Powerpropagation: A sparsity inducing weight reparameterisation

Anonymous Author(s)
Affiliation
Address
email

Abstract

The training of sparse neural networks is becoming an increasingly important tool for reducing the computational footprint of models at training and evaluation, as well enabling the effective scaling up of models. Whereas much work over the years has been dedicated to specialized pruning techniques, little attention has been paid to the inherent effect of gradient based training on model sparsity. In this work, we introduce Powerpropagation, a new weight-parameterisation for neural networks that leads to *inherently sparse* models. Exploiting the behaviour of gradient descent, our method gives rise to weight updates exhibiting a "rich get richer" dynamic, leaving low-magnitude parameters largely unaffected by learning. Models trained in this manner exhibit similar performance, but have a distribution with markedly higher density at zero, allowing more parameters to be pruned safely. Powerpropagation is general, intuitive, cheap and straight-forward to implement and can readily be combined with various other techniques. To highlight its versatility, we explore it in two very different settings: Firstly, following a recent line of work, we investigate its effect on sparse training for resource-constrained settings. Here, we combine Powerpropagation with a traditional weight-pruning technique as well as recent state-of-the-art sparse-to-sparse algorithms, showing superior performance on the ImageNet benchmark. Secondly, we advocate the use of sparsity in overcoming catastrophic forgetting, where compressed representations allow accommodating a large number of tasks at fixed model capacity. In all cases our reparameterisation considerably increases the efficacy of the off-the-shelf methods.

1 Introduction

Deep learning models are emerging as the dominant choice across several domains, from language [e.g. 31, 7] to vision [e.g. 41, 30] to RL [e.g. 69, 72]. One particular characteristic of these architectures is that they perform optimally in the overparameterised regime. In fact, their size seems to be mostly limited by hardware constraints. While this is potentially counter-intuitive, given a classical view on overfitting, the current understanding is that model size tends to have a dual role: It leads to better behaved loss surfaces, making optimisation easy, but also acts as a regulariser. This gives rise to the *double descent* phenomena, where test error initially behaves as expected, growing with model size due to overfitting, but then decreases again as the model keeps growing, and asymptotes as the model size goes to infinity to a better performance than obtained in the classical regime [5, 61].

As a consequence, many of the state of the art models tend to be prohibitively large, making them inaccessible to large portions of the research community despite scale still being a driving force in obtaining better performance. To address the high computational cost of inference, a growing body of work has been exploring ways to compress these models. As highlighted by several works [e.g.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

18, 21], size is only used as a crutch during the optimisation process, while the final solution requires a fraction of the capacity of the model. A typical approach therefore is to sparsify or prune the neural network after training by eliminating parameters that do not play a vital role in the functional behaviour of the model. Furthermore, there is a growing interest in sparse training [e.g. 57, 15, 35], where the model is regularly pruned or sparsified during training in order to reduce the computational burden.

Compressed or sparse representations are not merely useful to reduce computation. Continual learning, for example, focuses on learning algorithms that operate on non-iid data [e.g. 64, 20, 39]. In this setup, training proceeds sequentially on a set of tasks. The system is expected to accelerate learning on subsequent tasks as well as using newly acquired knowledge to potentially improve on previous problems, all of this while maintaining low memory and computational footprint. This is difficult due to the well studied problem of catastrophic forgetting, where performance on previous tasks deteriorates rapidly when new knowledge is incorporated. Many approaches to this problem require identifying the underlying set of parameters needed to encode the solution to a task in order to freeze or protect them via some form of regularisation. In such a scenario, given constraints on model size, computation and memory, it is advantageous that each learned task occupy as little capacity as possible.

In both scenarios, typically, the share of capacity needed to encode the solution is determined by the learning process itself, with no explicit force to impose frugality. This is in contrast with earlier works on L_0 regularisation that explicitly restrict the learning process to result in sparse and compressible representations [54]. The focus of our work, similar to the L_0 literature, is on how to encourage the learning process to be frugal in terms of parameter usage. However instead of achieving this by adding an explicit penalty, we enhance the "rich get richer" nature of gradient descent. In particular we propose a new parameterisation that ensures steps taken by gradient descent are proportional to the magnitude of the parameters. In other words, parameters with larger magnitudes are allowed to adapt faster in order to represent the required features to solve the task, while smaller magnitude parameters are restricted, making it more likely that they will be irrelevant in representing the learned solution.

2 Powerpropagation

The desired proportionality of updates to weight magnitudes can be achieved in a surprisingly simple fashion: In the forward pass of a neural networks, raise the parameters of your model (element-wise) to the α-th power (where $\alpha > 1$) while preserving the sign. It is easy to see that due to the chain rule of calculus the magnitude of the parameters (raised to $\alpha - 1$) will appear in the gradient computation, scaling the usual update. Therefore, small magnitude parameters receive smaller gradient updates, while larger magnitude parameters receive larger updates, leading to the aforementioned "rich get richer" phenomenon. This simple intuition leads to the name of our method.

More formally, we enforce sparsity through an implicit regulariser that results from reparameterising the model similar to Vaskevicius et al. [77], Zhao et al. [80]. This line of research builds on previous work on matrix factorisation [e.g. 26, 51]. In Vaskevicius et al. [77] a parameterisation of the form $w = v \odot v - u \odot u$ is used to induce sparsity, where $\odot$ stands for element-wise multiplication and we need both v and u to be able to represent negative values, since the parameters are effectively squared. In our work we rely on a simpler formulation where $w = v|v|^{\alpha-1}$, for any arbitrary power $\alpha \geq 1$ (as compared to fixing α to 2), which, since we preserved the sign of v, can represent both negative and positive values. For $\alpha = 1$ this recovers the standard setting.

If we denote by $\Theta = \mathcal{R}^M$ the original parameter space or manifold embedded in $\mathcal{R}^M$, our reparameterisation can be understood through an invertible map Ψ, where its inverse Ψ^{-1} projects $\theta \in \Theta$ into $\phi \in \Phi$, where Φ is a new parameter space also embedded in $\mathcal{R}^M$, i.e. $\Phi = \mathcal{R}^M$. The map is defined by applying the function $\Psi : \mathcal{R} \to \mathcal{R}, \Psi(x) = x|x|^{\alpha-1}$ element-wise, where by abuse of notation we refer to both the vector and element level function by Ψ. This new parameter space or manifold Φ has a curvature (or metric) that depends on the Jacobian of Ψ. Similar constructions have been previously used in optimisation, as for example in the case of the widely known Mirror Descent algorithm [4], where the invertible map Ψ is the link function. For deep learning, Natural Neural Networks [10] rely on a reparameterisation of the model such that in the new parameter space, at least initially, the curvature is close to the identity matrix, making a gradient descent step similar

to a second order step. Warp Gradient Descent [17] relies on a meta-learning framework to learn a nonlinear projection of the parameters with a similar focus of improving efficiency of learning. In contrast, our focus is to construct a parameterisation that leads to an implicit regularisation towards sparse representation, following [77], rather then improving convergence.

Given the form of our mapping Ψ, in the new parameterisation the original weight θ_i will be replaced by ϕ_i, where $\Psi(\phi_i) = \phi_i|\phi_i|^{\alpha-1} = \theta_i$ and i indexes over the dimensionality of the parameters. Note that we apply this transformation only to the weights of a neural network, leaving other parameters untouched. Given the reparameterised loss $\mathcal{L}(\cdot, \Psi(\phi))$, the gradient wrt. to ϕ becomes

$$\frac{\partial \mathcal{L}(\cdot, \Psi(\phi))}{\partial \phi} = \frac{\partial \mathcal{L}}{\partial \Psi(\phi)} \frac{\partial \Psi(\phi)}{\partial \phi} = \frac{\partial \mathcal{L}}{\partial \Psi(\phi)} \mathrm{diag}(\alpha|\phi|^{\circ\alpha-1}). \tag{1}$$

Note that diag indicates a diagonal matrix, and $|\phi|^{\circ\alpha-1}$ indicates raising element-wise the entries of vector $|\phi|$ to the power $\alpha - 1$. $\frac{\partial \mathcal{L}}{\partial \Psi(\phi)}$ is the derivative wrt. to the original weight $\theta = \phi|\phi|^{\circ\alpha-1}$ which is the gradient in the original parameterisation of the model. This is additionally multiplied (element-wise) by the factor $\alpha|\phi_i|^{\alpha-1}$, which will scale the step taken proportionally to the magnitude of each entry. Finally, for clarity, this update is different from simply scaling the gradients in the original parameterisation by the magnitude of the parameter (raised at $\alpha - 1$), since the update is applied to ϕ not θ, and is scaled by ϕ. The update rule (1) has the following properties:

(i) 0 is a critical point for the dynamics of any weight ϕ_i, if $\alpha > 1$. This is easy to see as $\frac{\partial \mathcal{L}}{\partial \phi_i} = 0$ whenever $\phi_i = 0$ due to the $\alpha|\phi_i|^{\alpha-1}$ factor.

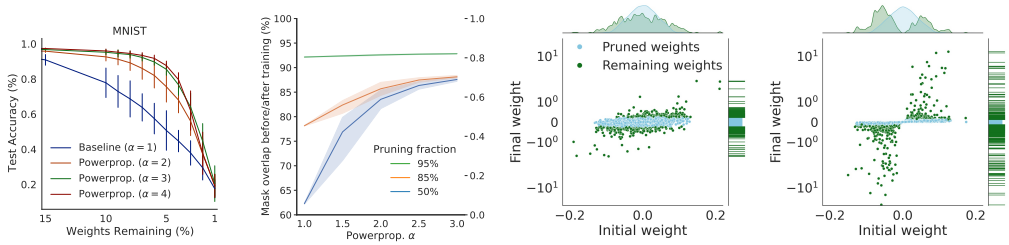
(ii) In addition, 0 is surrounded by a plateau and hence weights are less likely to change sign (gradients become vanishingly small in the neighbourhood of 0 due to the scaling). This should negatively affect initialisations that allow for both negative and positive values, but it might have bigger implications for biases.

(iii) This update is naturally obtained by the Backpropagation algorithm. This comes from the fact that Backpropagation implies applying the chain-rule from the output towards the variable of interest, and our reparameterisation simply adds another composition (step) in the chain before the variable of interest.

At this point the perceptive reader might be concerned about the effect of equation (1) on established practises in the training of deep neural networks. Firstly, an important aspect of reliable training is the initialisation ([e.g. 48, 22, 29]) or even normalisation layers such as batch-norm [34] or layer-norm [3]. We argue that our reparameterisation preserves all properties of typical initialisation schemes as it does not change the function. Specifically, let $\theta_i \sim p(\theta)$ where $p(\theta)$ is any distribution of choice. Then our reparameterisation involves initialising $\phi_i \leftarrow \mathrm{sign}(\theta_i) \cdot \sqrt[\alpha]{|\theta_i|}$, ensuring the neural network and all intermediary layers are functionally the same. This implies that hidden activations will have similar variance and mean as in the original parameterisation, which is what initialisation and normalisation focus on.

Secondly, one valid question is the impact of modern optimisation algorithms ([e.g. 12, 32, 37]) on our reparameterisation. These approaches correct the gradient step by some approximation of the curvature, typically given by the square root of a running average of squared gradients. This quantity will be proportional (at least approximately) to $\mathrm{diag}(\alpha|\phi|^{\circ\alpha-1})$[1]. This reflects the fact that our projection relies on making the space more curved and implicitly optimisation harder, which is what these optimisation algorithms aim to fix. Therefore, a naive use with Powerprop. would result in a reduction of the "rich get richer" effect. On the other hand, avoiding such optimisers completely can considerably harm convergence and performance. The reason for this is that they do not only correct for the curvature induced by our reparameterisation, but also the intrinsic curvature of the problem being solved. Removing the second effect can make optimisation very difficult. We provide empirical evidence of this effect in the Appendix.

[1] To see this assume the weights do not change from iteration to iteration. Then each gradient is scaled by the same value $\mathrm{diag}(\alpha|\phi|^{\circ 2(\alpha-1)})$ which factors out in the summation of gradients squared, hence the correction from the optimiser will undo this scaling. In practice ϕ changes over time, though slowly, hence approximately this will still hold.



(a) Sparsity Performance (b) Mask overlap (c) Baseline (d) $\alpha = 3.0$

Figure 1: Powerpropagation applied to Image classification. (a) Test accuracy at increasing levels of sparsity for MNIST (b) Overlap between masks computed before and after training (c) & (d) Analysis of weight distributions for a Baseline model and $\alpha = 4.0$. We use 10K weights chosen at random from the network. For a) & b) we show mean and standard deviation over 5 runs. *We provide code to reproduce the MNIST results (a) in the accompanying notebook.*

To mitigate this issue and make Powerprop. straightforward to use in any setting, we take inspiration from the target propagation literature [43, 44, 8, 49] which proposes an alternative way of thinking about the Backpropagation algorithm. In our case, we pretend that the exponentiated parameters are the de-facto parameters of the model and compute an update wrt. to them using our optimiser of choice. The updated exponentiated parameters are then seen as a *virtual target*, and we take a gradient descent step on ϕ towards these virtual targets. This will result in a descent step, which, while it relies on modern optimisers to correct for the curvature of the problem, does not correct for the curvature introduced by our parameterisation. Namely if we denote $optim : \mathcal{R}^M \to \mathcal{R}^M$ as the function that implements the correction to the gradient done by some modern optimiser, our update becomes $\Delta\phi = optim\left(\frac{\partial \mathcal{L}}{\partial \Psi(\phi)}\right) \operatorname{diag}(\alpha|\phi|^{\circ\alpha-1})$.

The proof that our update is indeed correct follows the typical steps taken in the target propagation literature. From a first order Taylor expansion of $\mathcal{L}(\phi - \eta\Delta\phi)$, we have that in order for $\Delta\phi$ to reduce the loss, the following needs to hold: $\langle \Delta\phi, \frac{\partial \mathcal{L}}{\partial \phi} \rangle > 0$. But we know that $\left\langle optim\left(\frac{\partial \mathcal{L}}{\partial \Psi(\phi)}\right), \frac{\partial \mathcal{L}}{\partial \Psi(\phi)} \right\rangle > 0$ as this was a valid step on $\Psi(\phi)$. Because $\operatorname{diag}(\alpha|\phi|^{\circ\alpha-1})$ is positive definite (diagonal matrix with all entries positive), we can multiply it on both sides, proving that $\langle \Delta\phi, \frac{\partial \mathcal{L}}{\partial \phi} \rangle > 0$. We provide more details in the Appendix. We will rely on this formulation in our empirical evaluation.

3 Effect on weight distribution and sparsification

At this point an empirical demonstration might illuminate the effect of equation (1) on model parameters and our ability to sparsify them. Throughout this work, we will present results from neural networks after the removal of low-magnitude weights. We prune such parameters by magnitude (i.e. $\min |\theta_i|$), following current best practice [e.g. 11, 59, 15]. This is based on a Taylor expansion argument [35] of a sparsely-parameterised function $f(x, \theta_s)$ which we would like to approximate its dense counterpart $f(x, \theta)$: $f(x, \theta_s) \approx f(\theta, x) + g^T(\theta_s - \theta) + \frac{1}{2}(\theta_s - \theta)^T H(\theta_s - \theta) + ...$ where g is the gradient vector and H the Hessian. As higher order derivatives are impractical to compute for modern networks, minimising the norm of $(\theta_s - \theta)$ is a practical choice instead.

Following the experimental setup in [18] we study the effect of Powerpropagation at different powers (α) relative to standard Backpropagation with otherwise identical settings. Figure 1a shows the effect of increasing sparsity on the layerwise magnitude-pruning setting for LeNet [45] on MNIST [47]. In both cases we notice a significant improvement over an otherwise equivalent baseline. While the choice of α does influence results, all choices lead to an improvement[2] in the MNIST setting. Where does this improvement come from? Figures 1c & 1d compare weights before and after training on MNIST at identical initialisation. We prune the network to 90% and compare the respective weight distributions. Three distinct differences are particularly noticeable: (i) Weights initialised close to zero are significantly less likely to survive the pruning process when Powerpropagation is applied (see green Kernel density estimate). (ii) Powerpropagation leads to a heavy-tailed distribution of

[2]For numerical stability reasons we typically suggest a choice of $\alpha \in (1, 3]$ depending on the problem.

trained weights, as (1) amplifies the magnitude of such values. These observations are what we refer to as the "rich-get-richer" dynamic of Powerpropagation.

One possible concern of (i) and (ii) are that Powerpropagation leads to training procedures where small weights cannot escape pruning, i.e. masks computed at initialisation and convergence are identical. This is undesirable as it is well established that pruning at initialisation is inferior [e.g. 21, 14, 19, 59]. To investigate this we plot the overlap between these masks at different pruning thresholds in Figure 1b. While overlap does increase with α, at no point do we observe an inability of small weights to escape pruning, alleviating this concern. Finally also note the aforementioned effect on weight signs, which can be observed in Figure 1d.

4 Overcoming catastrophic forgetting

In this section we discuss a strategy for harnessing Powerpropagation in the Continual Learning setting by overcoming catastrophic forgetting. As a reminder, this is the phenomenon of severe performance degradation on previous problems when a network is trained on another task. As the training of a single network to solve multiple tasks is inherently resource constraint, sparse networks and hence sparsity inducing techniques are a natural avenue by ensuring more compressed representations. Indeed there has been some interest in recent years [e.g. 55, 23] among which we focus and build upon the PackNet [55] model. The idea is simple and effective: Given a pre-defined number of tasks T, PackNet reserves $1/T\%$ of the weights per task by pruning at each task boundary. While this may lead to a small drop in performance, any further forgetting is then overcome by masking the gradients to weights that have previously survived pruning through a Backward Mask $\mathcal{M}^b$. Self-evidently, this procedure benefits from networks that maintain high performance at increased sparsity, which becomes particularly important for large T. Moreover, PackNet has the attractive property of merely requiring the storage of a binary mask per task, which comes at a cost of 1 bit per parameter. Hence space complexity grows linearly with the number of tasks T as $\mathcal{O}(D + T\log_2(D))$ (where D is the number of parameters), in stark contrast to a recent surge of methods involving the expensive storage (or generative modelling) of data for each task.

Algorithm 1: EfficientPackNet (EPN) + Powerpropagation.

Require : T tasks $[(X_1, y_1), \ldots, (X_T, y_T)]$; Loss & Eval. functions $\mathcal{L}, \mathcal{E}$; Initial weight distribution $p(u)$ (e.g. Uniform); Powerprop. α; Target performance $\gamma \in [0, 1]$; Sparsity range $S = [s_1, \ldots, s_n]$.

Output : Trained model ϕ; Task-specific Masks $\{\mathcal{M}^t\}$

1 $\mathcal{M}^b_i \leftarrow 1$ `// Backward mask`

2 $\phi_i \leftarrow \text{sign}(\theta) \cdot \sqrt[\alpha]{|\theta_i|}; \theta_i \sim p(\theta)$

3 **for** $t \in [1, \ldots, T]$ **do**

4 $\quad W \leftarrow \arg\min \mathcal{L}(X_t, y_t, \phi, \mathcal{M}_b)$ `// (Powerprop.)`

5 $\quad P \leftarrow \mathcal{E}(X_t, y_t, \phi)$ `// Performance of dense model`

6 $\quad l \leftarrow 1$

7 $\quad$ **do**

8 $\quad\quad \mathcal{M}^t_i = \left\{ \begin{array}{ll} 1 & \text{if } i \in \text{TopK}(\phi_i, s_l) \\ 0 & \text{otherwise} \end{array} \right\}$ `// New Forward mask at sparsity` s_i

9 $\quad\quad P_s \leftarrow \mathcal{E}(X_T, y_T, \phi \odot \mathcal{M}^t)$ `// Performance of sparse model`

10 $\quad\quad l \leftarrow l + 1$

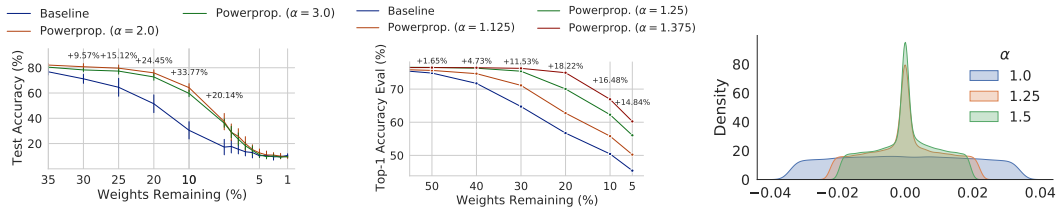
11 $\quad$ **while** $P_s > \gamma P \wedge i \leq n$;

12 $\quad$ Re-initialise pruned weights $\mathcal{M}^b \leftarrow \neg \bigvee_{i=1}^{t} \mathcal{M}^t$ `// Update backward mask`

$\quad$ `// Optionally retrain with masked weights` $\phi \odot \mathcal{M}^t$

13 **end**

Nevertheless, the approach suffers from its assumption of a known number of maximum tasks T and its possibly inefficient resource allocation: By reserving a fixed fraction of weights for each task, no distinction is made in terms of difficulty or relatedness to previous data. We overcome both issues through simple but effective modifications resulting in a method we term *EfficientPacknet* (EPN) that may be used with or without Powerprop. We show the procedure in Algorithm 1. Important differences to PackNet are: **(a) Lines 7-11**: We propose a simple search over pre-defined sparsity rates s_n, terminating the search when the sparse models performance falls short of the minimum accepted target performance γP or a maximum sparsity is reached. This can be computed on a held-out validation set. In smaller models the cost of this step maybe reduced by an error estimate based on a Taylor expansion [46]. This helps overcome the inefficient resource allocation of PackNet. **(b) Line 8**: The mask is chosen among all network parameters, including ones used by previous tasks,



(a) CIFAR-10 (b) ImageNet (c) 100K smallest weights

Figure 2: One-shot pruning on CIFAR-10 and ImageNet. Numbers in a) & c) show the absolute improvement of the best α wrt. to the baseline. For CIFAR-10 we show mean and standard deviation over five runs. Repeated runs lead to almost identical results for ImageNet and are omitted.

thus encouraging reuse to existing parameters. Thus, the fraction of newly masked weight is $\leq s_n \cdot P$. **(c) Line 12**: We re-initialise the weights as opposed to zeroing them (as default PackNet does), due to the critical point property (Section 2). While we could leave the weight at their previous value, we found this to lead to slightly worse performance (see ablation study in the Appendix). These changes make the algorithm more suitable for long sequences of tasks, as we will show in our experiments. In addition, they overcome the assumption of an a priori known number of tasks T. Note also that since the Backward pass becomes increasingly sparse, the method becomes more computationally efficient with larger T.

5 Related work

Sparsity in neural networks has been an active research since at least the first wave of interest in neural networks following Backpropagation [66]. Early work was based on the Hessian [e.g. 60, 46, 28], an attractive but impractical approach in modern architectures. The popularity of magnitude-based pruning dates back to at least [74], whereas iterative pruning was first noted to be effective in [73]. Modern approaches are further categorised as Dense→Spare or Sparse→Sparse methods, where the first category refers to the instantiation of a dense network that is sparsified throughout the training. Sparse→Sparse algorithms maintain constant sparsity throughout, giving them a clear computational advantage.

Among Dense→Spare methods, L_0 regularisation [54] uses non-negative stochastic gates to penalise non-zero weights during training. Variational Dropout [58] allows for unbounded dropout rates, leading to sparsity. Both are alternatives to magnitude-based pruning. Soft weight threshold reparameterisation [42] is based on learning layerwise pruning thresholds allowing non-uniform budgets across layers. Finally, the authors of [18] observe that re-training weights that previously survived pruning from their initial value can lead to sparsity at superior performance. Examples of Sparse→Spare methods are Single Shot Network Pruning [50] where an initial mask is chosen according to the salience and kept fixed throughout. In addition, a key insight is to iteratively drop & grow weights while maintaining a fixed sparsity budget. Deep Rewiring for instance [6] augments SGD with a random walk. In addition, weights that are about to change sign are set to zero, activating other weights at random instead. Rigging the Lottery (RigL) [15] instead activates new weights by highest magnitude gradients, whereas Top-KAST [35] maintains two set of masks for (sparse) forward and backward-passes. This also allows the authors to cut the computational budget for gradient computation. In addition, a weight-decay based exploration scheme is used to allow magnitude-based pruning to be used for both masks. Other notable works in this category are Sparse Evolutionary Training [57] and Dynamic sparse reparameterisation [59].

While continual learning has been studied as early as [64, 20], the work on Elastic Weight Consolidation (EWC) [39] has recently started a resurgence of interest in the field. Contemporary approaches are subdivided into methods relying on regularisation [e.g. 79, 62, 75], data replay [e.g. 71, 9, 65, 36] or architectural approaches [e.g. 68, 70]. Works such as PackNet [e.g. 55] have also introduced sparsity as a tool for more efficient continual learning. Other approaches also include PathNets which [16] use evolutionary training to find subnetworks within a larger model for individual tasks.

Method	0%				
Dense	76.8 ± 0.09				
	80%	**90%**	**95%**	**80% (ERK)**	**90% (ERK)**
Static [15]	70.6 ± 0.06	65.8 ± 0.04	59.5 ± 0.11	72.1 ± 0.04	67.7 ± 0.12
DSR [59]	73.3	71.6			
SNFS [11]	74.2	72.3		75.2 ± 0.11	72.9 ± 0.06
SNIP [50]	72.0 ± 0.10	67.2 ± 0.12	57.8 ± 0.40		
RigL [15]	74.6 ± 0.06	72.0 ± 0.05	67.5 ± 0.10	75.1 ± 0.05	73.0 ± 0.04
Iterative pruning [21]	75.6	73.9	70.6		
Iterative pruning (ours)	75.3 ± 0.07	73.7 ± 0.14	70.6 ± 0.05		
Powerprop. + Iter. Pruning	$\mathbf{75.7} \pm 0.05$	$\mathbf{74.4} \pm 0.02$	$\mathbf{72.1} \pm 0.00$		
TopKAST† [35]	75.47 ± 0.03	74.65 ± 0.03	$\mathbf{72.73} \pm 0.10$	75.71 ± 0.06	74.79 ± 0.05
Powerprop. + TopKAST†	$\mathbf{75.75} \pm 0.05$	$\mathbf{74.74} \pm 0.04$	$\mathbf{72.89} \pm 0.10$	$\mathbf{75.84 \pm 0.01}$	$\mathbf{74.98 \pm 0.09}$
TopKAST* [35]	76.08 ± 0.02	75.13 ± 0.03	73.19 ± 0.02	76.42 ± 0.03	75.51 ± 0.05
Powerprop. + TopKAST*	$\mathbf{76.24} \pm 0.07$	$\mathbf{75.23} \pm 0.02$	$\mathbf{73.25} \pm 0.02$	$\mathbf{76.76 \pm 0.08}$	$\mathbf{75.74 \pm 0.08}$

Table 1: Performance of sparse ResNet-50 on Imagenet. Baseline results from [15]. ERK: Erdos-Renyi Kernel [15]. */†: TopKAST at 0%/50% Backward Sparsity. Shown are mean and standard deviation over 3 seeds.

6 Experimental Evaluation

We now provide an experimental comparison of Powerprop. to a variety of other techniques, both in the sparsity and catastrophic forgetting settings. Throughout this section we will be guided by three key questions: (i) Can we provide experimental evidence for *inherent sparsity*? (ii) If so, can Powerprop. be successfully combined with existing sparsity techniques? (iii) Do improvements brought by Powerprop. translate to measurable advances in Continual Learning? What are strengths and limitations of this strategy?

This section covers varying experimental protocols ranging from supervised image classification to generative modelling and reinforcement learning. Due to space constraints we focus primarily on the interpretation of the results and refer the interested reader to details and best hyperparameters in the Appendix.

6.1 Inherent Sparsity

Turning to question (i) first, let us investigate this question by comparing the performance of a pruned method without re-training. This is commonly known as the one-shot pruning setting and is thus a Dense $\rightarrow$ Sparse method. If Powerprop. does indeed lead to inherently sparse networks its improvement should show most clearly in this setting. Figure 2 shows this comparison for image classification on the popular CIFAR-10 [40] and ImageNet [67] Datasets using a smaller version of AlexNet [41] and ResNet50 [30] respectively.[3] In both situations we notice a marked improvement, up to 33% for CIFAR and 18% for ImageNet. In Fig. 2c we show the smallest 100K weights, in which we observe a notable peak around zero for $\alpha > 1.0$, providing strong evidence in favour of the inherent sparsity argument. Finally is worth noting that the choice of α does influence the optimal learning rate schedule and best results were obtained after changes to the default schedule.

6.2 Advanced Pruning

Method	80% Sparsity	90% Sparsity
Iter. Pruning $_{(1.5\times)}$	76.5 [0.84x]	75.2 [0.76x]
RigL $_{(5\times)}$	$76.6 \pm_{0.06}$ [1.14x]	$75.7 \pm_{0.06}$ [0.52x]
ERK		
RigL $_{(5\times)}$	$77.1 \pm_{0.06}$ [2.09x]	$76.4 \pm_{0.05}$ [1.23x]
Powerprop. + TopKAST* $_{(2\times)}$	$77.51 \pm_{0.03}$ [1.21x]	$76.94 \pm_{0.10}$ [0.97x]
Powerprop. + TopKAST* $_{(3\times)}$	$\mathbf{77.64} \pm_{0.05}$ [1.81x]	$\mathbf{77.16} \pm_{0.19}$ [1.46x]

Table 2: Extended training. Numbers in square Brackets show FLOPs relative to training of a dense model for a single cycle.

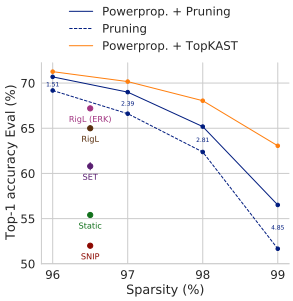


Figure 3: Extreme sparsity

[3]The motivating example in Section 3 also follows this protocol.

We now address question (ii) by comparing Powerprop. when used in conjunction with and compared to exiting state-of-the-art algorithms on ImageNet. We orient ourselves primarily on the experimental setup in [15] & [35], both of which present techniques among the strongest in the literature. As in the previous section we combine Powerprop. with Iterative Pruning for Dense→Sparse and select TopKAST [35] as our method of choice from the Sparse→Sparse literature.

We distinguish between three sub-settings a) Table 1: which shows ResNet 50@$\{80\%, 90\%95\%\}$ sparsity, for which the largest number of baseline results are available. We also provide results using the Erdos-Renyi Kernel [15], a redistribution of layerwise sparsity subject to the same fixed overall budget. While this improves results, it increases the floating point operations (FLOPs) at test time. b) Table 2: Extended training, where we scale training steps and learning rate schedule by a factor of the usual 32K steps. While this tends to lead to better results the computational training cost increases. c) Figure 3: Extreme sparsity at $95\% - 99\%$ thus testing methods at the limit of what is currently possible. All experiments are repeated 3 times and report mean and standard deviation. As Powerprop. introduces a negligible amount of extra FLOPs (over baseline methods) we only shows such values in the extended training setting to provide a fair comparison to the setup in [15]. A longer discussion of a FLOP trade-off can be found in the work on TopKAST [35].

Powerprop. improves both Pruning & TopKAST in all settings without any modification to the operation of those algorithms while outperforming a large array of other techniques. To the best of our knowledge those results constitutes a new state-of-the art performance in all three settings. We emphasise that many of the methods we compare against are based on pruning weights by their magnitude and would thus also be likely to benefit from our method.

6.3 Overcoming Catastrophic forgetting

Algorithm	MNIST	notMNIST
Naive	-258.40	-797.88
Laplace [13, 70]	-108.06	-225.54
EWC [39]	-105.78	-212.62
SI [79]	-111.47	-190.48
VCL [62]	-94.27	-187.34
Eff. PackNet (EPN)	-94.50	-177.08
Powerprop. + Laplace	-96.72	-196.87
Powerprop. + EWC	-95.79	-188.56
Powerprop. + EPN	**-92.15**	**-174.54**

Table 3: Results on the Continual generative modelling experiments. Shown is an importance-sampling estimate of the test log-likelihood.

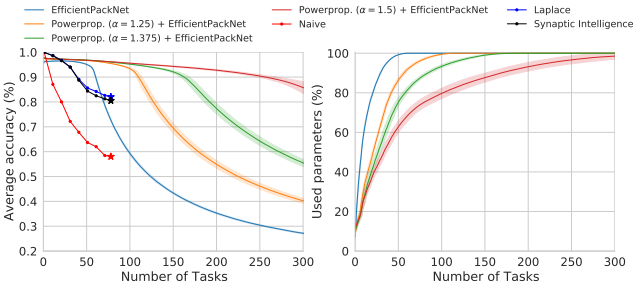


Figure 4: Powerpropagation on extremely long task sequences. **Left**: Average accuracy over all tasks. **Right**: Fraction of unmasked parameters. Shown are mean and standard deviation over 10 repetitions of the experiment.

We now turn to question (iii), studying the effectiveness of sparsity methods for Continual Learning. We first consider long task sequences on the Permuted MNIST benchmark, first popularised by [24]. While usually regarded to be an idealised and simple setting, we push its boundaries by drastically increasing the number of tasks, which was previously limited to 100 in [63] and 78 in [33], whereas we learn up to 300 in a single model. In all cases we use a network with a single output layer across all tasks, following the protocol by and comparing to [33]. Figure 4 shows the average accuracy and fraction of used weights. We significant outperform baselines with an average acc. of $\approx 86\%$ after 300 tasks, whereas the closest baseline (Laplace [70]) falls to 85.61% after merely 51 tasks. This also highlight the effect of α more noticeably, showing increased performance for higher α. Note also that the gradient of the used weights curve flattens, showcasing the dynamic allocation of larger number of parameters to tasks early in the sequence, whereas later tasks reuse a larger fraction of previous weights. While we also attempted a comparison with standard PackNet, the model failed, as its default allocation of <0.4% of weights per task is insufficient for even a single MNIST task (see Figure 1a). The perspective reader might notice higher performance for considered baselines at small number of tasks. This is due to our choice of $\gamma = 0.9$ (see Section 4) which we optimise for long sequences. However, we also show results for a short 10-task sequence and the Split-MNIST benchmark [79] in the Appendix. In both cases we outperform published results. It is worth mentioning that this increased α can be done up to a certain limit at which point training for high number of tasks can become unstable. We will investigate this effect in future work.

Table 4: Reinforcement Learning Results on Continual World [78]. Error bars provide 90% confidence intervals.

Method	Avg. success	Forgetting
Fine-tuning	0.00 [0.00, 0.00]	0.87 [0.84, 0.89]
A-GEM [9]	0.02 [0.01, 0.04]	0.89 [0.86, 0.91]
EWC [39]	0.66 [0.61, 0.71]	0.07 [0.04, 0.11]
MAS [2]	0.61 [0.56, 0.66]	-0.01 [-0.04, 0.01]
Perfect Memory	0.36 [0.32, 0.40]	0.07 [0.05, 0.10]
VCL [62]	0.52 [0.47, 0.57]	-0.01 [-0.03, 0.01]
PackNet [55]	0.80 [0.76, 0.84]	-0.02 [-0.04, -0.00]
Powerprop. + EfficientPackNet $_{(\alpha=1.375)}$	0.82 [0.77, 0.87]	-0.01 [-0.03, 0.01]
Powerprop. + EfficientPackNet $_{(\alpha=1.5)}$	**0.86** [0.82, 0.90]	**0.00** [-0.02, 0.02]

Moving onto more complex settings, we use Powerprop. on the continual generative modelling experiment in [62] where 10 tasks are created by modelling a single character (using the MNIST and notMNIST[4] Datasets) with a variational autoencoder [38]. We report quantitative results in Table 3. In both cases we observe Powerprop. + EfficientPackNet outperforming the baselines. Interestingly, its effectiveness for Catastrophic Forgetting is not limited to PackNet. In particular, we provide results for the Elastic Weight Consolidation (EWC) method and its online version (Laplace) [39, 70] both of which are based on a Laplace approximation (using the diagonal Fisher information matrix). As Powerprop. results in inherently sparse networks, the entries of the diagonal Fisher corresponding to low magnitude weights are vanishingly small, resulting in virtually no regularisation of those parameters. We observe significant improvements with no other change to the algorithms.

Finally, we move onto Catastrophic Forgetting in Reinforcement Learning using seven tasks from the recently published Continual World benchmark [78], a diverse set of realistic robotic manipulation tasks. This is most arguably the most complex setting in our Continual Learning evaluation. Following the authors, we use Soft actor-critic [27], training for 1M steps with Adam [37] (relying on the formulation in Section 2) on each task while allowing 100k retrain steps for PackNet (acquiring no additional data during retraining). We report the average success rate (a binary measure based on the distance to the goal) and a forgetting score measuring the difference of current performance to that obtained at the end of training (but importantly before pruning) on a certain task. Thus, if Powerprop. does improve results, this will show in the Forgetting metric. Results are shown in Table 4. Indeed, we notice that a higher α leads to a reduction of forgetting and hence an overall improvement over PackNet, resulting in superior performance. Finally, it is worth mentioning that (Efficient)PackNet still assume hard task boundaries as well as task labels (needed to select the mask $\mathcal{M}_t$) at test time. However, it has previously been shown that it is possible to overcome these limitations using a changepoint detection algorithm [e.g. 56, 1].

7 Discussion

In this work we introduced Powerpropagation, demonstrating how its effect on gradients leads to *inherently sparse* networks. As we show in our empirical analyses, Powerprop. is easily combined with existing techniques, often merely requiring a few characters of code to implement. While we show its effect on only a few established algorithms, we hypothesise that a combination with other techniques could lead to further improvements. Bridging gaps between the fields of sparse networks and Continual Learning, we hope that our demonstration of its effect on Catastrophic Forgetting will inspire further interest in such ideas. In terms of future work, we believe that devising automated schemes for learning α may be worthwhile, making the use of our method even simpler. Particular attention ought to be paid to its interaction with the learning rate schedule, which we found worthwhile optimising for particular choices of α. Finally, we also hope further work will bring additional theoretical insights into our method.

[4] `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`

References

[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, May 2003.

[5] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[6] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[8] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 10–19, 2014.

[9] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[10] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, and koray kavukcuoglu. Natural neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[11] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[13] Eleazar Eskin, Alex J Smola, and SVN Vishwanathan. Laplace propagation. In *Advances in neural information processing systems*, pages 441–448, 2004.

[14] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

[15] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[16] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[17] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkeiQlBFPB`.

[18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 8, 2019.

[20] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[21] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[22] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[23] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

[24] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[25] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[26] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/58191d2a914c6dae66371c9dcdc91b41-Paper.pdf`.

[27] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[28] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*, 2015.

[32] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[33] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[34] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[35] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33: 20744–20754, 2020.

[36] Christos Kaplanis, Claudia Clopath, and Murray Shanahan. Continual reinforcement learning with multi-timescale replay. *arXiv preprint arXiv:2004.07530*, 2020.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[39] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[42] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[43] Yann LeCun. Learning process in an asymmetric threshold network. In *Disordered Systems and Biological Organization*, pages 233–240. Springer Berlin Heidelberg, 1986.

[44] Yann Lecun. *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. Universite P. et M. Curie (Paris 6), June 1987.

[45] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[46] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[47] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[48] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[49] Dong-Hyun Lee, Saizheng Zhang, Antoine Biard, and Yoshua Bengio. Target propagation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[50] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[51] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Proceedings of the 31st Conference On Learning Theory*, pages 2–47, 2018.

[52] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[53] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6470–6479, 2017.

[54] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[55] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[56] Kieran Milan, Joel Veness, James Kirkpatrick, Michael Bowling, Anna Koop, and Demis Hassabis. The forget-me-not process. *Advances in Neural Information Processing Systems*, 29: 3702–3710, 2016.

[57] 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):1–12, 2018.

[58] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

[59] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019.

[60] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989.

[61] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[62] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[63] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*, 2018.

[64] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7 (2):123–146, 1995.

[65] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

[66] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[67] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[68] 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.

[69] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[70] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[71] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[72] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[73] Nikko Ström. Sparse connection and pruning in large dynamic artificial neural networks. In *Fifth European Conference on Speech Communication and Technology*, 1997.

[74] Georg Thimm and Emile Fiesler. Evaluating pruning methods. In *Proceedings of the International Symposium on Artificial neural networks*, pages 20–25. Citeseer, 1995.

[75] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*, 2019.

[76] Gido M Van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[77] Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/5cf21ce30208cfffaa832c6e44bb567d-Paper.pdf`.

[78] Maciej Wolczyk, Michal Zajac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Continual world: A robotic benchmark for continual reinforcement learning. *arXiv preprint arXiv:2105.10919*, 2021.

[79] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017.

[80] Peng Zhao, Yun Yang, and Qiao-Chu He. Implicit regularization via hadamard product over-parametrization in high-dimensional linear regression, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **Yes**
   (b) Did you describe the limitations of your work? **Yes, we discussed the need for learning rates to be adjusted for best performance with Powerpropagation. We also discussed that PackNet does require Task IDs at inference and assumes hard task boundaries.**
   (c) Did you discuss any potential negative societal impacts of your work? **No, to the best of our knowledge there are is no negative impact.**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **Yes**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **n/a**
   (b) Did you include complete proofs of all theoretical results? **n/a**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **We provide code for the MNIST experiment as this shows our method in the simplest setup without much additional code that is arguably not specific to our contributions. We aim to release code for the more sophisticated setups upon acceptance.**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **We list such values in the supplementary material.**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **For the majority of results we run experiments multiple times and report error bars.**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **No**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **Yes**
   (b) Did you mention the license of the assets? **Yes, where applicable**
   (c) Did you include any new assets either in the supplemental material or as a URL? **Yes, we included code to reproduce sparsity results on MNIST.**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **Yes, where applicable**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **The underlying Datasets used for our evaluation are: MNIST, CIFAR10, ImageNet, notMNIST, ContinualWorld/MetaWorld. To the best of our knowledge there is no personally identifiable information or offensive content in those sources.**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **n/a**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **n/a**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **n/a**