How Efficient Are Today's Continual Learning Algorithms?

Md Yousuf Harun[1] Jhair Gallardo[1] Tyler L. Hayes[1*] Christopher Kanan[2]
Rochester Institute of Technology[1], University of Rochester[2]
{mh1023, gg4099, tlh6792}@rit.edu, ckanan@cs.rochester.edu

Abstract

Supervised Continual learning involves updating a deep neural network (DNN) from an ever-growing stream of labeled data. While most work has focused on overcoming catastrophic forgetting, one of the major motivations behind continual learning is being able to efficiently update a network with new information, rather than retraining from scratch on the training dataset as it grows over time. Despite recent continual learning methods largely solving the catastrophic forgetting problem, there has been little attention paid to the efficiency of these algorithms. Here, we study recent methods for incremental class learning and illustrate that many are highly inefficient in terms of compute, memory, and storage. Some methods even require more compute than training from scratch! We argue that for continual learning to have real-world applicability, the research community cannot ignore the resources used by these algorithms. There is more to continual learning than mitigating catastrophic forgetting.

1. Introduction

In many real-world scenarios, the dataset used to train a deep neural network (DNN) grows over time. In industry settings, this is typically handled by periodically re-training the DNN from scratch after the dataset has grown (i.e., offline training); however, this is highly sub-optimal. In contrast, continual learning (CL) systems have the ability to update on novel inputs over time [18]. CL has the potential to yield significant computational benefits over periodically re-training from scratch. The vast majority of CL research has focused on solving catastrophic forgetting, which occurs when updating a DNN only on new data with non-CL methods. Thus, many CL algorithms have been designed for *class incremental learning*, since it is a problem where severe catastrophic forgetting occurs [12]. In this setting, the learner incrementally learns batches of mutually exclusive subsets of classes, rather than learning them all at once as in the offline setting. While state-of-the-art algorithms, e.g., DyTox [4] and DER [26], excel at class incremental learning under a limited replay memory budget, they ignore other critical factors such as model size, compute, total memory usage, data efficiency, and training time. We argue that these factors cannot be ignored. Here, we study the efficiency of existing CL methods for class incremental learning on ImageNet, which reveals that many of them are impractical for real-world CL applications.

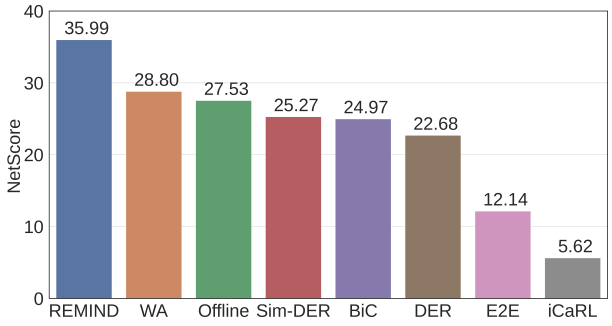


Figure 1. We use NetScore to evaluate CL methods based on their accuracy, parameter count, amount of memory used, and number of backpropagation updates (compute).

Most CL methods for incremental class learning on ImageNet perform a very large number of updates to the DNN, where we define an update as backpropagation on a single input. In opposition to one of the major goals of CL, computational efficiency, some models have become *more* computationally expensive than an offline model trained on all data. For example, on ImageNet, DER was used to incrementally train a ResNet18 model, but this required more updates (213.17 Million) than training ResNet18 offline (115.31 Million). They also use many more parameters than the offline DNN, e.g., DER [26] combines multiple feature extraction networks (i.e., one network per task for each 100 class increment) to learn a unified classifier that expands the total number of parameters to $10\times$ the offline DNN.

Another shortcoming of existing systems is that they do not fully account for the amount of memory or storage used during incremental training. State-of-the-art systems for in-

*Now at NAVER LABS Europe.

cremental class learning on ImageNet use replay to mitigate catastrophic forgetting, where past data, which is stored in a limited buffer, is mixed (i.e., replayed) with new incoming data. However, many of them, including iCaRL [20], End-to-End [1], BiC [25], WA [28], and DER [26], also keep new data temporarily in memory (or disk storage) during each batch, which is unaccounted for. Each incremental learning batch typically consists of around 100,000 images (100 classes). While smaller batches could be used to reduce memory overhead, it has been shown that many methods suffer from severe catastrophic forgetting unless large batches are used [7]. Additional memory is also used for the DNN parameters.

Ideally, a model should adapt to a growing training dataset without increasing the computational or memory budget. However, most CL methods lack this ability. While memory and storage may not be a concern for training in industry settings, it does matter for on-device learning applications, including autonomous robots, smart appliances, mobile phones, virtual/ augmented reality (VR/AR) headsets, and other wearable devices, which typically have very limited storage. For on-device learning, a CL system must adapt to new information quickly despite limited computation and memory.

Our contributions can be summarized as follows:

1. We use the NetScore metric [8] for evaluating state-of-the-art methods for incremental class learning on ImageNet-1K to summarize their properties in terms of accuracy, memory, parameters, and compute needed for training.

2. We find that most methods are grossly inefficient. Multiple methods use more compute than an offline learner, defeating one of the major reasons for studying CL.

3. We discuss the criteria needed for CL to be useful for real-world applications, and we call for the CL research community to focus on issues beyond catastrophic forgetting.

2. Methodology

2.1. Evaluation Criteria

We evaluate a model's efficiency in terms of four criteria: accuracy, model size, computational overhead (updates), and memory overhead. We adopt a modified variant of the *NetScore* metric from [8] that combines these criteria. For a model $\mathcal{G}$, the *NetScore* metric $\Omega(\mathcal{G})$ is defined as:

$$\Omega(\mathcal{G}) = s \log\left(\frac{a(\mathcal{G})^{\alpha}}{p(\mathcal{G})^{\beta} u(\mathcal{G})^{\gamma} m(\mathcal{G})^{\zeta}}\right) , \quad (1)$$

where $a(\mathcal{G})$ is the final top-5 accuracy, $p(\mathcal{G})$ is number of parameters, $u(\mathcal{G})$ is the number of backpropagation updates, and $m(\mathcal{G})$ is the total memory usage. The coefficients α, β, γ, and ζ control the contribution of each factor. Following [8], we set $\alpha = 2$ and $s = 20$. We set $\beta = \gamma = \zeta = 0.125$ to avoid having a negative $\Omega(\mathcal{G})$.

2.2. Algorithms Studied

We evaluate the following CL methods, which have all been shown to perform well on ImageNet-1K:

- iCaRL [20] learns representations using a distillation loss and makes predictions using a nearest class mean classifier in feature space. During each incremental learning step, iCaRL trains the entire network.

- End-to-End [1] upgrades iCaRL using augmentations and fine-tunes the CNN's output layer on a balanced set rather than using a nearest class mean classifier. The augmentations applied in End-to-End include brightness enhancement, contrast normalization, random crops, and mirror flips.

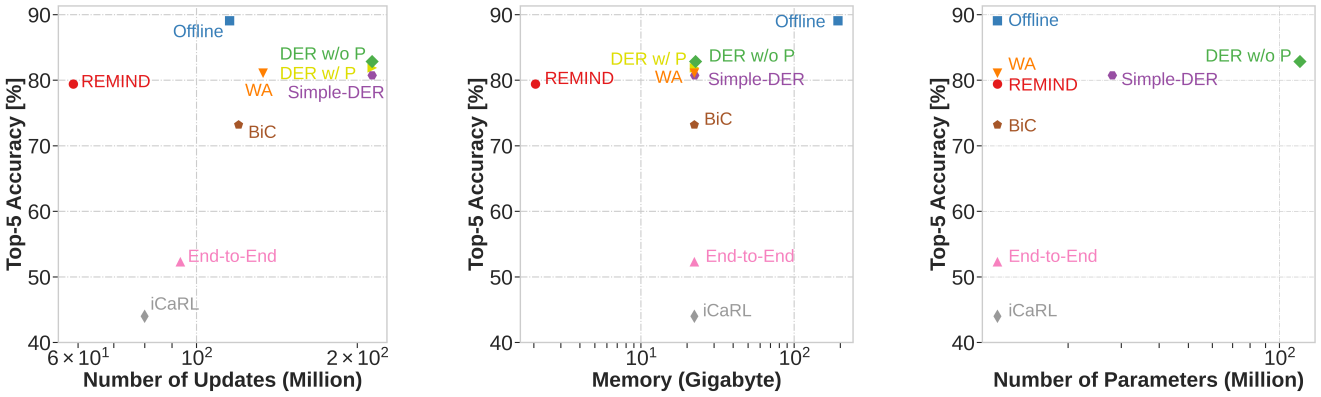
- BiC [25] upgrades iCaRL by re-adjusting the logits of new classes by training a linear model on a validation set. For this, two bias correction parameters on the final layer are optimized.

- WA [28] upgrades iCaRL by aligning the norms of new class parameters to those of old class parameters.

- DER [26] augments previously learned representations with representations for the new classes. It has a separate feature extractor per incremental batch and applies pruning to reduce number of parameters. It also employs an auxiliary classifier to discriminate between past and present observations. We compare three variants of DER: DER without pruning, DER with pruning, and Simple-DER [13].

- REMIND [7] uses compressed feature replay where mid-level CNN features are quantized to reduce memory overhead. In contrast, the other methods use replay of stored images. REMIND freezes earlier CNN layers and trains the remaining layers on reconstructed features. To do this, REMIND uses the first batch to initialize the CNN.

For our analysis, except REMIND, we use the numbers as reported in each method's paper on ImageNet and the settings they describe in their work. Since the data ordering used in the REMIND paper [7] is different, we implement REMIND with the same data ordering used by the compared methods in Table 1. For replay, REMIND uses



(a) Accuracy vs. Backprop Updates (b) Accuracy vs. Memory (c) Accuracy vs. Parameter Count

Figure 2. Comparison of methods in terms of top-5 accuracy and backprop updates, memory, and parameter count. In all sub-figures, REMIND uses the fewest resources and provides competitive accuracy. In contrast, DER w/o P uses the most resources, but achieves the highest accuracy compared to other CL methods. In (a), several methods such as BiC, WA, DER w/o P, DER w/ P, and Simple-DER require more computation than offline ResNet18. In (c), DER w/ P is not included since it does not report total number of parameters.

2 GB of storage when storing compressed mid-level features, whereas all the remaining methods use 3 GB of storage that corresponds to 20000 raw images (224×224 uint8). Except for REMIND, these methods also store the current data corresponding to 100 classes during each learning step. Hence, their total memory usage is much more than the replay memory budget (at least $7\times$ more). All methods use ResNet18 [9].

3. Results

3.1. Evaluation on ImageNet-1K

Following the common CL practice for ImageNet-1K [21], each method was incrementally trained on 10 batches, where each batch has images from 100 classes that are only seen in that batch. The first batch serves as the base initialization phase, and the subsequent 9 batches serve as the CL phase. We measure efficiency only for the CL phase. For all CL methods and the offline ResNet18 model, we compare the top-5 accuracy, number of parameters, amount of memory used, number of backpropagation updates used for training, and NetScore.

Results are summarized in Table 1, Fig. 1, and Fig. 2. Simple-DER and End-to-End are abbreviated as Sim-DER and E2E respectively in Fig. 1. In terms of NetScore, both REMIND and WA outperform the offline model. REMIND uses $10\times$ less memory and the least amount of compute. Shockingly, several methods use roughly as much or more compute than the offline model, including BiC, WA, and DER variants. While DER without pruning achieves the closest accuracy to the offline model, this comes at a cost of requiring $10\times$ more parameters and almost twice the compute as the offline ResNet18. Except for REMIND, all other systems require storing the entire batch in a storage.

Table 1. Comparison of CL methods evaluated on ImageNet-1K. #P indicates total number of parameters in Millions. Ω refers to NetScore which is calculated based on top-5 final accuracy (%), number of parameters, amount of memory and compute (updates). Acc. is top-5 final accuracy (%). Mem. is the total memory/storage used in Gigabytes. Comp. denotes compute in terms of total number of backprop updates in Millions. DER without pruning and DER with pruning are abbreviated as DER w/o P and DER w/ P, respectively. Total #P used by DER w/ P is unknown. All methods except End-to-End and Simple-DER follow same data ordering from [26]. Best values are indicated in bold.

Methods	#P	Acc.	Mem.	Comp.	Ω
Offline	11.68	89.08	192.89	115.31	27.53
iCaRL	**11.68**	44.00	22.32	79.94	5.62
End-to-End	**11.68**	52.29	22.32	93.26	12.14
BiC	**11.68**	73.20	22.32	119.91	24.97
WA	**11.68**	81.10	22.32	133.23	28.80
Simple-DER	28.00	80.76	22.39	213.17	25.27
DER w/o P	116.89	**82.86**	22.74	213.17	22.68
DER w/ P	—	81.89	22.32	213.17	—
REMIND	**11.68**	79.43	**2.05**	**58.78**	**35.99**

4. Discussion

In this paper, we have argued that there is more to CL than catastrophic forgetting, and that the research community is ignoring critical factors e.g., model size, computational overhead, training time that are essential to address for CL to have real-world impact. We illustrated that multiple recent methods use as much or more compute than an offline learner, which is in opposition to one of the major goals of CL: efficient learning.

We believe CL can help to reduce the economic and environmental costs of deep learning, but this will only oc-

cur if CL algorithms are computationally efficient. Large DNNs require massive amounts of electricity to train, which greatly contribute to the growing amount of carbon emission worldwide. For example, the 176-billion parameter language model BLOOM has been estimated to emit 50.5 tonnes carbon in its life cycle [14]. In [19], training GPT-3 was estimated to require as much energy as the annual consumption of 120 U.S. homes. CL models can help address this issue and offer additional functionality, but only if they are computationally cheaper than periodic retraining of offline models as the dataset grows.

Focusing on incremental class learning has led to many systems to be designed to only handle this scenario, which we consider an extreme edge case. They have bespoke features for this problem that are only appropriate if each batch contains unique classes, and their algorithms without major modifications break when this assumption does not hold. This assumption is invalid for almost all real-world applications, where we cannot make any assumptions about the distribution of new data. Ideally, a good CL system should learn from data in any order, including independent and identically distributed (iid) data or data that has temporal correlations. Of the methods we compared, only REMIND [7] has been designed to handle data with arbitrary orderings and to allow classes to be revisited.

We focused on CL methods that have been demonstrated to perform well on ImageNet-1K because it is much more closely aligned with real-world applications for DNNs than many of the tasks studied in the CL community. CL on ImageNet is just supervised learning, except we are progressively updating the DNN. In contrast, many CL systems are only evaluated on small-scale problems where they make additional assumptions beyond simply that the training data arrives incrementally. For example, many works still focus on tasks such as permuted MNIST and split-CIFAR100 [2, 3, 11, 15, 17, 23, 24, 27], which do not align with real-world supervised learning tasks where CL could be a drop-in alternative to periodically retraining from scratch. Many recent CL systems also assume task labels are available during training and/or evaluation [5, 6, 10, 22], which is also not aligned with typical real-world applications of supervised learning.

For CL to have real-world impact, we make the following recommendations to the CL research community:

1. CL researchers need to focus on more than catastrophic forgetting. We argue that computational efficiency is the most critical factor, but other factors such as the number of additional parameters or memory used need to be taken into account. A CL system that uses more computation than an offline learner is hard to justify.

2. Researchers should aim toward scaling up dataset sizes and the scope of the CL problems studied. While other areas of machine learning, e.g., generative methods, rapidly advanced such that reporting results on only toy datasets is unacceptable, much of the CL research community continues to focus on problems where CL is unnecessary.

3. Papers should clearly justify the CL paradigm they are studying and the limitations of their algorithm. Systems should be aligned toward real-world CL applications, unless trying to test a specific hypothesis regarding CL. For example, to enhance transparency in interpretation of experimental results a metric such as CLEVA [16] could be used.

4. Papers proposing new CL algorithms should report what an offline learner achieves and they should report the amount of compute needed to train the CL system relative to an offline learner.

5. CL systems should be tested robustly across multiple orders and should not be designed only for extreme edge-cases, e.g., only being capable of incremental class learning. For real-world CL applications, we typically cannot make any strong assumptions about the distribution of the training data stream.

6. Ideally, CL systems should be designed to be updated online, where if batches are used the system's robustness is assessed across multiple batch sizes. Many systems require extremely large batches to learn [4, 26], which may be acceptable for some applications but makes the system unacceptable for others, e.g., on-device learning.

5. Conclusion

In this paper, we studied the efficiency of recent CL methods in terms of compute, memory, and accuracy. We found that some state-of-the-art techniques use more compute than the equivalent offline learner, which for many industrial applications makes them irrelevant even if they avoid catastrophic forgetting. We urge the research community to take factors beyond catastrophic forgetting into consideration. Systems must be tested to work across data orderings, in addition to being efficient in terms of compute, and for many applications, e.g., on-device learning, must also take into account the number of parameters and memory/storage that they use for training. We believe CL can play a critical role in reducing the global energy expenditure resulting from training DNNs, but this requires the community to align their work with the needs of industry.

Acknowledgements. This work was supported in part by NSF awards #1909696 and #2047556.

References

[1] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018. 2

[2] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. 4

[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

[4] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 1, 4

[5] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 4

[6] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019. 4

[7] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. REMIND your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020. 2, 4

[8] Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. In *CoLLAs*, 2022. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[10] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[11] Prakhar Kaushik, Alex Gain, Adam Kortylewski, and Alan Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *arXiv preprint arXiv:2102.11343*, 2021. 4

[12] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 1

[13] Zhuoyun Li, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng. Preserving earlier knowledge in continual learning with the help of all previous feature extractors. *arXiv preprint arXiv:2104.13614*, 2021. 2

[14] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022. 4

[15] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020. 4

[16] Martin Mundt, Steven Lang, Quentin Delfosse, and Kristian Kersting. CLEVA-compass: A continual learning evaluation assessment compass to promote research transparency and comparability. In *International Conference on Learning Representations*, 2022. 4

[17] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020. 4

[18] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 1

[19] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021. 4

[20] 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*, pages 2001–2010, 2017. 2

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 3

[22] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018. 4

[23] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*, 2019. 4

[24] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020. 4

[25] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 2

[26] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 1, 2, 3, 4

[27] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International*

Conference on Machine Learning, pages 3987–3995. PMLR, 2017. 4

[28] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020. 2