Overcoming catastrophic forgetting in Sparse Evolutionary Training

Abstract

Catastrophic forgetting in neural networks can be overcome by slowing down weight value evolution through Elastic Weight Consolidation (EWC); the weight values are moved into a smaller subspace. This research investigates what the effects of dynamic sparsity are on EWC and whether Sparse Evolutionary Training (SET) can reduce parameter count while still overcoming catastrophic forgetting. We show that the training operation of SET counteracts EWC and obstructs its ability to overcome catastrophic forgetting.

1 Introduction

In the field of artificial intelligence, once a network is trained to perform well on one task, e.g., clothing classification, it can hardly be trained to perform well on another task, e.g., vehicle recognition. This problem of *incremental learning* in deep neural networks, where the previously learned performance is withered when new tasks are being introduced, is called *catastrophic forgetting*.

There have been multiple methods to mitigate this problem, one of which is the *elastic weight consolidation (EWC)*, which remembers old tasks by selectively slowing down weight development on the weights important for those tasks [8].

One other method to mitigate catastrophic forgetting is using sparse-coding [7], which takes a subset of the connections from a fully connected neural network in a way that it only keeps the most important ones. A method of training sparse neural networks is called *sparse evolutionary training (SET)* [10, 11]. This SET algorithm has not yet been used in the context of catastrophic forgetting.

Training a model with Elastic Weight Consolidation (EWC) on multiple tasks reduces the possible parameter combinations of the model to the subset of parameter combinations with a low error for all tasks that the model is being trained on. This paper investigates how this reduction of possible parameter space influences the ability of sparse training methods to reduce the number of parameters. More specifically our research question is defined as follows:

- Are neural networks trained with EWC and SET on multiple problems able to achieve similar accuracy compared to fully connected neural networks trained with EWC?

2 Background

2.1 Elastic Weight Consolidation

Inspired by synaptic weight consolidation in biological brains, Kirkpatricka et al. proposed Elastic Weight Consolidation (EWC) in neural networks to overcome catastrophic forgetting [8]. Catastrophic forgetting occurs in continual learning when training a neural network on multiple tasks rather than only one. A model that is trained on multiple different tasks, tends to lose the information gained from the previously learned tasks as information from the newest task is incorporated. The weights of the network that are important to solve the problem A are adjusted in the learning process to fit the task of problem B, hence leading the model to forget the previous task.

Kirkpatrick et al. [8] introduce the EWC algorithm, which slows down future learning for weights that are important to previous tasks, hence retaining important information for the previous tasks, while enabling to learn a new task. The general idea of the algorithm is outlined in the schematic figure 1. The grey and cream colored areas represent the set of low error parameter values for the network regarding task A or task B respectively. When first training a model on task A, followed by training it on task B, the parameter would move according to the blue line if there is no penalty in the loss function. Because the gradient steps that update the weights in the learning process focus on minimizing the error of only task B, the network will forget the parameters that have a low error on task A. Considering an L2 regularization which penalizes every weight equally, the model would not be able move into the low error parameter space as the restrictions imposed are too strong, as indicated by the green arrow. Therefore, the authors propose the idea to only penalize the weights that are important to task A, moving into the region in which both tasks can be learned with low error, indicated by the red arrow. Hence EWC utilizes the fact that there exist many configurations of parameters that can achieve a low error for a given task, to move into the area of intersection in which the parameter configuration can achieve a low error for both or multiple task.

The derivation of the EWC penalization stems from viewing neural network training from a probabilistic perspective. Optimizing the parameters of a model θ is equivalent to finding the most probable values for them given the dataset $\mathcal{D}$. This is the conditional probability $p(\theta|\mathcal{D})$ which can be defined as follows by using bayes rule and taking the log:

$$\log(p(\theta|\mathcal{D})) = \log(p(\mathcal{D}|\theta)) + \log(p(\theta)) - \log(p(\mathcal{D})) \tag{1}$$

Assuming that we can split the data into two independent subsets A ($\mathcal{D}_A$) and B ($\mathcal{D}_B$), it is possible to rearrange the equation in the following way:

$$\log(p(\theta|\mathcal{D})) = \log(p(\mathcal{D}_B|\theta)) + \log(p(\theta|\mathcal{D}_A)) - \log(p(\mathcal{D}_B)) \tag{2}$$

Given this formulation, one can observe that the posterior probability is described by the loss function on the task B ($\log(p(\mathcal{D}_B|\theta))$) and the only information about task A is contained in the posterior distribution ($\log(p(\theta)|\mathcal{D}_A)$). To come to the EWC penalizing term, the authors approximate this posterior distribution ($\log(p(\theta)|\mathcal{D}_A)$), which is intractable to compute. The posterior is approximated as a Gaussian with the mean defined as the parameters θ^*_A and a diagonal precision defined by the diagonal of the Fisher information matrix F. The loss function for training on a problem B, having previously trained on a problem A, is then defined as follows:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{A,i}\right)^2 \tag{3}$$

for which $\mathcal{L}_B(\theta)$ is the loss function for task B and θ^*_A are the parameters with low error after training on the previous task A, while i loops over each parameter and λ defines how important the previous task A is. The fisher information matrix can be computed only by first order derivatives and more importantly, its diagonal is equivalent to the second order derivative of the loss near a minimum [8]. Because the second order derivative can be interpreted as the curvature of the loss surface, it can also give us information about the sensitivity of each of these weights with regards to the loss. A high curvature would mean that a small change in a parameter weight θ_k might result in a high increase in the loss. EWC hence tries to keep the values of the weights that have an high impact on the loss of the previous tasks by regularizing changes on these weights. Kirkpatrick et al. [8] have shown that EWC overcame the issue of catastrophic forgetting in multi-layer perceptrons and reinforcement learning applications. It was able to learn multiple tasks and retain a high accuracy for all of them, also being able to play multiple Atari games sequentially with high performance.

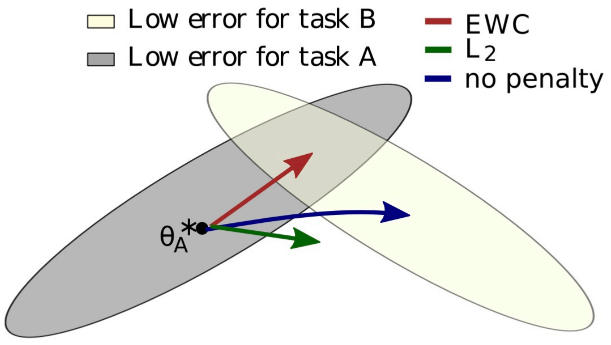

Figure 1: Schematic Representation of the subsets of the parameter space θ of a model that contain parameters for a low error for task A and B respectively. θ^*_A represents the parameters obtained after training a model on Task A and achieving low error. Courtesy of [8]

2.2 Sparse Evolutionary Training

Basic artificial neural networks contain fully connected layers with each neuron $n \in N_l$ in layer l being connected to every node $n \in \{N_{l-1}, N_{l+1}\}$. It has been shown that many weights in these fully connected models are near-zero and can be removed to decrease the required computing power for training while maintaining performance accuracy [3, 9, 6, 4, 1]. A fully connected neural network with at least one missing connection is a sparse neural network. One of the methods to introduce sparsity in neural networks is Sparse Evolutionary Training (SET) [10, 11]. The process of training a model with SET is shown in Figure 2. Different from former approaches, SET initializes a sparse network rather than to start with a dense network and then prunes the weights according to a salience criteria. The sparse network is initialized with a sparsity level ϵ and then the ζ number of smallest weights are removed, followed by an addition of ζ random weights. This procedure of removing and random addition is done in the training procedure after the weights have been updated and continues until the training procedure ends. The procedure is outlined in Algorithm1. Previous research has shown that SET is able to reduce the parameter count of models trained on F-MNIST [13] by 96% [12].

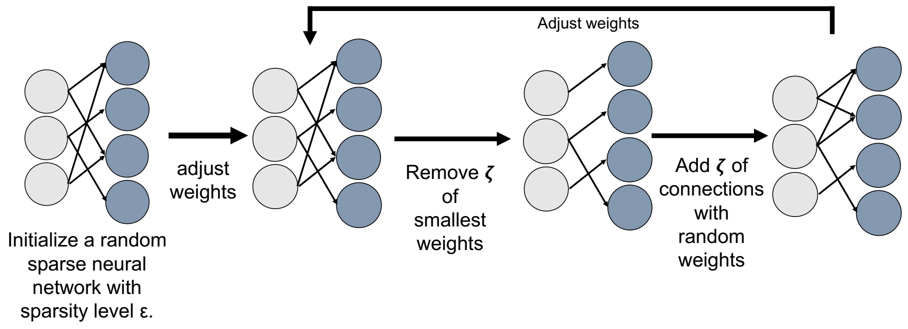
Adjust weights
Initialize a random sparse neural network with sparsity level ε.
adjust weights
Remove ζ of smallest weights
Add ζ of connections with random weights

Figure 2: The training cycle in sparse evolutionary training

3 Methodology

To answer our research question, we will compare the accuracy of two models that are sequentially trained and tested on eight different tasks. The first model will only be trained by applying the EWC algorithm, while the second model will include both the EWC and the SET algorithm. The sequential training and testing is done by first training and testing a model on task A in the traditional way, followed by training the same model on task B and testing it on both task A and B. We will also compare these models at different sparsity levels, to get a more holistic view. The classification tasks will revolve around the classification of the MNIST dataset [2] similar to the approach in the paper by Kirkpatrick et al. [8]. To ensure that each task is of equal difficulty but still needs a different solution to solve, the tasks are generated by creating a fixed random permutation on all the images of the MNIST dataset. This procedure is commonly used in continual learning [5].

In the fully connected model, any neuron $\mathbf{h}^k$ in layer k is connected to all arbitrary number of neurons $\mathbf{h}^{k-1}$ in the previous layer. In SET, any neuron is connected to an arbitrary number of neurons in the previous layer. The network in SET is initialized as an Erdös-Rényi random graph, in which the probability of a connection between the neurons h_i^k and h_j^{k-1} is given by

$$p\left(W_{ij}^k\right) = \frac{\varepsilon\left(n^k + n^{k-1}\right)}{n^k n^{k-1}} \tag{4}$$

where n is the number of neurons in layer k and $\varepsilon \in \mathbf{R}^+$is a parameter of SET defining the static sparsity level.

The accuracy values of the models that are trained on one task and the model that is trained on both tasks will be compared at different levels of sparsity for all models. From this comparison, the effect of sparsity on the generalization capability of the model on two tasks will be visible.

4 Experiments

The model used for this research is a neural network with dimensions $(784, 1000, 1000, 1000, 10)$ and activations (ReLU, ReLU, ReLU, Softmax). The number of training epochs is 3 with batchsize 100, dropout probability of 0.2, learning rate 0.1 and epsilon (ε) 40. We train on a total of eight consecutive tasks. The implementation can be found at the following GitHub Repository.

Algorithm 1 Toplogical Evolution with AccSET

```
 1: Require: $\mathbf{X}, \mathbf{M}, \eta$
 2: $c \leftarrow \eta\|\mathbf{X}\|_0$
 3: $c_p \leftarrow c/2$ $c_n \leftarrow c/2$
 4: $\tilde{\mathbf{X}}^p \leftarrow get\_c_p th\_smallest\_positive(\mathbf{X})$
 5: $\tilde{\mathbf{X}}^n \leftarrow get\_c_n th\_smallest\_negative(\mathbf{X})$
 6: Drop least important connections
 7: $\mathbf{M}^j \leftarrow \mathbf{M}^j - \mathbb{1}(\mathbf{X}^j < \tilde{\mathbf{X}}^p_\vee(\mathbf{X}^j > \tilde{\mathbf{X}}^n))$
 8: Grow new connections:
 9: Generate number of random integers according to $A(P_e)_{y^l,\lambda^l_e,k}$
10: $\mathbf{M}^j \leftarrow \mathbf{M}^j + \mathbb{1}(j == x) \wedge (\mathbf{X}^j == 0)$
11: $\mathbf{X} \leftarrow \mathbf{X} \odot \mathbf{M}$
```

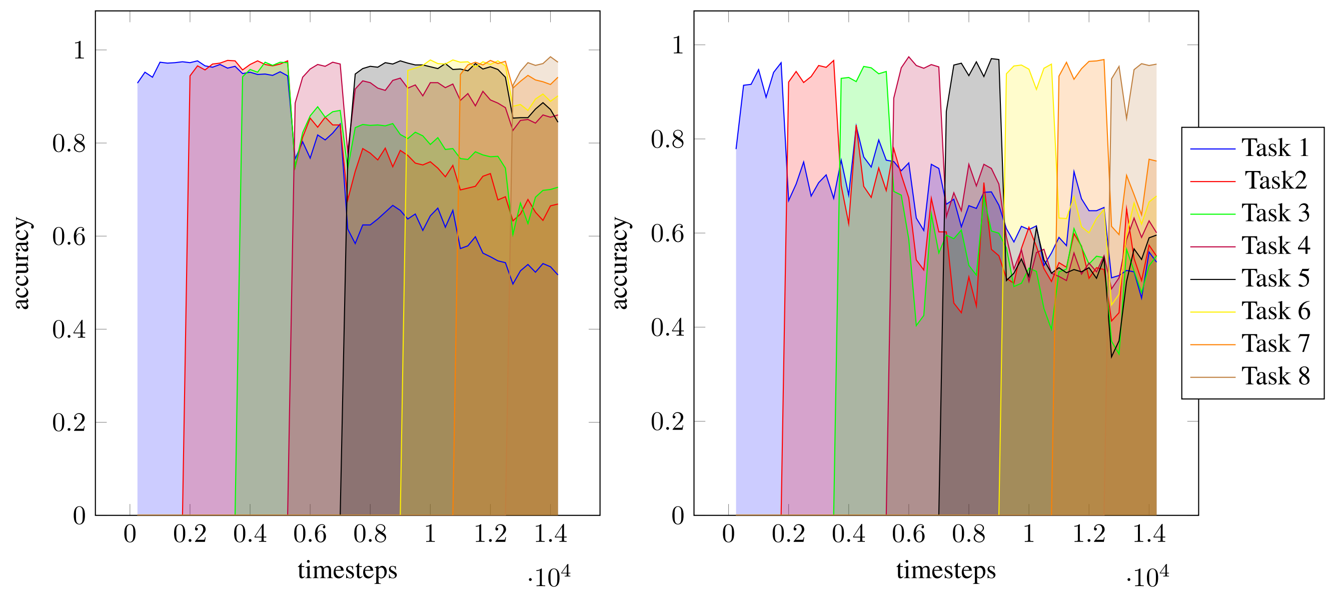

Figure 3: Accuracy of the various tasks for a fully connected MLP (left) and SET MLP with $\zeta = 0.05$ (right).

4.1 Results

Figure 3 showcases the accuracies of the fully connected model (FC) trained with EWC (left side) and the sparse model trained with EWC and SET (right side). It details the change in accuracy over time and over the sequential learning procedure for the eight different tasks. One can observe that the FC model generally seems to have more stable accuracy and is able to hold the high accuracy of previous tasks longer than its sparse counterpart. For example, the accuracy of task 1 in the FC model retains its high accuracy value even after training for task 2 and 3 while also maintaining high accuracy's for these tasks, but when learning for task 4 the accuracy of task 1 drops On the other hand the accuracy of task 1 for the sparse counterpart drops instantaneously when the model is trained for task 2. Similar observations can be made for the other tasks as well.

Table 1 presents the accuracies for each task after all tasks have been trained. The most recent tasks have the highest accuracy for both the FC and the sparse SET model, with the tendency that the accuracy decreases as we approach the first task of the sequential training procedure. In general the FC model outperforms the SET models on all tasks except the first one. Comparing between the performance of the sparse models, the models with higher sparsity level ($\zeta = 0.2, \zeta = 0.1$) perform better at the most recent tasks (from task 5 to 8) but worse in the older tasks (from task 1 to 4) than the low sparsity model ($\zeta = 0.05$). This is not the case for task 3, in which the model with $\zeta = 0.05$ performs worse than the model with $\zeta = 0.1$. For task 1 the low sparsity model with $\zeta = 0.05$ performs best out of all models including the FC model.

Method	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
FC	0.5175	**0.669**	**0.705**	**0.860**	**0.845**	**0.901**	**0.941**	**0.974**
SET(ζ = 0.2)	0.502	0.542	0.4375	0.555	0.544	0.634	0.739	0.959
SET(ζ = 0.1)	0.538	0.551	0.552	0.601	0.596	0.679	0.753	0.959
SET(ζ = 0.05)	**0.603**	0.642	0.491	0.692	0.451	0.612	0.726	0.925

Table 1: Accuracy of the fully connected (FC) and sparsely trained model (SET) on the tasks after having trained all tasks consecutively

5 Discussion & Future Work

5.1 Discussion

From the results in the previous section we have seen that the models trained with SET perform generally worse than the fully connected (FC) counterparts. The reason on why this is happening is not fully clear to us yet, however we think that the two algorithms, EWC and SET, are working against each other. This could happen due to the fact that SET removes connections that are not so important to the current task, however they might be important to previous tasks. Therefore, the *knowledge*, i.e., weights, accumulated with EWC might get removed with SET.

In the parameter space of a task, the SET algorithm is trying to remove connections that have low impact, however if multiple tasks' parameter space is intersected, the number of not-so-important features drop significantly. Thus any amount of weight pruning can have an negative impact on the accuracy. This can also be seen in the result of the lowest sparsity level, where even with $\zeta = 0.05$ accuracies for the different tasks are significantly lower than for the fully connected model. Adding the fact that the model with highest zeta value $\zeta = 0.2$ under-performs the model with middle zeta value $\zeta = 0.1$ for every tasks, we might also indicate that adding sparsity above a certain threshold only reduces accuracy.

When applying EWC, only the last hidden layers and output layers use the same weights for multiple tasks, while the input layers are allocated into separate parts for each task [8]. This means that the different tasks do not share the weights in the input layer, yielding a network that has a separated representation for each task in the input layers. The removal and random addition procedure of SET might change these structures in the input layer resulting in forgetting the older tasks representations. This hypothesis is in line with the results which show that higher zeta values are worse at remembering older tasks than lower zeta values.

Additionally, SET removes near zero weights as they have no significant impact on the model output and on the loss. On the other hand EWC tries not to change the weights for which a small change in value would lead to a big change in the loss function. Assuming that changing near zero weights has a greater impact on the loss function due to the higher ratio of change, EWC tries to protect these near zero weights from change while SET removes them. This might be another explanation on the fact that SET seemingly cancels the effects of EWC to a certain extend.

The fact that for $\zeta = 0.05$ sparsity level we obtained the highest accuracy only for task 1 (see table 1), might suggests that the results are highly dependent on the initial permutation, i.e., on the task itself.

5.2 Future Work

As for further work, we ought to run the experiment for multiple iterations, with different tasks, so to conclude that there is no correlation between the low sparsity level ($\zeta = 0.05$) and the high accuracy obtained for the first task. Also the creation of a baseline fully connected model without SET and without EWC will give more insights about the interplay of sparsity and catastrophic forgetting. Additionally, a more extensive hyper-parameter tuning is to be desired to ensure better conditions and performance for the comparison and the proper functioning of the SET and EWC algorithm.

Since sparse-coding has been used as a mitigation against catastrophic forgetting [7], another interesting question to look at, and try to answer in future research, could be to investigate whether EWC improves catastrophic forgetting in SET compared to SET trained without EWC.

6 Conclusion

This research aimed to explore whether the regularization of EWC could be combined with the dynamic sparsification of SET to yield better accuracy performance. To answer our research question, the model trained on both algorithms, EWC and SET, does not achieve a similar accuracy to the fully connected model. In our case it only performs better for the first task, at sparsity level of $\zeta = 0.05$. For all other tasks, this model fails to achieve similar accuracy. This can be due to the fact that SET removes connections that appear unimportant to the current task, but are required for previously learnt tasks.

References

[1] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, October 2019. ISSN 1557-9956. doi: 10.1109/TC.2019.2914438. Conference Name: IEEE Transactions on Computers.

[2] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting Parameters in Deep Learning. *arXiv:1306.0543 [cs, stat]*, October 2014. URL `http://arxiv.org/abs/1306.0543`. arXiv: 1306.0543.

[4] Tim Dettmers and Luke Zettlemoyer. Sparse Networks from Scratch: Faster Training without Losing Performance. *arXiv:1907.04840 [cs, stat]*, August 2019. URL `http://arxiv.org/abs/1907.04840`. arXiv: 1907.04840.

[5] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013. URL `https://arxiv.org/abs/1312.6211`.

[6] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both Weights and Connections for Efficient Neural Networks. *arXiv:1506.02626 [cs]*, October 2015. URL `http://arxiv.org/abs/1506.02626`. arXiv: 1506.02626.

[7] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL `https://ojs.aaai.org/index.php/AAAI/article/view/11651`.

[8] James Kirkpatrick, Razvan Pascanu, Neil 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*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1611835114`.

[9] Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990. URL `https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html`.

[10] D. C Mocanu, A Liotta, and G Exarchakos. *Network computations in artificial intelligence*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, 2017. ISBN: 9789038643052 OCLC: 993672622.

[11] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1), June 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-04316-3. URL `https://www.nature.com/articles/s41467-018-04316-3`.

[12] Vincent van Engers. Towards understanding and modelling sparse training algorithms at extreme sparsity regime, June 2021. URL `http://essay.utwente.nl/86887/`.

[13] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.