow 0$
    for $B_{\mathbb{P}} \sim \mathbb{P}_\tau$ **do** ▷ Sample without replacement a mini-batch from τ^{th} dataset
      $B_{\mathbb{M}} \sim \mathbb{M}$ ▷ Sample a mini-batch from the buffer
      if $iter$ mod $m == 0$ **then** ▷ Every m iterations
        $\mathbb{D} \leftarrow$ bd_decomposition($\mathbb{W}$) ▷ Compute death set from current weight set
      end if
      $g_{sgd} \leftarrow \nabla_w \mathcal{L}_{ERM}(B_{\mathbb{P}} \cup B_{\mathbb{M}})$ ▷ Compute gradient using aggregated mini-batch
      if $w \in \mathbb{D}$ **then**
        $g_{top} \leftarrow w - \phi^*(w)$ ▷ Compute closed form gradient
        $w \leftarrow w - \gamma \cdot (g_{sgd} + \lambda \cdot g_{top})$
      else
        $w \leftarrow w - \gamma \cdot g_{sgd}$
      end if
      $iter \leftarrow iter + 1$
      $\mathbb{M} \leftarrow$ mem_update($\mathbb{M}, B_{\mathbb{P}}$)
    end for
    $\mathcal{G}^{(\tau)} \leftarrow \mathcal{G}^{(\tau-1)}(p, q)$ ▷ Update barycenter
  end for
  return $\mathbb{W}$ ▷ Return optimal weight set
end procedure

C Related Work

Approaches to continual learning in neural networks can broadly be categorized into three classes. 1) Regularization-based methods estimate importance of neural network parameters from previously learned tasks, and penalize changes to those important parameters during current tasks to mitigate forgetting [9, 21, 31]. Our topological regularization based method does not estimate importance for each individual parameter, but rather focuses on global network attributes. Thus, our method does not penalize individual parameters in the network, but rather penalizes cycle structure. 2) Memory-based methods attempt to overcome catastrophic forgetting by either storing examples from past tasks in a memory buffer for rehearsal [4, 15, 18, 19], or synthesizing past-task data from generative models for pseudo-rehearsal [22]. 3) Expansion-based methods allocate a different set of parameters within a neural network for each task [13, 28–30]. These methods may expand network size in an attempt to eliminate forgetting by design. In contrast, our method does not require network growth, but continually learns within a fixed network architecture.

Gradient-based methods, such as ORTHOG-SUB and RGO, can offer strong performance using over-parameterized neural networks [4, 14]. However, their performance degrades substantially in a more challenging scenario of limited network width and a long sequence of learning tasks, as in our experiments. Such scenarios are especially important when computational power is limited. Large neural networks not only demand more computation to optimize, but also demand a large memory footprint to store their parameter set. Our topology-based method does not require over-parameterized neural networks to be effective and thus has limited computational burden.

D Experimental Details

D.1 Datasets

We perform continual learning experiments on four datasets. (1) *Permuted MNIST* (P-MNIST) [9] is a variant of the MNIST dataset of handwriten digits [11] where each task is constructed by a fixed random permutation of image pixels in each original MNIST example. (2) *Rotated MNIST* (R-MNIST) [15] is another variant of the MNIST dataset where each task contains digits rotated by a fixed angle between 0 and 180 degrees. Both P-MNIST and R-MNIST contain 30 sequential tasks, each has 10,000 training examples and 10 classes. Each task in P-MNIST and R-MNIST is constructed with different permutations and rotations, respectively, resulting in different distributions among tasks. The other two datasets, (3) *Split CIFAR* and (4) *split miniImageNet*, are constructed by randomly splitting 100 classes in the original CIFAR-100 dataset [10] and ImageNet dataset [20, 26] into 20 learning tasks each with 5 classes.

D.2 Network architecture

The P-MNIST and R-MNIST datasets use a fully-connected neural network with two hidden layers each with 128 neurons. The split CIFAR and split miniImageNet datasets use a downsized version of ResNet18 [8] with eight times fewer feature maps across all layers, similar to [15]. For P-MNIST and R-MNIST, we evaluate classification performance in a *single-head* setting where all tasks share the final classifier layer in the fully-connected network, and thus the inference is performed without task identifiers. For the split CIFAR and split miniImageNet datasets, the classification performance is evaluated in a *multi-head* setting where each task has a separate classifier in the reduced ResNet18.

D.3 Method comparison

We evaluate our method performance in relative to eight baseline approaches that learn a sequence of tasks in a fixed-size network architecture. (1) *Finetune* is a model trained sequentially without any regularization and past-task episodic memory. (2) *Elastic weight consolidation* (EWC) [9] is a regularization-based method that penalizes changes in parameters that were important for the previous tasks using the Fisher information. (3) *Recursive gradient optimization* (RGO) [14] combines use of gradient direction modification and task-specific random rearrangement to network layers to mitigate forgetting between tasks. (4) *Averaged gradient episodic memory* (A-GEM) [2] and (5) ORTHOG-SUB [4] methods store past-task examples in an episodic memory buffer used to project gradient updates to avoid interference with previous tasks. (6) *Experience replay* with *reservoir sampling* (ER-Res) and (7) ER with *ring buffer* (ER-Ring) mitigate forgetting by directly computing gradient based on aggregated examples from new tasks and a memory buffer [3]. The ring buffer [15] allocates equally sized memory for each class using FIFO scheduling, while reservoir sampling [27] maintains a memory buffer by randomly drawing out already stored samples when the buffer is full. (8) *Multitask* jointly learns the entire dataset in one training round, and thus is not a continual learning strategy but served as an upper bound reference for other methods.

In our topology-based method we regularize different aspects of the two types of neural networks employed. In the fully-connected network applied to P-MNIST and R-MNIST we apply subgraph regularization based on two separate bipartite graphs: one consisting of neurons and weights from the two hidden layers, the other consisting of neurons and weights from the last hidden layer and the output layer. For ResNet18, we use one bipartite graph comprising neurons and weights from the pooling layer and the fully-connected output layer. Bias neurons are not included in our regularization. We employ topological regularization with reservoir sampling and ring buffer strategies, termed TOP-Res and TOP-Ring. Universally, m is set to 5, p to 9, and q to 1 in our experiment.

D.4 Training protocol

We follow the training protocol of Chaudhry et al. [4] as follows. The training is done universally through plain stochastic gradient descent with batch size of 10. All training examples in the datasets are observed only once, except for past-task examples stored in an episodic memory buffer, which can be replayed multiple times. We consider a tiny episodic memory with the size of one *example per class* for all memory-based methods. For example, there are 100 classes in split CIFAR, and thus the memory buffer stores up to 100 past-task examples. The size of past-task batch sampled from

the episodic memory is also universally set to 10, the gradient descent batch size. Hyper-parameter tuning is performed on the first three tasks in each dataset. A separate test set not used in the training tasks is used for performance evaluation.

D.5 Implementation of Candidate Methods

For baseline methods, we used existing implementation codes from authors' publications and publicly available repository websites. Codes for EWC [9], A-GEM [2], ORTHOG-SUB [4], and ER-Res/ER-Ring [3] are available at `https://github.com/arslan-chaudhry/orthog_subspace` under the MIT License. Code for RGO [14] is available at `https://openreview.net/forum?id=7YDLgf9_zgm`.

D.6 Hyperparameters Tuning

Grid search across different hypeparameter values is used to choose a set of optimal hyperparameters for all candidate methods. We consider gradient descent learning rates in $\{0.003, 0.01, 0.03, 0.1, 0.3, 1.0\}$ following Chaudhry et al. [4], Liu & Liu [14]. Other additional hyperparameters of EWC, A-GEM, ORTHOG-SUB, and ER-Res/ER-Ring follow Chaudhry et al. [4], while those of RGO follow the authors' official implementation [14]. Specifically, below we report grids for the candidate methods with the optimal hyperparameter settings for different benchmarks given in parenthesis.

1. Finetune
   - Learning rate: 0.003, 0.01, **0.03 (miniImageNet)**, **0.1 (P-MNIST, R-MNIST)**, **0.3 (CIFAR)**, 1
2. EWC
   - Learning rate: 0.003, 0.01, **0.03 (CIFAR, miniImageNet)**, **0.1 (P-MNIST, R-MNIST)**, 0.3, 1
   - Regularization: 0.01, 0.1, 1, **10 (P-MNIST, R-MNIST, CIFAR, miniImageNet)**, 100, 1000
3. RGO
   - Learning rate: 0.003, 0.01, **0.03 (CIFAR, miniImageNet), 0.1 (P-MNIST, R-MNIST)**, 0.3, 1
4. A-GEM
   - Learning rate: 0.003, 0.01, **0.03 (miniImageNet)**, **0.1 (P-MNIST, R-MNIST, CIFAR)**, 0.3, 1
5. ORTHOG-SUB
   - Learning rate: 0.003, 0.01, **0.03 (R-MNIST)**, **0.1 (P-MNIST, CIFAR, miniImageNet)**, 0.3, 1
6. ER-Res
   - Learning rate: 0.003, 0.01, **0.03 (miniImageNet)**, **0.1 (R-MNIST, CIFAR)**, **0.3 (P-MNIST)**, 1
7. ER-Ring
   - Learning rate: 0.003, 0.01, **0.03 (miniImageNet)**, **0.1 (P-MNIST, R-MNIST, CIFAR)**, 0.3, 1
8. Multitask
   - Learning rate: 0.003, 0.01, **0.03 (CIFAR, miniImageNet)**, **0.1 (P-MNIST, R-MNIST)**, 0.3, 1
9. TOP-Res
   - Learning rate: 0.003, 0.01, 0.03, **0.1 (CIFAR, miniImageNet)**, **0.3 (P-MNIST, R-MNIST)**, 1
   - Regularization: **0.01 (CIFAR, miniImageNet)**, 0.1, **1 (P-MNIST, R-MNIST)**, 10
10. TOP-Ring
    - Learning rate: 0.003, 0.01, 0.03, **0.1 (P-MNIST, R-MNIST, CIFAR, miniImageNet)**, 0.3, 1
    - Regularization: **0.01 (R-MNIST, miniImageNet)**, 0.1, **1 (P-MNIST, CIFAR)**, 10

D.7 Computational Resources

$1\times$ NVIDIA GeForce GTX 1080