Continual Learning with Lottery Tickets

Abhay Singh, Han Qiao, Horace He, Tianxing Jiang, and Xiuyu Li

Cornell University, Ithaca, NY
{as2626,hq45,hh498,tj258,xl289}@cornell.edu

Abstract. In this report we propose a novel training scheme to resist catastrophic forgetting, in which a model fails to continually learn and instead overfits to recently seen data. We repeatedly train, prune, and reset parameters to identify the 'winning tickets' in a model for each task, and combine an arbitrary number of these winning tickets in a single model to perform continual learning. Our method performs well, and successfully learns multiple new tasks with reasonable accuracy.

1 Introduction

Although machine learning models have exhibited human-level performance on individual tasks in recent years, these models are static in design and cannot adapt or expand their behavior over time. Such models restart their training process each time new data from the distribution is made available as the model must update. In our dynamic world, retraining in such a manner quickly becomes intractable in real-time scenarios.

Conversely, *continual learning* is the ability of a model to learn a sequence of well-defined tasks, with each task building off previous knowledge, all the while retaining adequate performance on earlier tasks [2]. Continual learning is necessary in the goal of artificial general intelligence, as an agent must be able to adapt to a constantly changing environment, a key component of human intelligence. Continual learning is also significant in the supervised learning scenario, where a model that fails to continually learn overfits to recently seen data – a phenomenon known as *catastrophic forgetting* [4] – and occurs if there is a drift in the input data distribution, or if the data was not uniformly shuffled. Resistance to catastrophic forgetting, such that learning on a new a task does not degrade performance on past tasks, is what we tackle in this report.

One such approach to resist catastrophic forgetting in the aid of continual learning is to employ parameter isolation: freezing a certain subset of learned model parameters after each task, and thereafter adding new heads/branches of parameters for new tasks [11], or even making a complete copy of the model for each new task [1]. This intuitively dedicates some subset of parameters to certain potential future tasks. Weight-based pruning techniques [5] are often applied to free redundant parameters and identify important parameters for each given

task.

The inspiration of our work lies in investigating and applying recently proposed work by Frankle & Carbin [3] to alleviate catastrophic forgetting. Their work determines an important subset of randomly initialized parameters–known as *lottery tickets*–that when trained in isolation, reach test accuracy comparable to the original network in a similar number of iterations. Specifically, they propose the *lottery ticket hypothesis*, that one can find sparse, trainable subnetworks within larger networks through iterative pruning. These 'winning tickets' found can be trained from their initial state with no drop in performance compared to the original much larger, unpruned network – even with over 90% of weights pruned [3].

We borrow from this technique, and propose a novel approach to resisting catastrophic forgetting. We draw inspiration from the lottery ticket hypothesis to develop a method that utilizes these 'winning tickets' to identify subsets of parameters in a network to be isolated for a certain set of tasks. In this report, we empirically demonstrate that our method maintains similar accuracy on old tasks after training the model on a series of new tasks, overcoming catastrophic forgetting to an extent.

2 Related work

2.1 Continual Learning Taxonomy

Continuous learning has been a widely studied topic and the methods proposed to address catastrophic forgetting by prior works can be classified into 3 types based on how task-specific information is stored and used throughout out the sequential learning process [7]: **Replay-based methods**, which include iCaRL [10] and GEM [12], aim at alleviating forgetting when learning a new task by replaying stored samples from previous tasks through approaches like rehearsal or constraint optimization on new tasks loss and previous task loss; **Regularization-based methods**, which includes Elastic Weight Consolidation (EWC) [6] and Learning without Forgetting (LwF) [8], propose extra regularization terms in the loss function to consolidate previous knowledge when learning on new data; **Parameter isolation-based methods** like PackNet [9] prevent forgetting by freezing the set of parameters learned after each previous task and growing new branches for new tasks [11,1]. Replay-based methods require the storage of the older training data and can be limited by scalability; Regularization-based methods only use data from the current task, but this can cause issues when data for new tasks belong to a dissimilar distribution different as compared to those of prior tasks [9,8]. Instead of adding any soft constraints, Parameter isolation-based methods apply the techniques of pruning and do not suffer from the above two drawbacks.

Our method falls under the scope of Parameter isolation-based methods and this report focuses on comparison other methods in this scope, specifically in a task incremental setting, where tasks are identified during evaluation. [13].

2.2 PackNet

Our work is most similar to that of PackNet [9], which is a Parameter isolation-based method employing iterative pruning and network re-training to prevent catastrophic forgetting when training on multiple different tasks sequentially. For each new task T_n, PackNet is trained while a set of parameters $\theta_1, \ldots, \theta_{n-1}$ derived from previous tasks are fixed. Then a fixed percentage of non-fixed parameters, usually the lowest 50% or 75%, are selected for removal from every convolutional and fully connected layer. The remaining parameters will be retrained for T_n and become fixed θ_n for the new task. This method ensures minimal drop in performance and minimal storage overhead, and adding even unrelated new tasks does not change performance on older tasks at all [9]. However, the method of pruning and parameter isolation employed by PackNet inherently limits the number of tasks that it can train on.

2.3 Lottery Ticket Hypothesis

Our work is inspired by the lottery ticket hypothesis [3], which proposed the following pruning method. It randomly initializes a over-parameterized network, and then repeatedly trains, prunes, and resets the network over n rounds, with each round pruning $p^{\frac{1}{n}}\%$ of the weights that survive the previous round. At the end, $p\%$ of the weights are pruned, and the other remaining weights are kept as 'winning tickets'.

Most importantly about these winning tickets is that when trained in isolation, they can reach the same or sometimes better test accuracy than the full model, even when over 99% of the parameters have been removed [3]. An interpretation is that the winning tickets found when training for a certain task are crucial for the performance, and we base our methodology off this interpretation such that each task has a unique winning ticket in a network, and in conjunction can prevent catastrophic forgetting.

3 Methodology

3.1 Split MNIST

We use Split MNIST to evaluate our approach. Following the procedure as detailed in [14], we split the MNIST dataset into 5 subsets/tasks, where each task is two-way classification between two consecutive digits; that is, the first task is the classification between the digit 0 and 1, the second task is the classification the between the digit 2 and 3, and so on.

3.2 Model

For a fair comparison, the same neural network architecture was used for all methods. This was a multi-layer perceptron with 2 hidden layers whose weights were shared between all tasks, and a task-specific linear layer attached at the end of the model for each of the five tasks. ReLU non-linearities were used in all hidden layers. The first and second layer output 300 and 100 neurons respectively, and the last layer outputs the number of classes, which is 10 in the Split MNIST setting.

3.3 Finding and Combining Winning Tickets

We now introduce our method. We begin with a single randomly initialized model $f(x;\theta)$ as detailed above with parameters θ_0. For each task, we obtain a winning ticket as a mask m of model parameters. This is done in the following way:

1. Train the network n epochs on the task updating weights θ_n
2. Magnitude prune $p\%$ of the weights, obtaining a mask m
3. Reset unpruned weights to their previous values in θ_0, creating a winning ticket $f(x; m \odot \theta_0)$

The above 3 steps are repeated ℓ times to obtain a final mask m_i for a task i. Doing so for each task obtains the necessary winning tickets.

We then train on each task sequentially, beginning with $f(x;\theta_0)$, as is the continual learning setup. We train on a task i given $f(x;\theta_{i-1})$ by only updating those weights that were identified as a winning ticket: $m_i \odot \theta_{i-1}$. This gives us new weights $\tilde{\theta}_{i-1}$. Finally we set the parameters as $\theta_i := m_i \odot \tilde{\theta}_{i-1} + (J - m_i) \odot \theta_{i-1}$ to give us $f(x;\theta_i)$, where J is the all-ones matrix.

4 Results

We test both our method and the PackNet [9] model on the Split MNIST setting.

For our method, we train for $n = 100$ epochs to identify a ticket, pruning by $p = 10\%$, and repeating so $\ell = 35$ times.

For PackNet, we used 75% weights magnitude pruning and trained the PackNet with all other parameters as default. The model is then trained and pruned on each of the five tasks, and the final model is evaluated on each task.

Table 1 shows the accuracies of the our method and PackNet on each split.

Split	Our Method	PackNet
[0,1]	85.3%	100%
[2,3]	87.3%	93.6%
[4,5]	95.2%	94.1%
[6,7]	90.9%	98.5%
[8,9]	89.9%	81.8%
Average	89.7%	93.0%

Table 1: Classification accuracy of both methods on each of the split tasks

5 Discussion

We achieve reasonable accuracy using our method. We achieve higher 'plasticity', which is the abilitiy to successfully integrating new tasks' information, although we suffer from worse 'stability', which is the ability to retain previous tasks' knowledge. Also, we distinguish between Task Incremental Learning, in which we know a priori which task you are performing on, and Class Incremental Learning, which is the opposite of Task-IL. While PackNet is doing well on Task-IL, our method is performing better in Class-IL setting.

References

1. Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. CoRR **abs/1611.06194** (2016), http://arxiv.org/abs/1611.06194
2. Chen, Z., Liu, B.: Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **10**(3), 1–145 (2016)
3. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Training pruned neural networks. CoRR **abs/1803.03635** (2018), http://arxiv.org/abs/1803.03635
4. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
5. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. CoRR **abs/1506.02626** (2015), http://arxiv.org/abs/1506.02626
6. Kirkpatrick, J., Pascanu, R., Rabinowitz, N.C., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. CoRR **abs/1612.00796** (2016), http://arxiv.org/abs/1612.00796
7. Lange, M.D., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: Continual learning: A comparative study on how to defy forgetting in classification tasks (2019)
8. Li, Z., Hoiem, D.: Learning without forgetting. CoRR **abs/1606.09282** (2016), http://arxiv.org/abs/1606.09282
9. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. CoRR **abs/1711.05769** (2017), http://arxiv.org/abs/1711.05769

10. Rebuffi, S., Kolesnikov, A., Lampert, C.H.: icarl: Incremental classifier and representation learning. CoRR **abs/1611.07725** (2016), http://arxiv.org/abs/1611.07725
11. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. CoRR **abs/1606.04671** (2016), http://arxiv.org/abs/1606.04671
12. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. CoRR **abs/1705.08690** (2017), http://arxiv.org/abs/1705.08690
13. van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. CoRR **abs/1904.07734** (2019), http://arxiv.org/abs/1904.07734
14. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3987–3995. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017), http://proceedings.mlr.press/v70/zenke17a.html