A Comparative Study of Continual Backpropagation

Jacopo Silvestrin[1,2](✉), Francisco S. Melo[1,2], and Manuel Lopes[1,2]

[1] Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal
jacopo.silvestrin@tecnico.ulisboa.pt
[2] INESC-ID, Lisbon, Portugal

Abstract. In continual learning, artificial agents need to be able to learn in a sequential fashion, adapting to new data and maintaining relevant information about the past. A growing body of research shows that neural networks tend to overfit the first samples they encounter, making it challenging to adapt to future tasks. Algorithms that periodically reset part of their network parameters have been shown to maintain the ability to continuously learn from new data. However, these methods are prone to catastrophic forgetting as they do not keep information about the past. In this paper, we provide a comparative study on continual adaptation and catastrophic forgetting capabilities of several variations of the Continual Backprop algorithm, investigating the impact of increasing the network capacity and choosing which parameters to reset based on their Fisher information.

Keywords: Continual learning · Network plasticity · Catastrophic forgetting

1 Introduction

In the last decade, deep learning methods have become the state-of-the-art in many machine learning applications, ranging from natural language processing [3] to reinforcement learning [12] and supervised learning [8]. However, these algorithms fail to emulate the process of human learning, which is characterized by the ability of learning sequentially from a non-stationary world and generalizing previous knowledge to new scenarios. Many real world applications, such as recommendation systems and robotic agents deployed under non-stationary environment conditions, require the agent to adapt to shifting distributions of the data.

Continual learning is the area of research that studies machine learning methods that are able to learn new tasks sequentially while avoiding to forget the old ones (catastrophic forgetting). Leveraging the strengths of deep learning in a continual learning setting is an attractive perspective. However, neural networks have been shown to be prone to catastrophic forgetting [7], as they are trained to optimize only the present objective, ignoring past distributions of the data.

At the same time, they over-train on the first samples they encounter, losing plasticity over time and failing to adapt to new data [1,10].

In this paper, we perform a comparative study on the continual cackprop algorithm, proposed in [4] as a variation of the popular cack-propagation. This method preserves network plasticity by periodically selecting a set of hidden units to reset, injecting randomness back into the network parameters. We test it against standard back-propagation in a continual adaptation task, reproducing an experiment from the original article. Moreover, we test the limitations of this method by evaluating its robustness to catastrophic forgetting and compare it against alternative algorithms that use the Fisher information as metric to select the hidden units to be re-initialized and increase network capacity to add learning capability to train the new tasks.

2 State of the Art

Continual learning aims to achieve two major goals: learning from a dynamic data distribution and memorizing useful information for future tasks. In contrast with common machine learning approaches that specializes in one narrow task, a continual learner must be able to learn from a non-stationary world in a sequential manner, keep useful information about its lifetime and re-apply previously learned skills when needed.

Memorizing useful information is challenging for current deep neural networks as their parameters are constantly updated to adapt to the current data distribution, reducing the ability to capture the old ones. Works such as [7] try to avoid catastrophic forgetting by preserving the weights that contribute the most to the output according to their Fisher Information during the network update. Modular approaches such as [11,13] address this problem by adding new capacity to the network while protecting old parameters. However, an effective and scalable solution to catastrophic forgetting remains an open problem [4].

More works have focused on the continual adaptation part of the problem. Meta-learning is used in [2,5] to adapt to non-stationary environments in few training steps. This assumes knowledge of the task distribution during meta-training and it might be a strong assumption in many real-world applications. In [14], the authors use a latent variable to encode the non-stationarity of a reinforcement learning environment and predict its future changes. This works well as long as the new tasks are predictable (e.g. there is an underlying structure or the change is smooth), but fails when the dynamics change abruptly in an unpredictable manner.

Finally, continuous adaptation in deep learning is faced with the challenge of loss of plasticity over time. In fact, many works such as [4,10] show that neural networks that learn tasks sequentially tend to overfit the first samples they encounter and over time lose the ability to converge to good solutions.

In the next section, we describe in detail a state-of-the-art method that addresses the problem of network plasticity for standard feed-forward neural networks. We then define an alternative Fisher utility metric and discuss the possible benefits of dynamically increasing the network size to give the neural network plasticity to adapt and preserve information.

3 Continual Back-Propagation

3.1 Stochastic Gradient Descent with Persistent Randomness

The method we study in this work comes from [4], where the authors show the loss of plasticity of a standard feed-forward network using gradient descent and propose a solution that periodically re-initializes the weights of the hidden units with lower utility.

When a hidden unit is re-initialized, its outgoing weights are set to zero to avoid compromising the output of the network, and a counter is started that keeps that hidden unit from being replaced for the next m iterations of the algorithm.

The utility metric proposed to measure the contribution for the $i-th$ hidden unit in layer l at time-step t is defined as

$$v_{l,i,t} = \frac{|h_{l,i,t} - \hat{f}_{l,i,t}| \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|}{\sum_{j=1}^{n_{l-1}} |w_{l-1,j,i,t}|}. \tag{1}$$

where $h_{l,i,t}$ is the output of the $i-th$ feature in layer l at time t, $w_{l,i,k,t}$ is the weight connecting the $i-th$ unit in layer l to the $k-th$ unit in layer $l+1$ at time t, and n_{l+1} is the number of units at layer $l+1$. Finally

$$\hat{f}_{l,i,t} = \frac{f_{l,i,t}}{1-\eta^{a_{l,i,t}}} \tag{2}$$

and

$$f_{l,i,t} = (1-\eta)h_{l,i,t} + \eta f_{l,i,t-1} \tag{3}$$

where η is the decay rate of the moving average and $a_{l,i,t}$ measures the age (i.e. the number of steps from its last initialization) of the $i-th$ hidden unit of layer l layer at time-step t.

The utility in Eq. 1 gets updated each time a new training step gets done with a moving average

$$u_{l,i,t} = (1-\eta)v_{l,i,t} + \eta u_{l,i,t-1} \tag{4}$$

and finally gives the updated utility to be used in practice

$$\hat{u}_{l,i,t} = \frac{u_{l,i,t}}{1-\eta^{a_{l,i,t}}}. \tag{5}$$

The intuition behind the utility metric in (1) is that low values of the product between the unit activation $h_{l,i}$ and its outgoing weights contribute less to successive layers and therefore correspond to a low utility. Moreover, high magnitude of the input weights causes the hidden unit to be saturated and lose learning capability, further reducing its usefulness.

The pseudo-code for the Continual Backprop algorithm can be found in Algorithm 1. Notice that an age threshold m is used to protect the newly initialized hidden units from being selected again in the next iteration of the algorithm.

Algorithm 1. Continual Backprop Algorithm [4].

```
**Set**: step-size $\alpha$, replacement rate $\tau$, decay rate $\eta$ and maturity threshold $m$. We used $\alpha = 10^{-4}$, $\tau = 10^{-4}$, $m = 100$.
**Initialize** weights $\mathbf{w}_l$ for $l = 0, .., L$.
**Initialize** utilities $\mathbf{u}_l$, average feature activations $\mathbf{f}_l$ and ages $\mathbf{a}_l$ for $l = 0, .., L$.
**for** each input $\mathbf{x}_t$ **do**
    Pass input through the network and get output $\hat{\mathbf{y}}_t$.
    Compute loss depending on application. E.g. $\mathcal{L}(\mathbf{x}_t, \hat{\mathbf{y}}_t) = (\hat{\mathbf{y}}_t - \mathbf{y}_t)^2$.
    Update weights with stochastic gradient descent.
    **for** layer $l \in \{1, L\}$ **do**
        Update age: $a_l$ += 1.
        Update feature utility using Eq. 1, 2 and 3.
        Select $\tau n_l$ features with age greater than $m$ and smallest utility. Let their indices be $\mathbf{r}$.
        Re-initialize input weights $\mathbf{w}_{l-1,r,:,t}$.
        Set output weights $\mathbf{w}_{l,:,r,t}$ to zero.
        Set $\mathbf{u}_{l,r,t}$, $\mathbf{f}_{l,r,t}$, and $\mathbf{a}_{l,r,t}$ to zero.
    **end for**
**end for**
```

3.2 Fisher Information as Utility Metric

The Fisher Information Matrix (FIM) of a neural network $f_\theta(\mathbf{x})$ measures the amount of information that an observable output carries about the network parameters θ.

The FIM of a neural network $f_\theta(\mathbf{x})$ is defined as [6,9]

$$F_\theta = \mathbf{E}[\nabla_\theta \log \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y}) \nabla_\theta \log \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})^T] \qquad (6)$$

where $\mathcal{L}$ is a loss function, $\mathbf{x}$ is the input, and $\mathbf{y}$ is the loss target. The expectation is taken with respect to the input-output pairs.

In this work, we estimate the FIM using an exponential moving average of the sample estimate

$$\hat{F}_\theta = \nabla_\theta \log \mathcal{L}(f_\theta(\mathbf{x}_t), \mathbf{y}_t) \nabla_\theta \log \mathcal{L}(f_\theta(\mathbf{x}_t), \mathbf{y}_t)^T \qquad (7)$$

The Fisher Information is an attractive metric to select which hidden unit to re-initialize in the context of Continual Back-propagation as it measures the contribution of each parameter of the network to its output. This allows to maintain the plasticity of the network while not degrading its output. Moreover, in [7], it is shown how protecting the weights with higher Fisher Information is useful to avoid catastrophic forgetting, which is an important property for a continual learner. For these reasons we developed a variation of the method described in Sect. 3.1 that selectively resets the hidden units with the lowest Fisher Information.

3.3 Dynamically Growing the Network

Dynamically adding capacity can be a way to inject randomicity back in the neural network while preserving the current learned function. Initializing the outgoing weights of the new hidden unit to zero preserves the integrity of the network, while the augmented capacity can help adapt to the new tasks. We are interested in verifying if this approach can improve memory retention of the network while allowing to reach good solutions to new tasks. For this reason in the following section, we will include methods that increase the network size in our analysis.

4 Experiments

In this work, we evaluate variations of Continual Backprop along two axis: continuous adaptation and memory retention.

4.1 The BitFlipping Problem

We perform our tests solving the BitFlipping problem [4], a regression problem with a moving target. The target is the output of a neural network with one hidden layer of 100 neurons. The input at time t is a binary vector $\mathbf{x}_t \in \{0,1\}^m$. Periodically, every T time-steps, a random bit among the first f gets selected and its value flipped. The value of the first f bits (the *flipping bits*) does not change at other times. The other $m - f$ bits are uniformly sampled at each timestep. We use $m = 20$, $f = 15$ and $T = 1e4$.

In the target network, the weights are randomly sampled from $U\{0,1\}$ and the activation function is a Linear Threshold Unit (LTU) with threshold parameter $\beta = 0.7$. The goal is to train a neural network to perform regression on the output of the BitFlipping. The learning networks have the same depth as the target but have considerably fewer hidden units. This is done to allow the target to have higher complexity and therefore forcing the learning networks to continuously adapt to the moving target. We use learning networks with 5 neurons in the hidden layer.

4.2 Methods

In the following experiments, we evaluate the following methods.

– *Standard Back-prop (Bp)*. Benchmark with standard back-propagation.
– *Continual Backprop (CBp)*. Method presented in Sect. 3.1. Here the utility of the hidden units is computed according to Eq. 1 and the reset period is $1e4$ timesteps.
– *FBp*. A variation of CBp. This method re-initializes the hidden unit with the lowest summed Fisher Information of its weights with period $1e4$ timesteps.
– *RBp*. A variation of CBp. This method re-initializes a random hidden unit with period $1e4$ timesteps.
– *GrowingBp*. A method that restores network plasticity by adding a hidden unit to the network. The hidden unit is added every $1e4$ time-steps.
– *GrowingCBp*. A variation of CBp, where in addition to resetting the hidden unit we also add a new one to boost network capacity with period $1e4$.
– *GrowingFBp*. A variation of FBp, where in addition to resetting the hidden unit we also add a new one to boost network capacity with period $1e4$.
– *BigCBp*. A bigger network using CBp with dimension reached by GrowingCBp and Growing FBp at the last iteration.

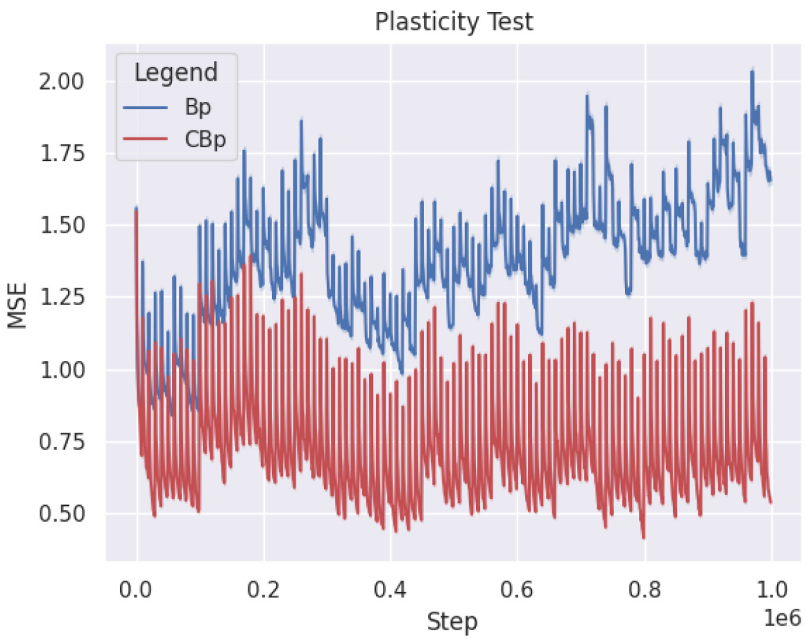

Fig. 1. Forget and Learn test. Mean squared error obtained by CBp and FBp in the BitFlipping regression problem with $m = 20$ and $f = 15$. Each plot is averaged across 30 runs. The timesteps are averaged into bins of 1000 samples, the target changes with period of 10k steps. We train for 1M steps

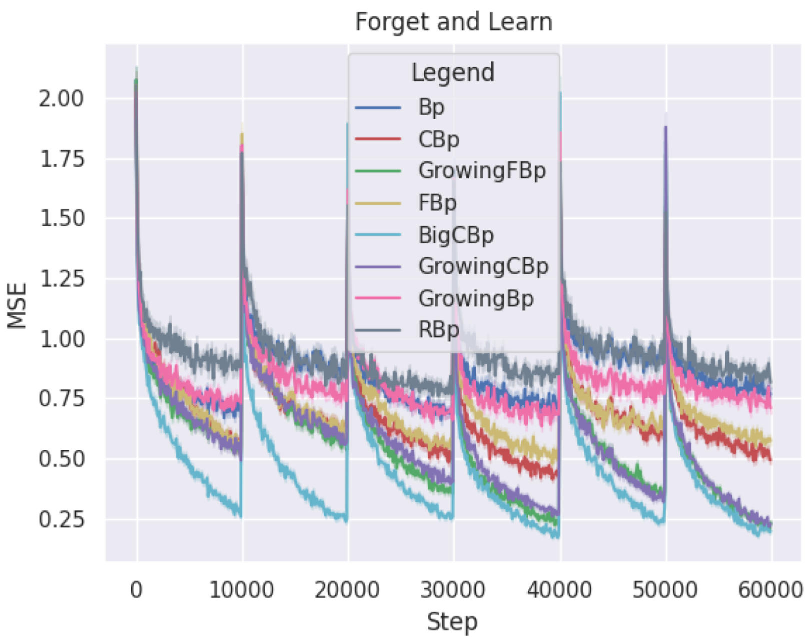

Fig. 2. Forget and Learn test. Mean squared error obtained by each method in the BitFlipping regression problem with $m = 20$ and $f = 15$. The shaded area around the plot is the standard error of the mean. Each plot is averaged across 30 runs. The timesteps are averaged into bins of 100 samples, the target changes with period of 100k steps

4.3 Forget and Learn Test

The first test was done to evaluate the continuous adaptation capabilities of CBp and verify the loss of plasticity of standard Bp. This experiment is reproduced directly from [4] and our results are consistent with the ones in the original article. We can see from the mean squared errors reported in Fig. 1 that Bp loses plasticity over time and is no longer able to converge to good solutions. Moreover, we notice that CBp outperforms standard Bp in the first iterations, we think this is because by re-initializing selected features the network is able to get out of local minima and to converge to a better solution.

The second test was done to evaluate the continuous adaptation capabilities of each method and if they would keep some useful information about previous tasks, allowing for faster learning and convergence to better solutions. Figure 2 shows the results for the considered methods. The learners here were trained in the BitFlipping problem for three different targets, at the end of this first phase, a second training phase was done on the already encountered targets.

The first thing we notice from the plot is that as expected the added capacity strictly improves the solutions, obtaining lower mean squared error values, although it doesn't seem to make a difference in terms of speed of convergence

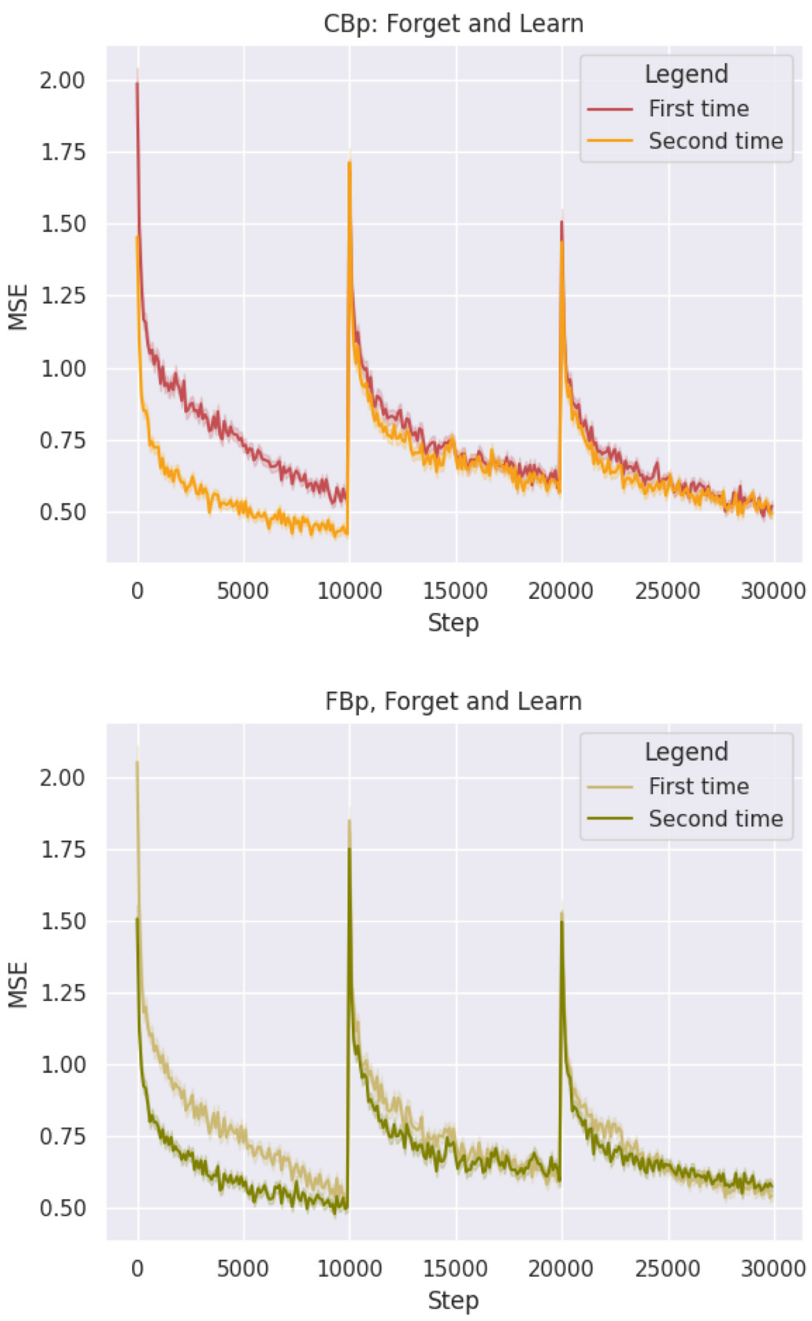

Fig. 3. Forget and Learn test. Mean squared error obtained by the CBp and FBp methods in the first and second time the same task is encountered. The shaded area around the plot is the standard error of the mean. The results are obtained from the three tasks in Fig. 2

when it encounters old tasks. Adding capacity therefore does not seem to allow the network to retain useful information about older tasks, since each 10^4 steps the MSE raises sharply. The method we studied, CBp, slightly outperforms our proposed method FBp. Differently from what expected, the FBp method does not show to have retained any relevant information about the old tasks. We were expecting this method to perform better right away in older tasks due to the preservation of the parameters containing the most amount of information.

In Fig. 3 we can observe in detail the average MSE obtained by CBp and FBp while training on the same task the first and the second time. We can see that both methods don't get significant benefits from having already encountered a task.

4.4 Memory Retention Test

This test was done to evaluate the performance of the methods in tasks recently encountered after training is stopped. Figure 4 shows the results. We notice that no method is able to avoid catastrophinc forgetting. In particular, protecting the

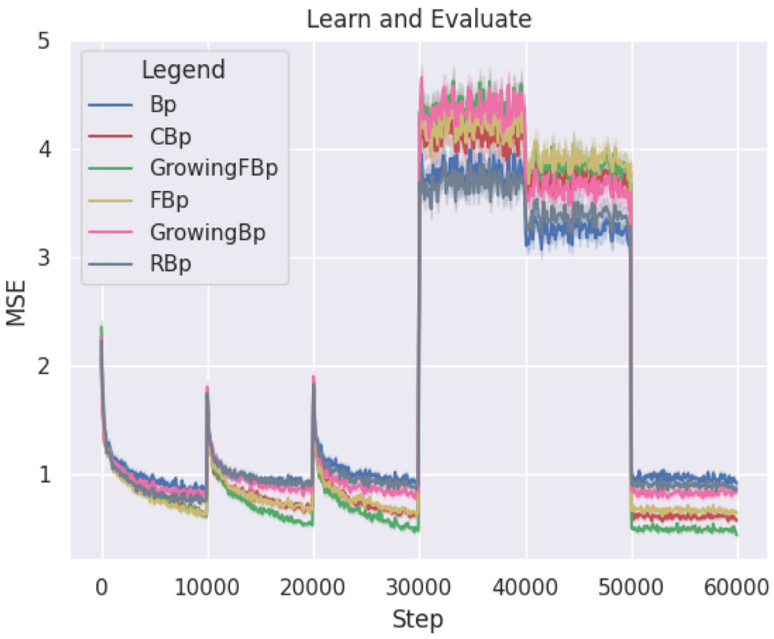

Fig. 4. Memory Retention test. Mean squared error obtained by training the methods for the first three targets, stopping training and evaluating on each of the encountered targets. The shaded area around the plot is the standard error of the mean. Each plot is averaged across 30 runs. The timesteps are averaged into bins of 100 samples, the target changes with period of 10k steps

weights with high Fisher Information in FBp did not lead to any benefit in this test, which is somewhat surprising.

It is interesting to notice that the methods that use standard Back-propagation, while losing plasticity over time, are the ones that forget the less among all the considered approaches.

5 Conclusions

In this paper we conducted a study on the Continual Backprop algorithm, evaluating its effectiveness in continual adaptation and ability to retain information about previous tasks. We benchmark the method against standard Back-propagation and simple variations of the CBp method. We show that Continual Backprop outperforms methods based on standard Back-propagation in continuous adaptation tasks and that changing the selection metric using an approach based on the Fisher Information of the network parameters, while reaching state-of-the-art performances in continual adaptation, does not provide robustness to catastrophic forgetting. Moreover, dynamically adding capacity to the network doesn't seem to allow it to retain useful information to re-learn previously observed tasks.

Acknowledgements. This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 - DOI: 10.54499/UIDB/50021/2020 and by the project RELEvaNT, ref. PTDC/CCI-COM/5060/2021.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abbas, Z., Zhao, R., Modayil, J., White, A., Machado, M.C.: Loss of plasticity in continual deep reinforcement learning. In: Conference on Lifelong Learning Agents, pp. 620–636. PMLR (2023)
2. Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., Abbeel, P.: Continuous adaptation via meta-learning in nonstationary and competitive environments. arXiv preprint arXiv:1710.03641 (2017)
3. Brown, T., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)
4. Dohare, S., Sutton, R.S., Mahmood, A.R.: Continual backprop: stochastic gradient descent with persistent randomness. arXiv preprint arXiv:2108.06325 (2021)
5. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. PMLR (2017)
6. Karakida, R., Akaho, S., Amari, S.I.: Universal statistics of fisher information in deep neural networks: mean field approach. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1032–1041. PMLR (2019)

7. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. **114**(13), 3521–3526 (2017)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
9. Liao, Z., Drummond, T., Reid, I., Carneiro, G.: Approximate fisher information matrix to characterize the training of deep neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **42**(1), 15–26 (2018)
10. Nikishin, E., Schwarzer, M., DOro, P., Bacon, P.L., Courville, A.: The primacy bias in deep reinforcement learning. In: International Conference on Machine Learning, pp. 16828–16847. PMLR (2022)
11. Rusu, A.A., et al.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
12. Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
13. Terekhov, A.V., Montone, G., O'Regan, J.K.: Knowledge transfer in deep block-modular neural networks. In: Wilson, S.P., Verschure, P.F.M.J., Mura, A., Prescott, T.J. (eds.) LIVINGMACHINES 2015. LNCS (LNAI), vol. 9222, pp. 268–279. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22979-9_27
14. Xie, A., Harrison, J., Finn, C.: Deep reinforcement learning amidst lifelong non-stationarity. arXiv preprint arXiv:2006.10701 (2020)