Training Recurrent Neural Networks for Lifelong Learning

Shagun Sodhani* **Sarath Chandar *** **Yoshua Bengio**
Mila, Université de Montréal, Canada

Abstract

Capacity saturation and catastrophic forgetting are the central challenges of any parametric lifelong learning system. In this work, we study these challenges in the context of sequential supervised learning with emphasis on recurrent neural networks. To evaluate the models in life-long learning setting, we propose a curriculum-based, simple, and intuitive benchmark where the models are trained on a task with increasing levels of difficulty. As a step towards developing *true* lifelong learning systems, we unify *Gradient Episodic Memory* (a catastrophic forgetting alleviation approach) and *Net2Net* (a capacity expansion approach). Evaluation on the proposed benchmark shows that the unified model is more suitable than the constituent models for lifelong learning setting.

1 Introduction

Lifelong Machine Learning considers systems that can learn many tasks (from one or more domains) over a lifetime [25, 28]. This has several names in the literature: incremental learning [26], continual learning [22], explanation-based learning [27, 29], never ending learning [3], etc. The underlying idea is that the lifelong learning systems would be more effective at learning and retaining knowledge across different tasks.

Lifelong learning is an extremely challenging task for machine learning models because of two primary reasons: (i) As the model is trained on a new task or data distribution, it is likely to forget the knowledge it acquired from the previous task (*catastrophic forgetting* or *catastrophic interference* [18]). (ii) Any parametric model, however large, can only have a fixed amount of representational capacity to begin with. Given that we want the model to retain knowledge as it progresses through tasks, the model would eventually run out of capacity to store the knowledge acquired in the successive tasks and would have to increase its capacity.

Based on these challenges, we compile a list of desired properties of a model suitable for lifelong learning settings:

1. **Knowledge Retention** - As the model learns to solve new tasks, it should not forget how to solve the previous tasks.
2. **Knowledge Transfer** - Model should be able to reuse the knowledge acquired during previous tasks to solve the current task. If the tasks are related, this knowledge transfer would lead to faster learning and better generalization.
3. **Parameter Efficiency** - The number of parameters in the model should be bounded, or grow sub-linearly as new tasks are added.
4. **Model Expansion** - The model should be able to increase its capacity by expanding itself.

[*] Equal contribution. Contact Address: sshagunsodhani@gmail.com, sarathcse2008@gmail.com

2nd Continual Learning Workshop, Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

In a true lifelong learning setting, the model would experience a continual stream of training data that can not be stored. Hence the model would, at best, have access to only a small sample of the historical data. In such setting, we can not rely on past examples to train the expanded model from scratch and a zero-shot knowledge transfer is desired.

We propose to unify the Gradient Episodic Memory (GEM) model [17] and the *Net2Net* framework [4] to develop a model suitable for lifelong learning. The GEM model provides a mechanism to alleviate catastrophic forgetting, while allowing for improvement in the previous tasks by beneficial transfer of knowledge. *Net2Net* is a technique for transferring knowledge from a smaller, trained neural network to another larger, untrained neural network. We discuss both these models in detail in the following sections.

One reason hindering research in the lifelong learning is the absence of standardized benchmarks. Lomonaco and Maltoni [16] proposed a new benchmark for Continuous Object Recognition (CORe50) in the context of computer vision. Lopez-Paz and Ranzato [17] considered variants of MNIST and CIFAR-100 datasets for lifelong supervised learning. While these models help in studying specific challenges like catastrophic forgetting by abstracting the other challenges, they are quite far from real-life setting and are focused on non-sequential tasks. There has been no such benchmark available for lifelong learning in the context of sequential supervised learning.

We propose a simple, curriculum-based benchmark for evaluating lifelong learning models in the context of sequential supervised learning. The model starts with the simplest task (first task) and subsequently moves to the more difficult tasks. Each task has a well defined criteria of completion and the model can start training on a task only after finishing all the previous tasks in the curriculum. Each time the model finishes a task, it is evaluated on all the tasks in the curriculum (including the tasks that it has not been trained on so far) so as to compare the performance in terms of both catastrophic forgetting (for seen tasks) and generalization (to unseen tasks). In case the model fails to clear a task, it is expanded and trained on the current task again, followed by the evaluation steps. This enables us to evaluate how the expansion step affects generalization and catastrophic forgetting. The benchmark and task setup is described in detail in the appendix.

Our main contributions are as follows:

1. We propose a simple benchmark of tasks for training and evaluating models for learning sequential problems in lifelong learning setting.
2. We propose to unify Gradient Episodic Memory (a lifelong learning technique to alleviate catastrophic forgetting) with *Net2Net* (a capacity expansion technique).
3. We evaluate the proposed unified model on the benchmark of tasks and show that the unified model is better suited to the lifelong learning setting as compared to the constituent models.

2 Related Work

We provide a brief review of the prominent works dealing with catastrophic forgetting, capacity saturation and model expansion as these are the important aspects of lifelong learning.

2.1 Catastrophic Forgetting

Two key approaches for handling catastrophic forgetting are:

Freezing parts of the model as it trains on successive tasks or regularizing them so as not to not change *too-much* as the model train across different tasks. This approach is adopted by elastic weight consolidation (EWC) model [12] where as the model trains through a sequence of tasks, it slows the learning of weights which are important to the previous tasks. Liu et al. [15] extended this model by reparameterizing the network so as to approximately diagonalize the Fisher information matrix of the network parameters.

Rehearsal setup [24] where when learning on a given task, the model is also shown examples from the previous tasks. The setup protects against catastrophic forgetting and if the tasks are related, it helps in transferring knowledge across the tasks. If the tasks are unrelated, the setup still helps to protect against catastrophic forgetting. *iCaRL* [21] focuses on the class-incremental learning setting where the number of classes (in the classification system) increase on the fly. Similarly, Gradient Episodic

Memory approach [17] stores a subset of the observed examples from each task. When training on a given task, it is ensured that the loss on the data (for the previous task) does not increases (it may or may not decrease). This "positive-transfer" on the backward task enables it to outperforms many other models. One limitation of the model is the need to compute gradients corresponding to the previous task at each learning iteration. Given that GEM needs to store only few examples per task (in our experiments, we stored just one batch of examples), the storage cost is negligible. Given the strong performance and low memory cost, we use GEM as the first component of our unified model.

Some other related work includes Li and Hoiem [14] which uses the distillation principle [10] to incrementally train a single network for learning multiple tasks. Lee et al. [13] proposed incremental moment matching (IMM) which incrementally matches the moment of the posterior distribution of the neural network which is trained on the first and the second task, respectively. While this approach seems to give strong results, it is evaluated only on datasets with very few tasks.

2.2 Capacity Saturation and Model Expansion

Capacity saturation and model expansion have been extensively studied from different perspectives. Some works [6, 9] model the expansion step as a means of transferring knowledge from a small network to a large network to ease the training of deep neural networks . Much of these approaches focus on training the new network on a single supervised task where the data distribution does not change much and the previous examples can be reused several times. These assumptions are not valid for a true online lifelong learning setting.

Chen et al. [4] proposed using function-preserving transformations to expand a small, trained network into a large, untrained network as a means to accelerate training of larger neural networks. The paper evaluated the technique in context of single task supervised learning and mentioned continual learning as one of the motivations. Given that *Net2Net* enables zero-shot transfer of knowledge to the expanded network, we evaluate the approach in context of RNNs and use it as one of the components of the unified model.

Other related works include Progressive Networks Rusu et al. [23] which starts with a single column (neural network) and new columns are added as more tasks are encountered. One limitation there is that the number of columns (and hence the number of model parameters) increases linearly with the number of tasks. Also, when a new column is added, only a fraction of the new capacity is actually utilized. Aljundi et al. [1] build upon this idea and use a Network of Experts where each expert is trained for one task. During inference, a set of gating autoencoders are used to select the expert network to query.

Table 1: Comparison of different models in terms of the desirable properties they fulfill.

Property Model	Knowledge Retention	Knowledge Transfer	Parameter Efficiency	Model Expansion
EWC	✓		✓	
IMM	✓	✓	✓	
iCaRL	✓		✓	
NCM	✓		✓	
LwF	✓			
GEM	✓	✓	✓	
Net2Net			✓	✓
Progressive Nets	✓	✓		✓
Network of Experts	✓	✓		✓

Table 1 compares the different models in terms of the desirable properties they fulfill. NCM refers to *Nearest Mean Classifier* [19] and LwF refers to *Learning without Forgetting* [14]. We can rule out all the parameter-inefficient models (else the unified model would be parameter inefficient). One of the components should have expansion property so that the model can expand on the fly. This narrows down the choice to *Net2Net*. Since this model lacks both knowledge retention and knowledge transfer (across tasks), we could pick either IMM or GEM. IMM is evaluated for very few tasks while GEM seems to work well for larger number of tasks (as reported in the paper). Hence Given these considerations, we choose GEM as the second component. The unified model has all the 4 properties.

3 Model

In this section, we describe how the GEM Model and the *Net2Net* model can be used together in the lifelong learning framework. The individual models are described in detail in appendix. The detailed task setup has also been described in the appendix.

Assume that the task setup (domain and model) have been specified. We reset the episodic memory to be empty and start training on the first task (simplest task) and subsequently move to more difficult tasks. When we are training the model on the l^{th} task, we compute the current gradient vector and project (as per GEM formulation) to ensure that it does not increase the loss corresponding to the data points that are stored in the memory. This projected gradient is used to update the weights of the model and we call this update step as the *GEM-Update*. The model is trained on the current task for a fixed number of iterations and the last m training examples are stored in the episodic memory. If the model completes the task, it starts training on the next task. Otherwise, if the model has not been expanded yet, we expand it to have larger size (increase capacity) train it further on the current task. The expanded model, after training, is re-evaluated. If it completes the task, it progress to the next task, else the training is terminated. Irrespective of weather a task is learnt or not, the model is evaluated on all the tasks - to measure its *Previous Task Performance* and *Future Task Performance*.

4 Experiments

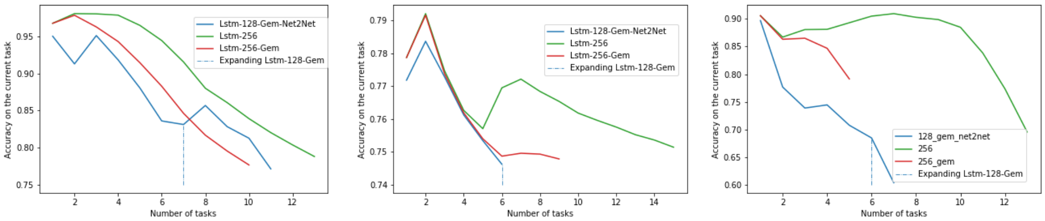

Figure 1: Task-wise accuracy for the different models on different domains (Copy, Associative Recall, SSMNIST respectively). Higher curves have higher *current task accuracy* and curves extending more have completed more tasks. For both Copy and SSMNIST, the proposed *Lstm-128-Gem-Net2Net* model clears more levels than *Lstm-256-Gem* model. Before the dotted line, the *Net2Net* models are of much smaller capacity (hidden size of 128) as compare to the larger models which have a high capacity from the starting. This is the reason why larger models have better accuracy initially.

4.1 Models

For each task, we consider a standard recurrent model operating in the lifelong learning setting. We use an LSTM model with hidden state size of 128 and refer the model as *Lstm-128*. We now consider the different aspects of training a lifelong learning system and describe how the model variants capture these aspects. We start the training with a standard model (*Lstm-128*). Since we want to avoid catastrophic forgetting, we use the *GEM* update (*Lstm-128-Gem* model). At some level, the model's capacity would saturate and we would expand the model (*Lstm-128-Gem-Net2Net*). Alternatively, we could have started the training with a larger model (*Lstm-256*) and used the GEM update (*Lstm-256-Gem*). Strategy of always starting training with a large network would not work in practice because in the lifelong learning setting, we can not know what network would be sufficiently large to solve all the tasks beforehand. *Lstm-128-Gem-Net2Net* gets around this problem by increasing the capacity on the fly as and when needed. For the performance on the current task, *LSTM-256* should be treated as the gold standard. To test the viability of the model expansion approach, we could treat the *Lstm-256* and *Lstm-256-Gem* models as gold standard for *Current Task Accuracy* and *Previous Task Accuracy* respectively. We can evaluate how close our unified model (*Lstm-128-Gem-Net2Net*) is to these gold standards. While we do not have a gold standard for the *Future Task Accuracy*, both *Lstm-256-Gem* and *Lstm-256* are reasonable baselines. Note that we have 3 different baselines for 3 settings and our proposed model is compared to these 3 models (each specialised for a specific usecase).

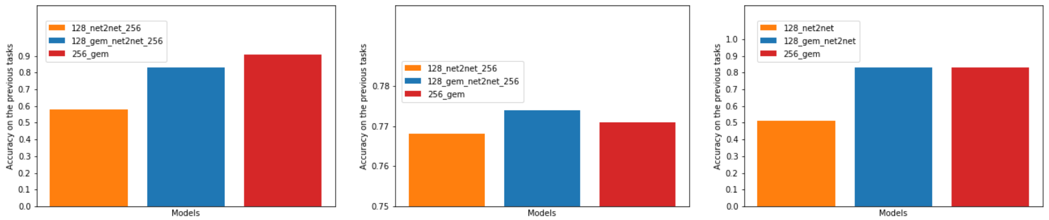

Figure 2: *Previous Task Accuracy* for the different models on different domains (Copy, Associative Recall, SSMNIST respectively). Higher bars have better accuracy on previously seen tasks (more robust to catastrophic forgetting). For Copy and SSMNIST, the *GEM* based models are much better than their counterparts and for the Associative Recall, all the 3 models are very similar. Importantly, the unified model (*Lstm-128-Gem-Net2Net*) is either better or as good as the gold-standard (*Lstm-256-Gem*) model which suggests that the unified model is quite robust against catastrophic forgetting.

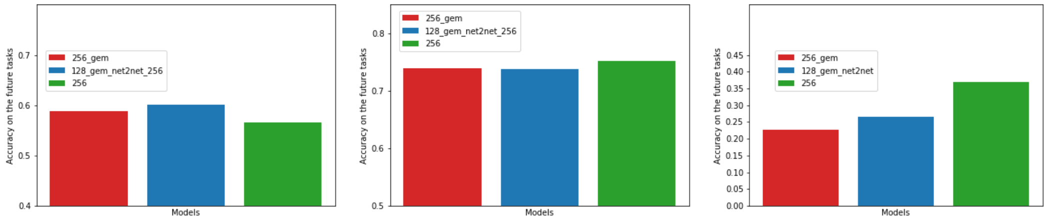

Figure 3: *Future Task Accuracy* for the different models on different domains (Copy, Associative Recall, SSMNIST respectively). Higher bars have better accuracy on unseen tasks and are more beneficial for achieving knowledge transfer to future tasks. Even though the unified model does not have any component for specifically generalizing to the future tasks, we expect the unified model to have at least as good future task accuracy as *Lstm-256-Gem* model and comparable to *Lstm-256*. Interestingly, our model outperforms the *Lstm-256* model for Copy task.

4.2 Results

Figure 1 shows the trend of the current task accuracy for the different models on different domains. Models with higher curves have better accuracy (current task) and models which learn more tasks are spread out more along the x-axis. We compare the performance of the proposed model *Lstm-128-Gem-Net2Net* with the gold standard *Lstm-256*. We additionally compare with *Lstm-256-Gem* model as both this model and the proposed model are constrained to use some of their capacity on the previous tasks. The dotted line corresponds to the expansion step when the model is not able to learn the current task and had to expand. This shows that using *Net2Net* enables learning on newer tasks. We highlight that before expansion, the proposed model *Lstm-128-Gem-Net2Net* have a much smaller capacity (128 hidden dim) as compared to 2 other models which started with a much larger capacity (256 hidden dim). This explains why the larger models have much better performance in the initial stages. Post expansion, the proposed model overtakes the GEM based model in 2 out of 3 cases. This suggests that using *GEM* update comes at the cost of reducing the capacity for the current task. Using *Net2Net* with *GEM* enables the model to account for this loss of capacity. Refer appendix for additional discussion on results for the Associative Recall.

Figure 2 shows the *Previous Task Accuracy* for the proposed model *Lstm-128-Gem-Net2Net* and gold standard *Lstm-256-Gem*. We additionally consider the *Lstm-128-Net2Net* to demonstrate that Gem update is essential to have good performance on the previous tasks. The most important observation is the relative performance of *Lstm-128-Gem-Net2Net* and *Lstm-256-Gem* models. The *Lstm-128-Gem-Net2Net* model started as a smaller model, consistently learnt more tasks than *Lstm-256-Gem* model and is still almost as good as *Lstm-256-Gem* model in terms of *Previous Task Accuracy* . This shows that the unified model is very robust to catastrophic forgetting.

Figure 3 shows the *Future Task Accuracy* for the proposed model. Since we do not have any gold standard for this setting, we consider both *Lstm-256-Gem* and *Lstm-256* models as they both are reasonable baselines. The general trend is that our proposed model is quite close to the baseline models for 2 out of 3 tasks. Note that both the larger models started training with a much larger capacity and further, the *Lstm-256* model is not constrained to preserve performance on the previous tasks. This explains why this model is able to outperform our proposed model for 1 task.

It is important to note that we are using a single model (*Lstm-128-Gem-Net2Net*) and comparing it with gold-standards in 3 different contexts - performance on the current task, previous task and future tasks. Our model can provide strong performance on the previous tasks (countering catastrophic forgetting) and expand as many times as needed (ensuring capacity expansion).

5 Conclusion

In this work, we study the problem of capacity saturation and catastrophic forgetting in lifelong learning in the context of sequential supervised learning. We propose to unify Gradient Episodic Memory (a catastrophic forgetting alleviation approach) and Net2Net (a capacity expansion approach) to develop a model that is more suitable for lifelong learning. We also propose a curriculum based evaluation benchmark where the models are trained on a task with increasing levels of difficulty. This enables us to sidestep some of the challenges that arise when studying lifelong learning. We conduct experiments on the proposed benchmark tasks and show that the unified model is better suited to the lifelong learning setting as compared to the two constituent models.

References

[1] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert Gate: Lifelong Learning with a Network of Experts. *ArXiv e-prints*, November 2016.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[3] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta, 2010.

[4] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[5] William S Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162, 1960.

[6] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born Again Neural Networks. *ArXiv e-prints*, May 2018.

[7] A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines. *ArXiv e-prints*, October 2014.

[8] Çaglar Gülçehre, Sarath Chandar, and Yoshua Bengio. Memory augmented neural networks with wormhole connections. *CoRR*, abs/1701.08718, 2017. URL `http://arxiv.org/abs/1701.08718`.

[9] Steven Gutstein, Olac Fuentes, and Eric Freudenthal. Knowledge transfer in deep convolutional neural nets. *International Journal on Artificial Intelligence Tools*, 17(03):555–567, 2008.

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

[11] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *ArXiv e-prints*, December 2016.

[13] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4652–4662. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7051-overcoming-catastrophic-forgetting-by-incremental-moment-matching.pdf`.

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[15] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *arXiv preprint arXiv:1802.02950*, 2018.

[16] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017. URL `http://proceedings.mlr.press/v78/lomonaco17a.html`.

[17] David Lopez-Paz and Marc Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6467–6476. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7225-gradient-episodic-memory-for-continual-learning.pdf`.

[18] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[19] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Computer Vision–ECCV 2012*, pages 488–501. Springer, 2012.

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[21] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

[22] Mark B Ring. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

[23] 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.

[24] Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer, 2002.

[25] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, page 05, 2013.

[26] Ray J Solomonoff. A system for incremental learning based on algorithmic probability. In *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition*, pages 515–527, 1989.

[27] Sebastian Thrun. Explanation-based neural network learning. In *Explanation-Based Neural Network Learning*, pages 19–48. Springer, 1996.

[28] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[29] Sebastian Thrun. *Explanation-based neural network learning: A lifelong learning approach*, volume 357. Springer Science & Business Media, 2012.

6 Appendix

6.1 Hyper Parameters

All the models are implemented using PyTorch 0.4.1 [20]. Adam optimizer [11] with a learning rate of 0.001 and ReLU nonlinearity are used. We used one layer LSTM models with hidden dimensions of size 128 and 256. *Net2Net* is used to expand LSTM models of size 128 to 256. For GEM model, we keep one minibatch of data per task for obtaining projected gradients. We follow the guidelines and hyperparameter configurations as specified for both *GEM* and *Net2Net* models.

6.2 Gradient Episodic Memory (GEM)

GEM [17] helps to alleviate the catastrophic forgetting while enabling positive transfer on the backward tasks. It uses an episodic memory buffer B that stores some training examples from each task that the model has been trained on so far. In practice, the buffer has a fixed size, say B_{size}. If the number of tasks T is known beforehand, B_{size}/T memory slots can be reserved for each task. Otherwise, we can reduce the number of memory slots per task as new tasks are encountered. As the model is training on the l^{th} task, care is taken to ensure that the current gradient updates do not increase the loss on the examples already saved in the memory. A simple approach could be save only the last few examples from each task. For our case, we store only 1 minibatch of examples (10 examples) per task.

Given that the model is training on the l^{th} task, the gradient g_l is computed with respect to the current task. If the current gradient g_l increases the loss on any of the previous tasks, it is projected to the closest gradient $\tilde{g}_l$ (where closeness is measured in terms of *l2* norm) such that the condition is no more violated. The projected gradient update $\tilde{g}_l$ can be obtained by solving the following set of equations

$$\begin{aligned} \text{minimize}_{\tilde{g}} \quad & \frac{1}{2} \quad \|g - \tilde{g}\|_2^2 \\ \text{subject to} \quad & \langle \tilde{g}, g_k \rangle \geq 0 \text{ for all } k < t. \end{aligned} \tag{1}$$

To solve (1) efficiently, the authors use the primal of a Quadratic Program (QP) with inequality constraints:

$$\begin{aligned} \text{minimize}_z \quad & \frac{1}{2} z^\top C z + p^\top z \\ \text{subject to} \quad & Az \geq b, \end{aligned} \tag{2}$$

where $C \in \mathbb{R}^{p \times p}$, $p \in \mathbb{R}^p$, $A \in \mathbb{R}^{(t-1) \times p}$, and $b \in \mathbb{R}^{t-1}$. The dual problem of (2) is:

$$\begin{aligned} \text{minimize}_{u,v} \quad & \frac{1}{2} u^\top C u - b^\top v \\ \text{subject to} \quad & A^\top v - Cu = p, \\ & v \geq 0. \end{aligned} \tag{3}$$

If $(u^\star, v^\star)$ is a solution to (3), then there is a solution $z^\star$ to (2) satisfying $Cz^\star = Cu^\star$ [5].

The primal GEM QP (1) can be rewritten as:

$$\begin{aligned} \text{minimize}_z \quad & \frac{1}{2} z^\top z - g^\top z + \frac{1}{2} g^\top g \\ \text{subject to} \quad & Gz \geq 0, \end{aligned}$$

where $G = -(g_1, \ldots, g_{t-1})$, and the constant term $g^\top g$ is discarded. This new equation is a QP on p variables (where p is the number of parameters of the neural network). Since the network could have a lot of parameters, it is not feasible to solve this equation and the dual of the GEM QP is considered:

$$\begin{aligned} \text{minimize}_v \quad & \frac{1}{2} v^\top G G^\top v + g^\top G^\top v \\ \text{subject to} \quad & v \geq 0, \end{aligned} \tag{4}$$

since $u = G^\top v + g$ and the term $g^\top g$ is constant. This is a QP on $t - 1 \ll p$ variables (where t is the number of observed tasks so far). Solution for the dual problem (4), $v^\star$, can be used to recover the projected gradient update as $\tilde{g} = G^\top v^\star + g$. The authors recommend adding a small constant $\gamma \geq 0$ to $v^\star$ as it helps to bias the gradient projection to updates that favoured benefitial backwards transfer.

6.3 Net2Net

Training a lifelong learning system on a continual stream of data can be seen as training a model with infinite amount of data. As the model experiences more data points, the size of its effective training dataset increases and the network needs to expand its capacity to continue training. *Net2Net* [4] is a prominent technique for zero shot knowledge transfer from a small, trained (teacher) network to a larger, untrained (student) network using function preserving transformations.

Given a teacher network represented by the function $y = f(x, \theta)$ (where θ refers to the network parameters), a new set of parameters ϕ are chosen such that $\forall x, f(x, \phi) = g(x, \theta)$. The paper considered two variants of this approach - *Net2WiderNet* which increases the width of an existing network and *Net2DeeperNet* which increases the depth of the existing network. The main benefit of using function-preserving transformations is that the student network immediately performs as well as the original network without having to go through a period of low performance. The newly created larger network can then continue training on the incoming data. We used the *Net2WiderNet* variant of *Net2Net* model for all our experiments. In theory, there is no restriction on how many times the *Net2Net* operation is invoked though we limit our experiments to using the operation just once.

We use the *Net2WiderNet* for expanding the capacity of the model. The *Net2WiderNet* formulation is as follows:

Assume that we start with a fully connected network where we want to widen layers i and $i + 1$. The weight matrix associated with layer i is $\boldsymbol{W}^{(i)} \in \mathbb{R}^{m \times n}$ and that associated with layer $i + 1$ is $\boldsymbol{W}^{(i+1)} \in \mathbb{R}^{n \times p}$. Layer i may use any element-wise non-linearity. When we widen layer i, the weight matrix $\boldsymbol{W}^{(i)}$ expands into $\boldsymbol{U}^{(i)}$ to have q output units where $q > n$. Similarly, when we widen layer $i + 1$, the weight matrix $\boldsymbol{W}^{(i+1)}$ expands into $\boldsymbol{U}^{(i+1)}$ to have q input units.

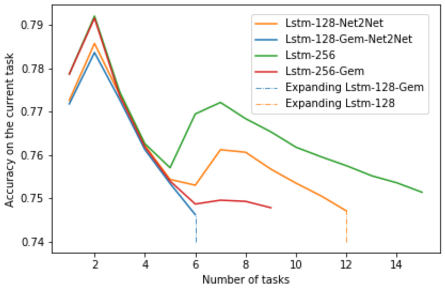

Figure 4: Task-wise accuracy for the different models for Associative Recall. Note that using *Net2Net* variant gives almost no performance improvement. Further, *Lstm-128-Gem-Net2Net* and *Lstm-256* models are within 1% gap even though the former model has much lesser capacity than the latter model. This indicates that the different tasks in associative recall are quite similar to each other and a transfer learning based model would be a better fit for such tasks.

A random mapping function $g : \{1, 2, \cdots, q\} \to \{1, 2, \cdots, n\}$, is defined as:

$$g(j) = \begin{cases} j & j \le n \\ \text{random sample from } \{1, 2, \cdots n\} & j > n \end{cases}$$

For expanding $\boldsymbol{W}^{(i)}$, the columns of $\boldsymbol{U}^{(i)}$ are randomly chosen from $\boldsymbol{W}^{(i)}$ using g as shown:

$$\boldsymbol{U}^{(i)}_{k,j} = \boldsymbol{W}^{(i)}_{k,g(j)}$$

Notice that the first n columns of $\boldsymbol{W}^{(i)}$ are copied directly into $\boldsymbol{U}^{(i)}$.

The rows of $\boldsymbol{U}^{(i+1)}$ are randomly chosen from $\boldsymbol{W}^{(i+1)}$ using g as shown:

$$\boldsymbol{U}^{(i+1)}_{j,h} = \frac{1}{|\{x|g(x) = g(j)\}|}\boldsymbol{W}^{(i+1)}_{g(j),h}$$

Similar to the previous case, the first n rows of $\boldsymbol{W}^{(i+1)}$ are copied directly into $\boldsymbol{U}^{(i+1)}$.

The replication factor, (given by $\frac{1}{|\{x|g(x)=g(j)\}|}$), is introduced to make sure that the output of the two models is exactly the same. This procedure can be easily extended for multiple layers. Similarly, the procedure can be used for expanding convolutional networks (where layers will have more convolution channels) as convolution is multiplication by a doubly block circulant matrix).

6.4 Results

6.4.1 Discussion on Associative Recall

Associative recall is the only task where our proposed model does worse than *LSTM-256-Gem*. To understand this better, we plot different model variants in the same plot. This includes *LSTM-256*, *LSTM-128-Net2Net*, *LSTM-128-Gem-Net2Net* and *LSTM-256-Gem* in figure 4. We observe that using *Net2Net* variant gives almost no performance improvement. We speculate that the reason is that different tasks in associative recall are quite similar to each other. This can be seen in figure-1 where the current level accuracy for *Lstm-128-Gem-Net2Net* and *Lstm-256* models are within 1% gap even though the former model has much lesser capacity than the latter model. Similarly in Figure-2 and Figure-3, the performance on previously seen and unseen tasks is very similar for all the models. This provides an interesting insight about our model - while it works reasonably well in general, the unified model may not be very strong when the tasks are very related to each other. We believe a transfer learning based model would be a better fit for such tasks.

6.5 Tasks and Setup

In this section, we describe the tasks and the training and evaluation setup that we use for benchmarking the lifelong learning models in the context of sequential supervised learning. In a true lifelong learning setting, the training distribution can change arbitrarily and no explicit demarcation exists between data distribution corresponding to different tasks. This makes it extremely hard to study how catastrophic forgetting and generalization capabilities of the model evolve with training. We sidestep these challenges by using an experimental setup where we have full control over the training data distributions. This way, we can explicitly control when the model experience different data distributions and in what order. Specifically, we train our model in a curriculum style setting [2] where the tasks are ordered by difficulty. The model starts from the simplest task and progresses towards harder tasks. We consider the following three tasks:

6.5.1 Copy Task

The copy task is an algorithmic task introduced in [7] to test whether the networks can learn to store and recall a long sequence of random vectors. Specifically, the network is presented with a sequence of seven bit random vectors followed by a delimiter flag. An eighth bit is used for the delimiter flag which is zero at all time steps except for the end of the sequence. The network is trained to generate the entire sequence except the delimiter flag. The levels are defined by the length of the input sequence. We start with a sequence length of 5 and increase the length in the steps of 3 and go till the maximum sequence length of 62 (20 levels). Bit-wise accuracy is the metric.

6.5.2 Associative Recall Task

The associative recall task is another algorithmic task introduced in [7]. In this task, the network is shown a list of items where each item is a sequence of 8-bit binary vectors, bounded on the left and the right by delimiter symbols. Once the network is shown some items, it is shown one of the items at random and required to produce the next item in the sequence. We set the length of each item to be 3. The levels are defined by the number of items. We start with 5 items per sequence and increase the number of items in the steps of 3 and go till 62 items per sequence (20 levels). Bit-wise accuracy is the metric.

6.5.3 Sequential Stroke MNIST Task

Sequential Stroke MNIST (SSMNIT) task was introduced in [8] to test the long-term dependency modeling capabilities of RNNs. In this task, each MNIST digit image I is represented as a sequence of quadruples $\{dx_i, , dy_i, eos_i, eod_i\}_{i=1}^{T}$. T is the number of pen strokes needed to define the digit, (dx_i, dy_i) denotes the pen offset from the previous to the current stroke (can be 1, -1 or 0), eos_i is a binary valued feature to denote end of stroke and eod_i is another binary valued feature to denote end of the digit. The average number of strokes per digit is 40. Given a sequence of pen-stroke sequences, the task is to predict the sequence of digits corresponding to each pen-stroke sequences in the given order. This is an extremely challenging task as the model is required to predict the digit based on pen-stroke sequence, count the number of digits and then generate the digits in the same order after seeing the entire sequence of pen-strokes. Number of digits define the levels. Given that this task is more challenging than the other two tasks, we start the task with 1 digit per sequence and increase it in steps of 1, till it reaches the maximum of 20 digits per sequence. Per-digit accuracy is the metric.

6.5.4 The Setup

So far, we have defined 3 tasks and multiple levels within each task. Alternatively, the tasks can be seen as 3 domains and the multiple difficulty levels can be seen as tasks. From now on, we employ the *domain-task* analogy to be consistent with the literature in lifelong learning. Thus we have 3 domains and multiple tasks (in increasing order of difficultly) per domain. To be closely aligned with the *true* lifelong learning setting, we train all the models in an online manner where none of the training examples are repeated. However the network sees mini-batches of 10 examples at a time (instead of one example at a time which is a common setup in online learning). This choice is made to exploit the computational benefits in using mini-batches. However, every mini-batch is generated randomly and examples are not repeated. Hence a separate validation or test set is not needed. For each task, we report the *current task accuracy* as an indicator of performance on the current task. If the running-average of the accuracy, averaged over last k batches, is greater-than or equal-to $c\%$, the model is said to have completed the task and we start training on the next task. If the model fails, we stop the training. We let all networks see m number of mini-batches irrespective of when they clear the task (to make sure all models see similar number of examples). This training procedure is repeated for all the domains. k = 100, and m = 10000 for all the tasks. c = 80 for Copy and c = 75 for Associative Recall and SSMNIST.

In a lifelong learning setting, it is very important for the model to retain knowledge from the previous tasks while generalizing to new levels. Hence, each time the model learns a task, we evaluate it on all the previous tasks (that it has been trained on so far) and report the average of these accuracies as the *per-task-previous-accuracy*. When the model's training is stopped (because the model fails to clear a task), we report the average of the different *per-task-previous-accuracy* metrics (corresponding to the different tasks completed by the model) as *previous-task-accuracy*. This single metric can be used to quantify the effect of catastrophic forgetting.

Another interesting aspect of lifelong learning is generalization to the unseen tasks. Analogous to the *per-task-previous-accuracy* and *previous-task-accuracy*, we consider the *per-task-future-accuracy* and *future-task-accuracy*. There is no success criteria associated with this evaluation phase and the metrics are interpreted as a proxy of model's resilience to catastrophic forgetting and model's ability to generalize across levels. We are able to test generalization to new tasks simply because future tasks are similar to the current task but with higher difficulty level. Note that the benchmark tasks can have levels beyond 20 as well. We limited our evaluation to 20 levels as none of the models could complete all the levels.

In the context of lifelong learning systems, the model needs to be made larger once it has saturated its capacity so that it can keep learning from the incoming data. We simulate this scenario in our experimental setting as follows: If the model fails to complete a given task, we use some capacity expansion technique and expand the original RNN into a network with larger hidden state size. The expanded model is then allowed to train on the current task for 20000 iterations. From there, the expanded model is evaluated (and trained on subsequent tasks) just like a regular model. If the expanded model fails on any task, the training is terminated. Note that this termination criteria is a part of our evaluation protocol. The model can be expanded multiple times.

6.5.5 Discussion

For all the three domains, it can be reasonably argued that the tasks become more challenging as the sequence length increases. Hence, we define a curriculum over these tasks by controlling the length of the input sequences. We note that our experimental setup is different from the real life setting where we have no control over the difficulty or complexity of the incoming data points. This trade off has several advantages. First, as the tasks are arranged in increasing order of difficulty, it becomes easier to quantify the change in performance as the evaluation data distribution becomes different from the training data distribution. Second, it enables us to extrapolate the capacity of the model with respect to the unseen tasks. If the model is unable to solve the n^{th} task, it is unlikely to solve any of the subsequent tasks as they are harder than the current task. Thus, we can use the number of tasks solved by the model as a ordinal indicator of its capacity - more tasks a model can solve, more is its capacity. Further, as the data distribution becomes more difficult, model is forced to use its representational capacity to the optimal.

Our choice of data-sets also makes it easier for us to evaluate the robustness of our unified model in different scenarios. Consider the *associative recall* task where the model has to search for a fixed length block in the given input sequence of blocks. For any sequence length, the block to be searched is chosen randomly. So even though the sequence length changes across different tasks, the tasks remain closely related. Further, even though the number of blocks in the input sequence changes, the model always has to produce a fixed size output. Contrast this with the remaining two tasks where the size of the output changes as the input also changes. In this sense, we expect the different levels in the *associative recall* task to be more related than the other two tasks and hence would expect a model to generalize more easily across different tasks. Now consider the *SSMNIST* task where each digit could have a different number of strokes. The output is considered to be correct only when the model produces all the digits in the desired order. Training on different levels of this task makes the model highly susceptible to over-fitting (as we demonstrate in Figure-5). Having these varied data-sets enables us to benchmark our model when the different levels are very similar or are so difficult that the model is highly susceptible to over-fitting to the current level.

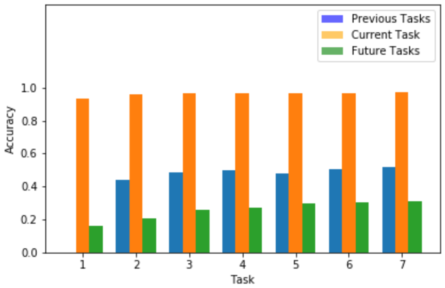

Figure 5: Per-level accuracy on previous task, current task, and future tasks for a 128 dimensional LSTM trained in the SSMNIST domain. The model heavily overfits to the sequence length.