Continual Learning Through Expandable Elastic Weight Consolidation

Andrew Jones
Department of Computer Science
James Madison University
Harrisonburg, VA 22807
`jonesaj@dukes.jmu.edu`

Nathan Sprague
Department of Computer Science
James Madison University
Harrisonburg, VA 22807
`spragunr@jmu.edu`

Abstract

Deep neural networks have exhibited impressive performance across a wide range of individual tasks. However, progress toward artificial general intelligence will require learning systems that are capable of *continual learning*. There has been some recent promising work in the area of continual learning for deep neural networks. In particular Elastic Weight Consolidation (EWC) introduces a loss function that penalizes weight changes in proportion to their importance for earlier tasks [3]. We propose solutions to two key weaknesses of the EWC algorithm: First, as the number of tasks increases, preserving weights via EWC eventually results in numerical instability in the training process as the weight preservation penalty grows without bound. Second, any fixed-sized network has a limited capacity to learn new tasks. We address the first issue by introducing a per-parameter dynamic learning rate. We address the second issue through an algorithm that monitors performance and automatically expands the network when necessary. We demonstrate the usefulness of our approach on sequential versions of MNIST and CIFAR 100.

Introduction

Continual learning requires retaining sufficient performance on previously encountered tasks while adapting to a newly presented task. Continual learning also provides opportunities for transfer of knowledge between tasks [13]. Forward transfer allows learning of future tasks more efficiently [8]. Backward transfer improves accuracy on previously encountered tasks [8].

Simply presenting multiple tasks in sequence to single neural networks results in “Catastrophic Forgetting” [1]: weight updates for later tasks quickly degrade performance on previousl tasks.

There have been several recent proposals for overcoming the problem of catastrophic forgetting [3, 11, 15, 8, 12, 6, 10, 7, 9]. Most of these work by introducing modified learning algorithms that encourage the network to retain previously learned information when learning new tasks. A particularly elegant proposal is Elastic Weight Consolidation (EWC). EWC selectively penalizes weight changes by incorporating a metric into the loss function which accounts for each weight’s importance to preserving accuracy on previously encountered tasks [3]. The EWC loss function combines (a) the loss on the input data for the most recent task with respect to target labels and (b) a penalty term that penalizes weight changes according to the Fisher Information calculated for each weight for the previous task(s).

A weakness of EWC, and similar approaches, is that they assume a fixed-sized network. This requires the system designer to anticipate the required network size in advance. A network that is too

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

small will eventually run out of capacity as new tasks are introduced. A network that is too large will waste computational resources and may be prone to over-fitting.

An alternative direction is suggested by progressive networks [11]. The progressive network architecture involves creating a new model for each new task. All previously learned models are provided as input to the new model in the form of lateral connections. This makes it possible for new models to benefit from previously learned representations. Since the previous models do not experience weight updates, the problem of catastrophic forgetting is avoided. The downside of this approach is that the number of network parameters grows quadratically with the number of tasks.

This paper makes two main contributions. First, we describe a previously unrecognized weakness of the Elastic Weight Consolidation algorithm that leads to numerical instability during training. We suggest a method for addressing this problem and demonstrate that it results in successful learning. Second, we introduce a simple algorithm for automatically increasing a network's size when additional capacity is required to learn new tasks. The proposed algorithm can be seen as a hybrid approach that combines the strengths of EWC and progressive networks.

Prior work

Two particular approaches to continual learning, Gradient Episodic Memory and Dynamically Expandable Networks, are especially relevant to our work.

Gradient Episodic Memory (GEM) provides the network with an "Episodic Memory", consisting of a subset of observed samples from previous tasks [8]. Incorporated into the GEM loss function is a term that constrains weight changes to ensure that loss on these memories is not increased (but may be decreased) as the network integrates new task information. The average accuracy across all tasks attained by GEM is a function of the size of its episodic memory, which is allocated upon network initialization and shared by all learned tasks. As gradients are computed for each new task, expansion of this memory presents an increase in both the computational complexity of training a new task and the storage space required to maintain remembered data samples.

Developed concurrently with our proposed method, the Dynamically Expandable Network (DEN) determines which network parameters are relevant to a new task when the task is presented, and selectively retrains those weights to accommodate the new task. It also incrementally adds new units until a performance target is reached, and drops units that are not needed for the new task. Finally, instead of using an EWC-style penalty, DEN monitors units to see which have changed significantly and duplicates those individual units [6]. The DEN approach is similar in spirit to the work presented here. It focuses more on limiting network expansion at the expense of a significantly more complex algorithm requiring more hyperparameter selection.

Methods

Using EWC, loss is calculated as follows for a network which is being trained on task $\mathcal{T}_B$ after having learned task $\mathcal{T}_A$:

$$L(\theta) = L_{\mathcal{T}_B}(\theta) + \sum_i \frac{\lambda}{2} F_{\mathcal{T}_A,ii}(\theta_i - \theta^*_{\mathcal{T}_A,i})^2 \tag{1}$$

where $L_{\mathcal{T}_B}(\theta)$ is the loss on task $\mathcal{T}_B$ alone, i is the parameter number, λ determines the relative weighting of learning the new task versus retaining the old task, $F_{\mathcal{T}_A,ii}$ is the Fisher Information Matrix (FIM) diagonal value corresponding to parameter i, and $\theta^*_{\mathcal{T}_A,i}$ is the value of parameter i after training on task $\mathcal{T}_A$.

When generalizing to n tasks, Equation 1 becomes:

$$L_{1:n}(\theta) = L_{\mathcal{T}_n}(\theta) + \sum_{t=1}^{n-1} \sum_i \frac{\lambda}{2} F_{\mathcal{T}_t,ii}(\theta_i - \theta^*_{\mathcal{T}_t,i})^2 \tag{2}$$

where $L_{\mathcal{T}_n}(\theta)$ is the loss on the current task, $F_{\mathcal{T}_t,ii}$ is the FIM diagonal value corresponding to parameter i, calculated after training on task $\mathcal{T}_t$, and $\theta^*_{\mathcal{T}_t,i}$ is the value of parameter i after training on task $\mathcal{T}_t$. The subscript on $L_{1:n}$ indicates that the overall loss function is dependent on all $\mathcal{T}_{1:n}$.

Equation 2 can be algebraically manipulated to yield:

$$L_{1:n}(\theta) = L_{\mathcal{T}_n}(\theta) + \frac{\lambda}{2} \sum_i \left(\theta_i^2 \sum_{t=1}^{n-1} F_{\mathcal{T}_t,ii} - 2\theta_i \sum_{t=1}^{n-1} F_{\mathcal{T}_t,ii} \theta^*_{\mathcal{T}_t,i} + \sum_{t=1}^{n-1} F_{\mathcal{T}_t,ii} \theta^{*2}_{\mathcal{T}_t,i} \right) \tag{3}$$

By using this form we can avoid explicitly storing previous FIM diagonals and the weight vectors learned for each previous task.

EWC instability

The inner sums in Equation 3 may grow without bound as the number of tasks increases. This means that the magnitude of the corresponding term in the loss function may eventually become much larger than the loss on the current task. In essence, this places the current weight settings at the bottom of a steep valley in the minimization landscape. This will tend to cause learning to diverge rather than gradually freezing those weights as the penalty for changing them increases. This issue was probably not encountered by Kirkpatrick et. al. [3] because in their experiments the number of tasks was kept relatively small relative to the size of the network. Instead of a gradual reduction in performance as the number of tasks increases, Stochastic Gradient Descent (SGD) eventually diverges as the gradient values related to the loss function increase.

We observed this issue in the sequential version of MNIST introduced by Kirkpatrick et. al. [3]. The sequential MNIST domain involves generating new tasks by performing random pixel permutations on the MNIST images. In our experiments, we observed that EWC consistently diverged within 18 tasks when training a network with 1 hidden layer containing 20 nodes (learning rate = 0.1, λ = 15).

We considered several approaches to handling this issue, including gradient clipping and standard adaptive learning rate algorithms such as Adam [2] or Adadelta [14]. In preliminary experiments, we found the most effective approach to be adaptively scaling the learning rate on a per-parameter basis according to the Fisher Information values. We use the following variable learning rate:

$$\alpha_i = \frac{\alpha}{\max\{1, \frac{\rho\lambda}{2} \sum_{t=1}^{n-1} F_{\mathcal{T}_t,ii}\}} \tag{4}$$

where α_i is the updated learning rate for parameter i, α is the overall learning rate and ρ determines how sensitive the learning rate will be to increasing Fisher Information values. For the experiments presented below, ρ is set to 2.0.

The denominator in Equation 4 is structured so that it will never be less than 1. This prevents increasing the learning rate while allowing the learning rate to be reduced for weights that are particularly important for maintaining accuracy on previously encountered tasks. This addresses the issue of training instability by decreasing the learning rate for exactly the parameters that are subject to large gradients as a result of the EWC penalty term.

Using our adaptive learning rate prevented network divergence up to 100 tasks in the sequential MNIST domain with only a gradual degradation of network performance. A focus of future work will be to perform a rigorous comparison of our approach to other adaptive learning rate algorithms.

Network expansion

While the variable learning rate described above addresses the problem of divergence during training, it does not address the more fundamental issue of limited network capacity. As the number of tasks increases, any fixed-size network will reach a point where it must either fail to learn the new task, or suffer a decrease in performance on previously learned tasks, depending on the value of λ.

We address this by monitoring the performance on the training set for the most recently encountered task. Our algorithm expands the size of the hidden layers to increase network capacity if the network fails to reach a threshold performance value after a fixed number of training epochs. Network parameters which were part of the original, smaller network are reset to their values at the end of training on the last task on which the network successfully met the threshold. Parameters are then expanded in all required dimensions to accommodate the updated layer sizes and randomly initialized. The Fisher Information sums from Equation 3 are initialized to zero for the newly created

Algorithm 1: Train Expandable Network

```
Input: Tasks $\langle \mathcal{T}_1, \mathcal{T}_2, ...\mathcal{T}_N \rangle$, Expansion threshold $\tau$, Epochs $K$
Output: Final Weights $\theta_N$
1: $\theta_1 \leftarrow IntializeWeights()$
2: for all $\mathcal{T}_t$ do
3:     repeat
4:         $\theta_{save} \leftarrow \theta_t$
5:         $\theta_t \leftarrow TrainWithEWC(\theta_t, \mathcal{T}_t, K)$
6:         if $TestingAccuracy(\theta_t, \mathcal{T}_t) < \tau$ then:
7:             $\theta_t \leftarrow ExpandNetwork(\theta_{save})$
8:     until $TestingAccuracy(\theta_t, \mathcal{T}_t) \geq \tau$
9: return $\theta_N$
```

parameters after expansion to the new network dimensions. Note that a summed Fisher Information value $\sum_{t=1}^{n-1} F_{\mathcal{T}_t,ii}$ of zero allows free movement of the weight i for learning the subsequent task. For weights which existed in the pre-expansion network, the values in these sums are carried over to the new network.

When testing network accuracy on a previously encountered task we mask all network weights which were not present in the network when the network was trained on that task. Our network model is also provided with weights specific to each task connecting the final hidden layer to the output layer. This is necessary to ensure that the network is able to learn multiple tasks even in cases where the same input may require a different label. As a trivial example, consider the case where task A involves recognizing odd digits in the MNIST data set and task B involves recognizing even digits. There are no compromise weights that could allow both tasks to be learned. The training of the output weights is not dependent upon the Fisher Information, nor is it subject to our variable learning rate algorithm. Note that in the experiments presented below, the non-expanding control networks also use a separate output layer for each task.

Results

An MLP architecture consisting of two hidden layers, each 20 nodes in size, was used in experiments comparing our expandable model to a fixed-size model[1]. Both models employed EWC, our variable learning rate from Equation 4, and task-specific weights and biases from the final hidden layer to the output layer. The models were sequentially trained on the same 100 permuted MNIST tasks. When the expandable model fails to meet the 90% accuracy threshold on the most recent task, the node count of each of its hidden layers doubles and its parameters are reset to their values prior to training on that task. The network then attempts training on the most recent task again, with the aid of the newly introduced, unrestricted, weights. This allows our expandable model to achieve both greater average accuracy across all tasks (Figure 1a) and better per-task accuracy (Figure 1b) compared to the fixed-size model. For these experiments, each network was trained for 10 epochs on each task, with a learning rate of 0.01 and $\lambda = 150$.

In generalizing our algorithm for use with convolutional neural networks (CNNs), we used a modified form of AlexNet [5], consisting of a series of convolutional and pooling layers followed by several dense layers which serve as a classifier mapping learned features to predictions. An expandable and fixed size network were each trained on incremental CIFAR 100 [9], a variation of the original CIFAR 100 dataset [4] in which the 100 image classes are divided into groups which comprise incrementally learnable tasks. Each task consists of images from 5 classes in the CIFAR 100 dataset, for a total of 20 tasks. The expandable network was initialized with 8 convolutions in the first layer and 2 dense layers consisting of 256 nodes each. On each expansion, 8 convolutions were added to the first convolutional layer (and other layers scaled appropriately to maintain their initial relative sizes compared to the first layer) and node counts of the hidden layers of the classifier were doubled. Again, both the fixed size and expandable models utilized EWC, our variable learning rate algorithm, and task-specific weights and biases from the final hidden layer to the output layer. As

[1]Source code: `https://github.com/jonesaj/Continual-Learning-Expandable-EWC`.

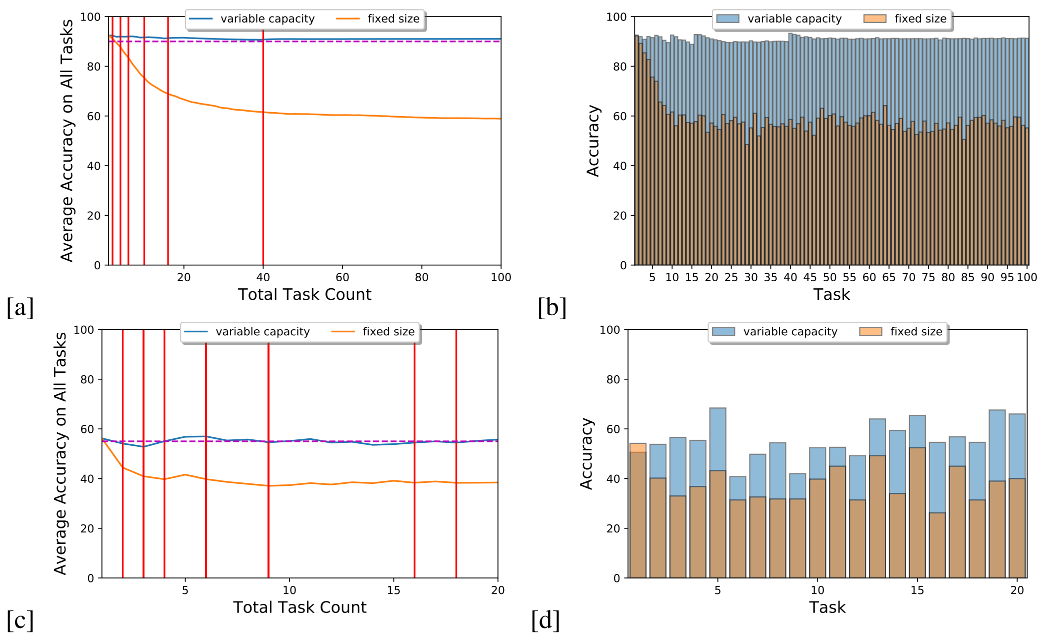

Figure 1: **[a]** MLP- Average accuracy on all tasks after training each permuted MNIST task. The purple dotted line marks the accuracy threshold (90%). Red lines mark tasks for which the network expanded in size once before learning to meet the accuracy threshold. **[b]** MLP- Accuracy on each permuted MNIST task in each network's final post-training state. **[c]** Modified AlexNet- Average accuracy on all tasks after training each incremental CIFAR 100 task. The purple dotted line marks the accuracy threshold (55%). Red lines mark tasks for which the network expanded in size one or more times before learning to meet the accuracy threshold. **[d]** Modified AlexNet- Accuracy on each incremental CIFAR 100 task in each network's final post-training state.

was the case with permuted MNIST, expansion allowed the network to attain both higher average (Figure 1c) and per-task (Figure 1d) accuracies for almost every task. For these experiments, each network was trained on each task for 10 epochs with a learning rate of 0.01 and $\lambda = 95,000$. The relative difficulty of incorporating new CIFAR 100 class prediction knowledge into a network varies greatly based on the classes from which data samples are drawn for any given incremental task. Therefore, unlike the network used in our permuted MNIST experiments, this network required more than one expansion before learning several tasks to meet the accuracy threshold on those tasks. Network expansion occurred as follows: 1 time before task 2, 3 more times before task 3, 1 more time before task 4, 6 more times before task 6, 9 more times before task 9, 1 more time before task 16, and 1 more time before task 18.

Future work

One weakness of the proposed algorithm is the need for a fixed performance threshold to determine when a network expansion is required. This may be impractical in cases where a different threshold would be appropriate for each task. Future work will explore the possibility of tracking increases in the EWC penalty term as a task-agnostic indicator that an expansion may be required.

Conclusions

We proposed an approach to continual learning that combines complementary features of elastic weight consolidation and progressive neural networks. We have shown that the resulting algorithm is able to adaptively resize to accommodate new tasks without compromising performance on previously learned tasks. Starting from a small network, the algorithm is able to handle a large number of tasks with relatively few expansions. The approach extends naturally to deep convolutional networks by expanding both convolutional and dense layers as necessary.

Acknowledgments

We gratefully acknowledge the support of NVIDIA through the provision of a Quadro P6000 GPU.

References

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

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] 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*, 114 13:3521–3526, 2017.

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Technical Report, University of Toronto*.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[6] Jeongtae Lee, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *CoRR*, abs/1708.01547, 2017.

[7] Sang-Woo Lee, Jin-Hwa Kim, JungWoo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*, 2017.

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

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017.

[10] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-cnn: A hierarchical deep convolutional neural network for incremental learning. 2018.

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

[12] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[13] S. Thrun. A lifelong learning perspective for mobile robot control. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 23–30 vol.1, Sep 1994.

[14] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[15] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.