On Efficient Lifelong Learning with A-GEM

Arslan Chaudhry[1], MarcAurelio Ranzato[2], Marcus Rohrbach[2], Mohamed Elhoseiny[2]
[1]University of Oxford, [2]Facebook AI Research
arslan.chaudhry@eng.ox.ac.uk, {ranzato,mrf,elhoseiny}@fb.com

Abstract

In lifelong learning, the learner is presented with a sequence of tasks, incrementally building a data-driven prior which may be leveraged to speed up learning of a new task. In this work, we investigate the *efficiency* of current lifelong approaches, in terms of sample complexity, computational and memory cost. Towards this end, we first introduce a new and a more realistic evaluation protocol, whereby learners observe each example only once and hyper-parameter selection is done on a small and disjoint set of tasks, which is not used for the actual learning experience and evaluation. Second, we introduce a new metric measuring how quickly a learner acquires a new skill. Third, we propose an improved version of GEM [10], dubbed Averaged GEM (A-GEM), which enjoys the same or even better performance as GEM, while being almost as computationally and memory efficient as EWC [5] and other regularization-based methods. Finally, we show that all algorithms including A-GEM can learn even more quickly if they are provided with task descriptors specifying the classification tasks under consideration. Our experiments on several standard lifelong learning benchmarks demonstrate that A-GEM has the best trade-off between accuracy and efficiency.[1]

1 Introduction

Arguably, a good set of desiderata for lifelong learning (LLL) are: 1) **Sample efficiency**: the learner should need only a handful of samples provided one-by-one in a single pass for each task, 2) **Memory efficiency**: the memory and size of the architecture should not grow with the number of tasks, 3) **Time efficiency**: the learner should not engage in computations which cannot be easily performed in real time. While existing LLL methods [1, 5, 10–12] perform well in various settings, they fail on at least one of these essential desiderata, as shown in Fig 1 and described in Sec. 6.1.

In this work, we propose an evaluation methodology and an algorithm that better match our desiderata, namely learning efficiently from a stream of tasks. First, we propose a new learning paradigm, whereby the learner performs cross validation on a set of tasks which is disjoint from the set of tasks actually used for evaluation (Sec. 2). In this setting, the learner will have to learn and will be tested on an entirely new sequence of tasks and it will perform just a single pass over this data stream. Second, we build upon GEM [10], an algorithm which leverages a small episodic memory to perform well in a single pass setting, and propose a small change to the loss function which not only improves GEM performance, but it also makes GEM orders of magnitude faster at training time; we dub this variant of GEM, A-GEM (Sec. 4). Third, we explore the use of compositional task descriptors in order to improve the few-shot learning performance within LLL showing that with this additional information the learner can pick up new skills more quickly (Sec. 5). Fourth, we introduce a new metric to measure the speed of learning, which is useful to quantify the ability of a learning algorithm to learn a new task (Sec. 3). And finally, we demonstrate the better trade-off between average accuracy and computational/memory cost of A-GEM on a variety of benchmarks and against several baselines (Sec. 6).

[1]**Under review as a conference paper at ICLR 2019.**

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Learning Protocol

We consider two streams of tasks, described by the following ordered sequences of datasets $\mathcal{D}^{CV} = \{\mathcal{D}_1, \cdots, \mathcal{D}_{T^{CV}}\}$ and $\mathcal{D}^{EV} = \{\mathcal{D}_{T^{CV}+1}, \cdots, \mathcal{D}_T\}$, where $\mathcal{D}_k = \{(\mathbf{x}_i^k, t_i^k, y_i^k)_{i=1}^{n_k}\}$ is the dataset of the k-th task, $T^{CV} < T$ (in all our experiments $T^{CV} = 3$ while $T = 20$), and we assume that all datasets are drawn from the same distribution over tasks. To avoid cluttering of the notation, we let the context specify whether $\mathcal{D}_k$ refers to the training or test set of the k-th dataset.

$\mathcal{D}^{CV}$ is the stream of datasets which will be used during cross-validation; $\mathcal{D}^{CV}$ allows the learner to replay all samples multiple times for the purposes of model hyper-parameter selection. Instead, $\mathcal{D}^{EV}$ is the actual dataset used for final training and evaluation on the test set; the learner will observe training examples from $\mathcal{D}^{EV}$ once and only once, and all metrics will be reported on the test sets of $\mathcal{D}^{EV}$.

Each example in these dataset consists of a triplet defined by an input ($\mathbf{x}^k \in \mathcal{X}$), task descriptor ($t^k \in \mathcal{T}$, see Sec. 5 for examples) and a target vector ($y^k \in \mathbf{y}^k$), where $\mathbf{y}^k$ is the set of labels for task k and $\mathbf{y}^k \subset \mathcal{Y}$. While observing the data, the goal is to learn a predictor $f_\theta : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^P$ (a neural network in our case), that can map a test pair $(\mathbf{x}, t)$ to a target y.

3 Metrics

In addition to Average Accuracy (A_T) and Forgetting Measure (F_T) [2] after seeing T tasks, we define a new metric, the **Learning Curve Area (LCA** $\in [0, 1]$), that captures how quickly a model learns.

Let $a_{k,i,j} \in [0, 1]$ be the accuracy evaluated on the test set of task j, after the model has been trained with the i-th mini-batch of task k. Assuming the first learning task in the continuum is indexed by 1 (it will be $T^{CV} + 1$ for $\mathcal{D}^{EV}$) and the last one by T (it will be T^{CV} for $\mathcal{D}^{CV}$), let us first define an average b-shot performance (where b is the mini-batch number) after the model has been trained for all the T tasks as: $Z_b = \frac{1}{T}\sum_{k=1}^{T} a_{k,b,k}$. LCA at β is the area of the convergence curve Z_b with $b \in [0, \beta]$:

$$\text{LCA}_\beta = \frac{1}{\beta + 1}\int_0^\beta Z_b db = \frac{1}{\beta + 1}\sum_{b=0}^{\beta} Z_b \tag{1}$$

LCA has an intuitive interpretation. LCA_0 is the average 0-shot performance, the same as forward transfer in Lopez-Paz & Ranzato [10]. LCA_β is the area under the Z_b curve, which is high if the 0-shot performance is good and if the learner learns quickly. In particular, there could be two models with the same A_T, but very different LCA_β because one learns much faster than the other while they both eventually obtain the same final accuracy. Since we are interested in models that learn quickly, we will consider small values of β.

4 Averaged Gradient Episodic Memory (A-GEM)

In this work we build upon GEM [10], as it has proven to work well in a single pass setting. The inner optimization loop of GEM becomes prohibitive in terms of memory and compute time when the size of the episodic memory $\mathcal{M}$ and the number of tasks is large. To alleviate this computational burden, we propose a more efficient version of GEM, called Averaged GEM (A-GEM).

Whereas GEM ensures that at every training step the loss of each *individual* previous tasks, approximated by the samples in episodic memory, does not increase, A-GEM tries to ensure that at every training step the *average* episodic memory loss over the previous tasks does not increase. Formally, while learning task t, the objective of A-GEM is:

$$\text{minimize}_\theta \quad \ell(f_\theta, \mathcal{D}_t) \quad \text{s.t.} \quad \ell(f_\theta, \mathcal{M}) \leq \ell(f_\theta^{t-1}, \mathcal{M}) \qquad \text{where } \mathcal{M} = \cup_{k<t}\mathcal{M}_k \tag{2}$$

The corresponding optimization problem reduces to:

$$\text{minimize}_{\tilde{g}} \quad \frac{1}{2}||g - \tilde{g}||_2^2 \quad \text{s.t.} \quad \tilde{g}^\top g_{ref} \geq 0 \tag{3}$$

where g_{ref} is a gradient computed using a batch randomly sampled from the episodic memory, $(\mathbf{x}_{ref}, y_{ref}) \sim \mathcal{M}$, of all the past tasks. In other words, A-GEM replaces the $t - 1$ constraints of GEM with a single constraint, where g_{ref} is the average of the gradients from the previous tasks computed from a random subset of the episodic memory.

The constrained optimization problem of Eq. 3 can now be solved very quickly; when the gradient g violates the constraint, it is projected via: $\tilde{g} = g - \frac{g^\top g_{ref}}{g_{ref}^\top g_{ref}} g_{ref}$. This makes A-GEM not only memory efficient, as it does not need to store the parameter gradient vector of each previous task, but also orders of magnitude faster than GEM because 1) it only needs to compute the gradient on an additional minibatch, 2) it does not need to solve any QP but just an inner product, and 3) it will incur in less violations particularly when the number of tasks is large.

5 Joint Embedding Model Using Compositional Task Descriptors

Borrowing ideas from literature in few-shot learning [8, 16, 3, 14], we propose the use of a *joint embedding* model. Let $\mathbf{x}^k \in \mathcal{X}$ be the input (e.g., an image), t^k be the task descriptor in the form of a matrix of size $C_k \times A$, where C_k is the number of classes in the k-th task and A is the total number of attributes for each class in the dataset. The joint embedding model consists of a feature extraction module, $\phi_\theta : \mathbf{x}^k \rightarrow \phi_\theta(\mathbf{x}^k)$, where $\phi_\theta(\mathbf{x}^k) \in \mathbb{R}^D$, and a task embedding module, $\psi_\omega : t^k \rightarrow \psi_\omega(t^k)$, where $\psi_\omega(t^k) \in \mathbb{R}^{C_k \times D}$. During training, the parameters θ and ω are learned by minimizing the cross-entropy loss (with the additional constraint of Eq. 3):

$$\ell_k(\theta, \omega) = \frac{1}{N} \sum_{i=1}^{N} -\log(p(y_i^k | \mathbf{x}_i^k, t^k; \theta, \omega)) \quad \text{where} \quad p(c|\mathbf{x}_i^k, t^k; \theta, \omega) = \frac{\exp([\phi_\theta(\mathbf{x}_i^k)\psi_\omega(t^k)^\top]_c)}{\sum_j \exp([\phi_\theta(\mathbf{x}_i^k)\psi_\omega(t^k)^\top]_j)} \tag{4}$$

where $c = y_i^k$ is the target class of the i-th example of task k ($\mathbf{x}_i^k, t^k, y_i^k$) and $[a]_i$ denotes the i-th element of the vector a. Note that the architecture and loss functions are general, and apply not only to A-GEM but also to any other LLL model (e.g., regularization based approaches). See Sec. 6 for the actual choice of parameterization of these functions.

6 Experiments

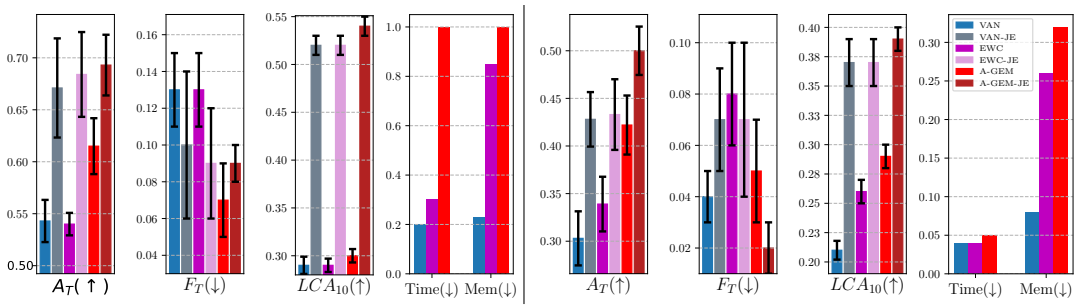
We consider four dataset streams; **Permuted MNIST** [5] which is a variant of MNIST [9] where each task has a certain random permutation of the input pixels applied to all the images of that task; **Split CIFAR** [15] consists of splitting the original CIFAR-100 dataset [6] into 20 disjoint subsets, where each subset is constructed by randomly sampling 5 classes *without* replacement from a total of 100 classes; **Split CUB** which is an incremental version of the CUB dataset [13] of 200 bird categories split into 20 disjoint subsets of classes similar to CIFAR; and **Split AWA** which is an incremental version of the AWA dataset [7] of 50 animal categories, where each task is constructed by sampling 5 classes *with* replacement from the original of 50 classes. Although a class may appear in different tasks in Split AWA, the training data of a each class is split into disjoint sets, so that no training sample is seen more than once. On Permuted MNIST and Split CIFAR we provide integer task descriptors. On Split CUB and Split AWA, we assemble together the attributes of the classes belonging to the current task to form a task descriptor.

In terms of architectures, we use a fully-connected network with two hidden layers of 256 ReLU units each for Permuted MNIST, a reduced ResNet18 for Split CIFAR [10], and a standard ResNet18 [4] for Split CUB and Split AWA. For a given dataset stream, all models use the same architecture and all models are optimized via stochastic gradient descent with mini-batch size equal to 10. We refer to the joint-embedding model version of these models by appending the suffix '-JE' to the method name.

6.1 Results

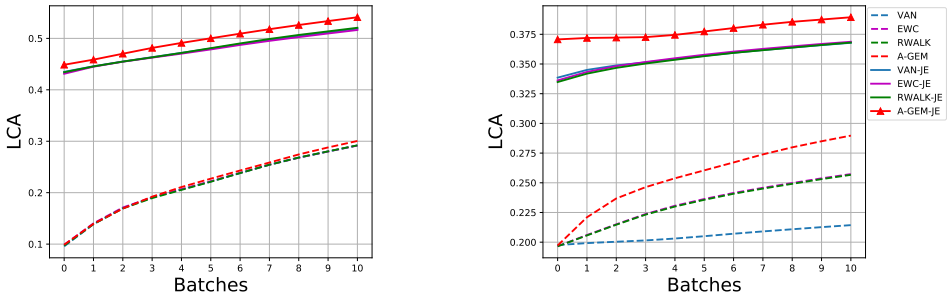
On Split CIFAR, GEM and A-GEM have the average accuracy of 61.2 and 62.9, and forgetting of 0.06 and 0.07. This shows that A-GEM and GEM perform comparably in terms of average accuracy and forgetting, but A-GEM has much lower time (about 100 times faster) and memory cost (about 10 times lower). We obtained similar findings in Permuted MNIST.

Fig. 1 show the overall results on Split CUB and Split AWA. First, we notice that A-GEM achieves the best average accuracy with both the standard and the joint embedding model. Second, A-GEM has the lowest forgetting among methods that use a fixed capacity architecture. Third, we found that although PROG-NN [11] has no forgetting by construction, it has the worst memory cost by the end of training – as its number of parameters grows super-linearly with the number of tasks. On these two datasets, PROG-NN runs out of memory (OoM). Fourth, EWC performs only slightly better than VAN on this single pass LLL setting. Next, all methods perform similarly in terms of LCA, with A-GEM slightly better than all the other approaches. Finally, the use of task descriptors improves



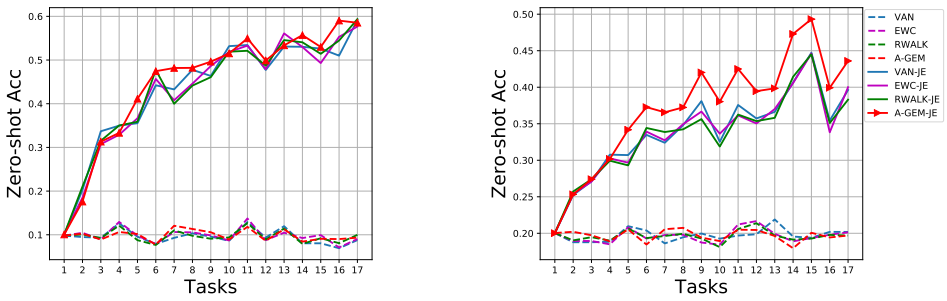
(a) Split CUB (b) Split AWA

Figure 1: *Performance of LLL models across different measures on Split CUB and Split AWA. On both the datasets,* PROG-NN *runs out of memory. The memory and time complexities of joint embedding models are the same as that of corresponding standard models and are hence omitted.*



(a) Split CUB (b) Split AWA

Figure 2: LCA *evaluated on the first ten mini-batches.*



(a) Split CUB (b) Split AWA

Figure 3: *Evolution of zero-shot performance as the learner sees new tasks.*

average accuracy across the board with A-GEM a bit better than all the other methods we tried. All joint-embedding models using task descriptors have better LCA performance.

Fig. 2 shows the effectiveness of task descriptors in the few-shot regime. Fig. 3, shows the 0-shot performance of LLL methods over time, showing a clear advantage of using compositional task descriptors and A-GEM. Overall, A-GEM offers the best trade-off between average accuracy performance and efficiency in terms of sample, memory and computational cost.

7 Conclusion

We studied the problem of Lifelong Learning when the learner can only do a single pass over the input data stream. Our experiments show that our approach, A-GEM, has the best trade-off between average accuracy and computational/memory cost. Compared to the original GEM algorithm, A-GEM is about 100 times faster and has 10 times less memory requirements; compared to regularization based approaches, it achieves significantly higher average accuracy at a small increase of compute/memory requirements. By using compositional task descriptors all methods can improve their few-shot performance, with A-GEM often being the best.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[2] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

[3] Mohamed Elhoseiny, Ahmed Elgammal, and Babak Saleh. Write a classifier: Predicting visual classifiers from unstructured text. *IEEE TPAMI*, 39(12):2539–2553, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

[5] James Kirkpatrick, Razvan Pascanu, Neil C. 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 of the United States of America (PNAS)*, 2016.

[6] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/cifar.html*, 2009.

[7] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 951–958. IEEE, 2009.

[8] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[9] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[10] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. In *NIPS*, 2017.

[11] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[12] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress and compress: A scalable framework for continual learning. In *International Conference in Machine Learning*, 2018.

[13] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[14] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[15] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[16] Ji Zhang, Yannis Kalantidis, Marcus Rohrbach, Manohar Paluri, Ahmed Elgammal, and Mohamed Elhoseiny. Large-scale visual relationship understanding. *arXiv preprint arXiv:1804.10660*, 2018.