Understanding the Role of Training Regimes in Continual Learning

Seyed Iman Mirzadeh
Washington State University, USA
seyediman.mirzadeh@wsu.edu

Mehrdad Farajtabar
DeepMind, USA
farajtabar@google.com

Razvan Pascanu
DeepMind, UK
razp@google.com

Hassan Ghasemzadeh
Washington State University, USA
hassan.ghasemzadeh@wsu.edu

Abstract

Catastrophic forgetting affects the training of neural networks, limiting their ability to learn multiple tasks sequentially. From the perspective of the well established plasticity-stability dilemma, neural networks tend to be overly plastic, lacking the stability necessary to prevent the forgetting of previous knowledge, which means that as learning progresses, networks tend to forget previously seen tasks. This phenomenon coined in the *continual learning* literature, has attracted much attention lately, and several families of approaches have been proposed with different degrees of success. However, there has been limited prior work extensively analyzing the impact that different training regimes – learning rate, batch size, regularization method– can have on forgetting. In this work, we depart from the typical approach of altering the learning algorithm to improve stability. Instead, we hypothesize that the geometrical properties of the local minima found for each task play an important role in the overall degree of forgetting. In particular, we study the effect of dropout, learning rate decay, and batch size, on forming training regimes that *widen* the tasks' local minima and consequently, on helping it not to forget catastrophically. Our study provides practical insights to improve stability via simple yet effective techniques that outperform alternative baselines.[1]

1 Introduction

We study the *continual learning* problem, where a neural network model should learn a sequence of tasks rather than a single one. A significant challenge in continual learning (CL) is that during training on each task, the data from previous ones are unavailable. One consequence of applying typical learning algorithms under such a scenario is that as the model learns newer tasks, the performance of the model on older ones degrades. This phenomenon is known as *"catastrophic forgetting"* [52].

This forgetting problem is closely related to the *"stability-plasticity dilemma"* [53], which is a common challenge for both biological and artificial neural networks. Ideally, a model needs plasticity to obtain new knowledge and adapt to new environments, while it also requires stability to prevent forgetting the knowledge from previous environments. If the model is very plastic but not stable, it can learn fast, but it also forgets quickly. Without further modifications in training, a naively trained neural network tends to be plastic but not stable. Note that plasticity in this scenario does not necessarily imply that neural nets can learn new tasks *efficiently*. In fact, they tend to be extremely data inefficient. By being plastic, we mean a single update can change the function considerably.

[1]The code is available at: https://github.com/imirzadeh/stable-continual-learning

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

With the recent advances in the deep learning field, continual learning has gained more attention since the catastrophic forgetting problem poses a critical challenge for various applications [43, 37]. A growing body of research has attempted to tackle this problem in recent years [58, 72, 57, 29].

Despite the tangible improvements in the continual learning field, the core problem of catastrophic forgetting is still under-studied. In particular, a variety of neural network models and training approaches have been proposed, however, to the best of our knowledge, there has been little work on systematically understanding the effect of common training regimes created by varying dropout regularization, batch size, and learning rate on overcoming catastrophic forgetting[2]. Fig. 1 shows how significantly these techniques can overcome catastrophic forgetting.

In this work, we explore the catastrophic forgetting problem from an optimization and loss landscape perspective (Section 3) and hypothesize that the geometry of the local minima found for the different learned tasks correlates with the ability of the model to not catastrophically forget. Empirically we show how a few well-known techniques, such as dropout and large learning rate with decay and shrinking batch size, can create a training regime to affect the stability of neural networks (Section 4). Some of them, like dropout, had been previously proposed to help continual learning [22, 54]. However, in this work, we provide an alternative justification of why these techniques are effective. Crucially, we empirically show that jointly with a carefully tuned learning rate schedule and batch size, these simple techniques can outperform considerably more complex algorithms meant to deal with continual learning (Section 5). Our analysis can be applied to any other training technique that widens the tasks' local minima or shrinks the distance between them.

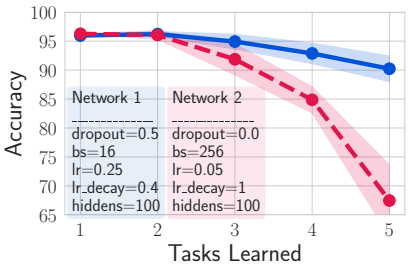


Figure 1: For the same architecture and dataset (Rotation MNIST) and only changing the training regime, the forgetting is reduced significantly at the cost of a relatively small accuracy drop on the current task. Refer to appendix C for details.

Our work shows that plain neural networks can be much stronger baselines for continual learning than previously thought, provided that we use the right hyperparameters. Moreover, the choice for the hyperparameters is orthogonal to other continual learning methods and can be integrated with these methods, as we show in Appendix C.8.

2 Related work

Several continual learning methods have been proposed to tackle catastrophic forgetting. Following [43], we categorize these algorithms into three general groups.

The first group consists of replay based methods that build and store a memory of the knowledge learned from old tasks [48, 62, 82, 68, 63], known as *experience replay*. iCaRL [61] learns in a class-incremental way by having a fixed memory that stores samples that are close to the center of each class. Averaged Gradient Episodic Memory (A-GEM) [7] is another example of these methods which build a dynamic episodic memory of parameter gradients during the learning process while ER-Reservoir [9] uses a Reservoir sampling method as its selection strategy.

The methods in the second group use explicit regularization techniques to supervise the learning algorithm such that the network parameters are consistent during the learning process [41, 80, 44, 1, 42]. As a notable work, Elastic weight consolidation (EWC) [41], uses the Fisher information matrix as a proxy for weights' importance and guides the gradient updates. They are usually inspired by a Bayesian perspective [56, 71, 67, 13, 64]. With a frequentist view, some other regularization based methods have utilized gradient information to protect previous knowledge [14, 24, 79]. For example, Orthogonal Gradient Descent (OGD) [14] uses the projection of the prediction gradients from new tasks on the subspace of previous tasks' gradients to maintain the learned knowledge.

Finally, in parameter isolation methods, in addition to potentially a shared part, different subsets of the model parameters are dedicated to each task [65, 78, 35, 60, 46]. This approach can be viewed

[2]Potential exceptions being the early work of Goodefellow et. al [22] and a recent one by Mirzadeh et. al [54]

as a flexible gating mechanism, which enhances stability and controls the plasticity by activating different gates for each task. [50] proposes a neuroscience-inspired method for a context-dependent gating signal, such that only sparse, mostly non-overlapping patterns of units are active for any one task. PackNet [49] implements a controlled version of gating by using network pruning techniques to free up parameters after finishing each task.

While continual learning is broader than just solving the catastrophic forgetting, in this work we will squarely focus on catastrophic forgetting, which has been an important aspect, if not the most important one, of research for Continual Learning in the last few years. Continual Learning as an emerging field in Artificial Intelligence is connected to many other areas such as Meta Learning [4, 35, 24, 62], Few Shot Learning [75, 20], Multi-task and Transfer Learning [24, 35] and the closely related problem of exploring task boundary or task detection [60, 2, 25, 36]. Very recently, Mirzadeh et al. [55] have studied the mode connectivity of continual learning and multitask learning minima. Moreover, Wallingford et al. [73] proposed a framework for integration of solutions across supervised learning, few-shot learning, continual learning, and efficient machine learning to facilitate the research in the intersection of these fields.

3 Forgetting during training

Let us begin this section by introducing some notation to express the effect of *forgetting* during the sequential learning of tasks. For simplicity, let us consider the supervised learning case, which will be the focus of this work. We consider a sequence of K tasks $\mathcal{T}_k$ for $k \in \{1, 2, \ldots, K\}$. Let $\mathcal{W} \in \mathbb{R}^d$ be the parameter space for our model. The total loss on the training set for task k is denoted by

$$L_k(w) = \mathbb{E}[\ell_k(w; x, y)] \approx \frac{1}{|\mathcal{T}_k|} \sum_{(x,y)\in\mathcal{T}_k} , \ell_k(w; x, y) \tag{1}$$

where, the expectation is over the data distribution of task k and ℓ_k is a differentiable non-negative loss function associated with data point (x, y) for task k. In the continual learning setting, the model learns sequentially, without access to examples of previously seen tasks. For simplicity and brevity, let us focus on measuring the *forgetting* in continual learning with two tasks. It is easy to extend these findings to more tasks.

Let w_1^* and w_2^* be the convergent or optimum parameters after training has been finished for the first and second task sequentially. We formally define the *forgetting* (of the first task) as:

$$F_1 \triangleq L_1(w_2^*) - L_1(w_1^*). \tag{2}$$

We hypothesize that F_1 *strongly correlates with properties of the curvature of* L_1 *around* w_1^* *and* L_2 *around* w_2^*. In what follows, we will formalize this hypothesis.

One important assumption that we rely on throughout this section is that we can use a second-order Taylor expansion of our losses to understand the learning dynamics during the training of the model. While this might seem as a crude approximation in general — for a nonlinear function and non-infinitesimal displacement this approximation can be arbitrarily bad — we argue that the approximation has merit for our setting. In particular, we rely on the wealth of observations for overparametrized models where the loss tends to be very well behaved and almost convex in a reasonable neighborhood around their local minima. E.g. for deep linear models this property has been studied in [66]. Works as [11, 23] make similar claims for generic models. [31] also corroborates that within the NTK regime learning is well behaved. In continual learning, similar strong assumptions are made by most approaches that rely on approximating the posterior on the weights by a Gaussian [41] or a first-order approximation of the loss surface around the optimum [14]. Armed with this analytical tool, to compute the forgetting, we can approximate $L_1(w_2^*)$ around w_1^*:

$$L_1(w_2^*) \approx L_1(w_1^*) + (w_2^* - w_1^*)^\top \nabla L_1(w_1^*) + \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*) \tag{3}$$

$$\approx L_1(w_1^*) + \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*), \tag{4}$$

where, $\nabla^2 L_1(w_1^*)$ is the Hessian for loss L_1 at w_1^* and the last equality holds because the model is assumed to converge to a stationary point where gradient's norm vanishes, thus $\nabla L_1(w_1^*) \approx \mathbf{0}$.

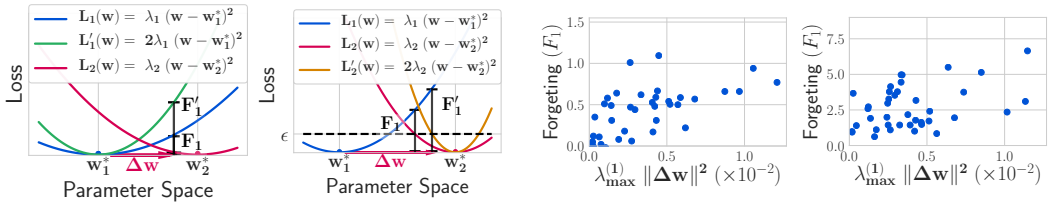


Figure 2: **(a)**: For a fixed Δw, the wider the curvature of the first task, the less the forgetting. **(b)**: The wider the curvature of the second task, the smaller $\|\Delta w\|$. **(c) and (d):** Empirical verification of (5) for Rotated MNIST and Permuted MNIST, respectively.

Under the assumption that the critical point is a minimum (or the plateau we get stuck in surrounds a minimum), we know that the Hessian needs to be positive semi-definite. Defining $\Delta w = w_2^* - w_1^*$ as the relocation vector, we can bound the forgetting F_1 as follows:

$$F_1 = L_1(w_2^*) - L_1(w_1^*) \approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1^*)\Delta w \leq \frac{1}{2}\lambda_1^{max} \|\Delta w\|^2 , \tag{5}$$

where λ_1^{max} is the maximum eigenvalue of $\nabla^2 L_1(w_1^*)$. Fig. 2a shows how wider L_1 (lower λ_1^{max}) leads to less forgetting, both in terms of an illustrative example as well as showing empirical evidence for this relationship on Rotated MNIST and Permuted MNIST.

A few notes on this bound. First, the bound is achieved when Δw is co-aligned to the eigenvector corresponding to λ_1. Given that the displacement is generated by the learning process on task 2, we can think of it as a random vector with respect to the eigenvectors of the Hessian of the first task. We know, therefore, that the tightness of the bound correlates with the dimensionality of w. In the extreme one-dimensional case, the Hessian matrix becomes a scalar given by λ_1^{max}, and the bound is exact. As we increase the number of dimensions, the probability of two vectors to be perpendicular goes to 1. Hence in high-dimensional spaces is more likely for the bound to be relatively loose and for the entire spectrum of eigenvalues to play a much more important role in defining F_1. Namely, as the number of eigenvalues with low magnitude increases, the more likely it is for F_1 to be small. Assuming that the displacement vector is equally distributed over all the eigenvectors, then the trace of the Hessian will correlate stronger with F_1 than the largest eigenvalue. However, reasoning in terms of the spectrum can be impractical (note, for example, that one can not trivially re-write the bound in terms of the trace). So we believe it is useful to think about λ_1^{max} as long as any conclusion is contextualized correctly and the training regime we consider, implicitly, is aimed at lowering the entire spectrum, not just the largest eigenvalue.

We also want to highlight λ_1^{max} has been used previously to describe the *width of a local minima* [28, 39], with similar notes regarding the role of the entire spectrum [12, 39]. This property is central to the wide/narrow minima hypothesis for why neural networks generalize well. Our hypothesis is not tied to the generalization of wide minima, but we rely on the same or at least a very related concept of *width*. Therefore, to reduce forgetting, each task should push its learning towards *wider minima* and can employ the same techniques used to widen the minima to improve generalization.

Resuming from Eq. 5, controlling the Hessian spectrum without controlling the norm of the displacement, however, might not ensure that F_1 is minimized.

$\|\Delta w\|$ is technically controlled by the subsequent tasks. We first notice, empirically, that enforcing widening the minima of the next task (for the same reason of reducing forgetting on itself) inhibits additionally forgetting for the first task (see Table 1; not stable/plastic means relying on training regimes that encourage/do not encourage wide minima. We empirically estimate the width of minima as well, see appendix C for details). We make the observation that the width of the minima (norm of the eigenvalues) correlates with the norm of the weights. Hence the solutions in the stable learning regime tend to be closer to $\mathbf{0}$, which automatically decrease $\|\Delta w\|$.

Table 1: Disentangling the forgetting on Permuted MNIST. Details are left to appendix C.

Task 1	**Task 2**	**Forgetting (F1)**
Stable	Stable	1.61 ± (0.48)
Stable	Plastic	4.77 ± (1.72)
Plastic	Stable	12.45 ± (1.58)
Plastic	Plastic	19.37 ± (1.79))

Additionally, $\|\Delta w\|$ relates to λ_2^{max} also due to typical learning terminating near a minima, rather than at the minima. Refer to fig. 2b for an illustration. Specifically, the convergence criterion is usually satisfied in the ϵ-neighborhood of w_2^*. If we write the second order Taylor approximation of L_2 around w_2^*, we get:

$$L_2(\hat{w}_2) - L_2(w_2^*) \approx \frac{1}{2}(\hat{w}_2 - w_2^*)^\top \nabla^2 L_2(\hat{w}_2)(\hat{w}_2 - w_2^*) \leq \frac{1}{2}\lambda_2^{max} \|w_2^* - \hat{w}_2\|^2 \leq \epsilon, \quad (6)$$

where, the first equality holds since $\nabla L_2(w_2^*) = 0$. Thus, by decreasing λ_2^{max}, $\hat{w}_2$ can be reached further from w_2^* since the ϵ-neighborhood is larger, and closer to w_1^*. A more formal analysis is given in the appendix. Note that as we enforce the error on task 2 to be lower, the argument above weakens. In the limit, if we assume you converged on task 2, the distance does not depend on curvature, just $\|w_1^* - w_2^*\|$. However, the choice of which minima w_2^* learning prefers will still affect the distance, and as argued above, if wider minima tend to be closer to 0, then they tend to be closer to each other too. Collating all of these observations together we propose the following hypothesis:

Hypothesis. *The amount of forgetting that a neural network exhibits from learning the tasks sequentially, correlates with the geometrical properties of the convergent points. In particular, the wider these minima are, the less forgetting happens.*

We empirically verify the relationship between forgetting and the upper bound derived in E.q. (5). We approximate the Hessian with the largest eigenvalue of the loss function. The results in two common continual learning benchmarks is shown in Figures 2c and 2d. In the figure, the dots represent different neural network training regimes with different settings (e.g., with and without dropout, with and without learning rate decay, different initial learning rates, different batch sizes, different random initialization). See section 4 to find out how these techniques can lead to different loss geometries. All of the models have roughly 90% accuracy on task 2. We can see that our derived measure has high correlation with the forgetting.

4 Training Regimes: techniques affecting stability and forgetting

In this section, we describe a set of widely used techniques that are known to affect the width of the minima (eigenspectrum of Hessian) as well as the length of the path taken by learning ($\|\Delta w\|$). These observations had been generally made with respect to improving generalization, following the wide/narrow minima hypothesis. Based on the argumentation of the previous section, we believe these techniques can have an important role in affecting forgetting as well. In the following section, we will validate this through solid empirical evidence that agrees with our stated hypothesis.

4.1 Optimization setting: learning rate, batch size, and optimizer

There has been a large body of prior work studying the relationship between the learning rate, batch size, and generalization. One common technique of analysis in this direction is to measure the largest eigenvalues of the Hessian of the loss function, which quantifies the local curvature around minima [39, 16, 33, 32, 33]. Followed by the work by Keskar et al. [39], several other papers studied the correlation between minima wideness and generalization [32, 51, 33].

The learning rate and batch size influence both the endpoint curvature and the whole trajectory [77, 17, 45]. A high learning rate or a small batch size limits the maximum spectral norm along the path found by SGD from the beginning of training [33]. This is further studied by Jastrzebski et al. [34], showing that using a large learning rate in the initial phase of training reduces the variance of the gradient, and improves the conditioning of the covariance of gradients which itself is close to the Hessian in terms of the largest eigenvalues [81].

Although having a higher learning rate tends to be helpful since it increases the probability of converging to a wider minima [34], considering a continual optimization problem, we can see it has another consequence: it contributes to the rate of change (i.e., Δw in (5)). Using a higher learning rate means applying a higher update to the neural network weights. Therefore, since the objective function changes thorough time, having a high learning rate is a double-edged sword. Intuitively speaking, decreasing the learning rate across tasks prevents the parameters from going far from the current optimum, which helps reduce forgetting. One natural solution could be to start with a high initial learning rate for the first task to obtain a wide and stable minima. Then, for each subsequent task, slightly decrease the learning rate but also decrease the batch-size instead, as suggested in [69].

Regarding the choice of an optimizer, we argue for the effectiveness of SGD in continual learning setting compared to adaptive optimizers. Although the adaptive gradient methods such as Adam [40] tend to perform well in the initial phase of training, they are outperformed by SGD at later stages of training due to generalization issues [38, 10]. Wilson et al. [76] show that even for a toy quadratic problem, Adam generalizes provably worse than SGD. Moreover, Ge et al. [19] study the effectiveness of exponentially decaying learning rate and show that in its final iterate, it can achieve a near-optimal error in stochastic optimization problems.

Connection to continual learning. The effect of learning rate and batch-size has not been directly studied in the context of continual learning, to the extent of our knowledge. However, we find that the reported hyper-parameters in several works match our findings. Using a small batch size is very common across the continual learning methods. OGD [14] uses a small batch size of 10, similar to several other works [9, 7, 8]. EWC [41] uses a batch size of 32 and also the learning rate decay of 0.95. iCaRL [61] starts from a large learning rate and decays the learning at certain epochs exponentially by a factor of 0.2, while it uses a larger batch size of 128. Finally, PackNet [49] also reports using a learning rate decay by a factor of 0.1. When it comes to choosing the optimizer, the literature mostly prefers SGD with momentum over the adaptive gradient methods. With the exception of VCL [56], which uses Adam, several other algorithms such as A-GEM, OGD, EWC, iCaRL, and PackNet use the standard SGD.

4.2 Regularization: dropout and weight decay

We relate the theoretical insights on dropout and L_2 regularization (weight decay) to our analysis in the previous section. We first argue for the effectiveness of dropout, and then we discuss why L_2 regularization might hurt the performance in a continual learning setting.

Dropout [27] is a well-established technique in deep learning, which is also well-studied theoretically [3, 70, 26, 18, 74]. Wei et al. [74] showed that dropout has both implicit and explicit but entangled regularization effects: More specifically, they provided an approximation for the explicit effect on the i-th hidden layer (denoted by h_i) under input x by: $(p/p-1)[\nabla^2_{h_i} L]^T\ [\text{diag}(h_i^2)]$, where p is the dropout probability, L is the loss function, and $\text{diag}(v)$ is a diagonal matrix of vector v. This term encourages the flatness of the minima since it minimizes the second derivative of the loss with respect to the hidden activation (that is tightly correlated with the curvature with respect to weights). Thus, dropout regularization reduces the R.H.S of Eq. (5). Note that by regularizing the activations norm, it also pushes down the norm of the weights, hence encouraging to find a minima close to 0, which in turn could reduce the norm of Δw. Intuitively, we can understand this effect also from the tendency of dropout to create redundant features. This will reduce the effective number of dimensions of the representation, increasing the number of small-magnitude eigenvalues for the Hessian. As a consequence, gradient updates of the new tasks are less likely to lie on the space spanned by significant eigendirections of previous losses, which results in lesser forgetting.

With respect to L_2 regularization, while intuitively it should help, we make two observations. First, dropout is data-dependent, while L_2 is not. That means balancing the effect of regularization with learning is harder, and in practice, it seems to work worse (both for the currently learned task and for reducing forgetting while maintaining good performance on the new task). Secondly, when combined with Batch Normalization [30], L_2 regularization leads to an Exponential Learning Rate Schedule [47], meaning that in each iteration, multiplying the initial learning rate by $(1 + \alpha)$ for some $\alpha > 0$ that depends on the momentum and weight decay rate. Hence, using L_2 regularization is equivalent to increasing the learning rate overall, potentially leading to larger displacements Δw.

Connection to continual learning. To the best of our knowledge, the work by Goodfellow et al. [22] is the first to empirically study the importance of the dropout technique in the continual learning setting. They hypothesize that dropout increases the optimal size of the network by regularizing and constraining the capacity to be just barely sufficient to perform the first task. However, by observing some inconsistent results on dissimilar tasks, they suggested dropout may have other beneficial effects too. Very recently, [54] studied the effectiveness of dropout for continual learning from the gating perspective. Our work extends their analysis in a more general setting by studying the regularization effect of dropout and its connection to loss landscape. Finally, [43] conducted a comprehensive empirical study on the effect of weight decay and dropout on the continual learning performance and reported that the model consistently benefits from dropout regularization as opposed to weight decay which results in increased forgetting and lower performance on the final model.

5 Experiments and results

In this section, after explaining our experimental setup, we show the relationship between the curvature of the loss function and the amount of forgetting. We use the terms *Stable* and *Plastic (Naive)* to distinguish two different training regimes. The stable network (or stable SGD) exploits the dropout regularization, large initial learning rate with exponential decay schedule at the end of each task, and small batch size, as explained in Sec. 4. In contrast, the plastic (naive) SGD model does not exploit these techniques. In the second experiment, we challenge the stable network and compare it to various state of the art methods on a large number of tasks and more difficult benchmarks. In Appendix C.8, we will show that the stable training regime can be integrated into other continual learning methods and improve their performance significantly.

5.1 Experimental setup

Here, we discuss our experimental methodologies. The decisions regarding the datasets, network architectures, continual learning setup (e.g., number of tasks, training epochs per task), hyper-parameters, and evaluation metrics are chosen to be consistent with several other studies [7, 9, 8, 14], making it easy to compare our results. For all experiments, we report the average and standard deviation over five runs, each with a different random seed. For brevity, we include the detailed hyper-parameters, the code, and instructions for reproducing the results in the supplementary file.

Datasets. We perform our experiments on three standard continual learning benchmarks: Permuted MNIST [22], Rotated MNIST, and Split CIFAR-100. While we agree with [15] regarding the drawbacks of Permuted MNIST in continual learning settings, we believe for consistency with other studies, it is essential to report the results on this dataset as well. Moreover, we report our results on Rotated MNIST and CIFAR-100 that are more challenging and realistic datasets for continual learning benchmarks, once the number of tasks is large. Each task of permuted MNIST is generated by random shuffling of the pixels of images such that the permutation is the same for the images of the same task, but different across different tasks. Rotated MNIST is generated by the continual rotation of the MNIST images where each task applies a fixed random image rotation (between 0 and 180 degrees) to the original dataset. Split CIFAR-100 is a variant of the CIFAR-100 where each task contains the data from 5 random classes (without replacement) out of the total 100 classes.

Models. In our first experiment (Sec. 5.2), we evaluate the continual learning performance over five sequential tasks to provide fine-grained metrics for each task. For this experiment, we use a feed-forward neural network with two hidden layers, each with 100 ReLU neurons and use the deflated power iteration for computing eigenvalues [21]. For the second experiment (Sec. 5.3), we scale the experiments to 20 tasks and use a two-layer network with 256 ReLU neurons in each layer for MNIST datasets, and a ResNet18, with three times fewer feature maps across all layers for CIFAR experiments. These architectures have been previously chosen in several studies [7, 9, 14, 8].

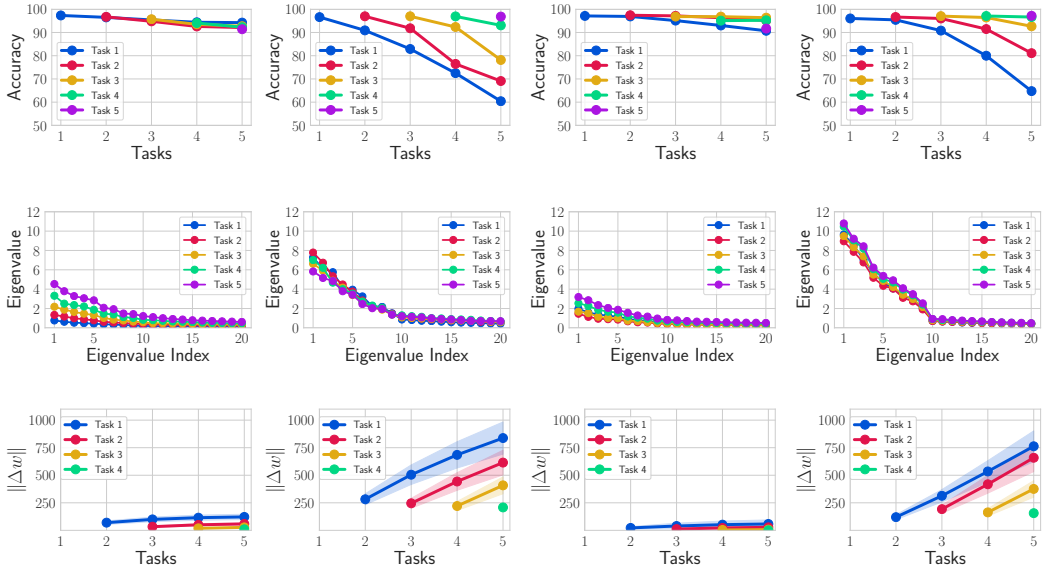
Evaluation. We use two metrics from [6, 7, 9] to evaluate continual learning algorithms when the number of tasks is large.
(1) Average Accuracy: The average validation accuracy after the model has been trained sequentially up to task t, defined by:

$$\mathbf{A_t} = \frac{1}{t}\sum_{i=1}^{t} a_{t,i} \tag{7}$$

where, $a_{t,i}$ is the validation accuracy on dataset i when the model finished learning task t.
(2) Average Forgetting: The average forgetting after the model has been trained sequentially on all tasks. Forgetting is defined as the decrease in performance at each of the tasks between their peak accuracy and their accuracy after the continual learning experience has finished. For a continual learning dataset with T sequential tasks, it is defined by:

$$\mathbf{F} = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t\in\{1,\dots,T-1\}} \left(a_{t,i} - a_{T,i}\right) \tag{8}$$

To prevent confusion, we note that this definition of forgetting is different from what we studied in Section 3. Here, the average forgetting is an evaluation metric that is computed from validation accuracy of the model.



(a) Permuted - Stable (b) Permuted - Plastic (c) Rotated - Stable (d) Rotated - Plastic

Figure 3: Comparison of training regimes for MNIST datasets: **(Top)** *Evolution of validation accuracy for each task*: stable networks suffer less from catastrophic forgetting. **(Middle)** *The spectrum of the Hessian for each task*: the eigenvalues in stable networks are significantly smaller. **(Bottom)** *The degree of parameter change*: in stable networks,the parameters remain closer to the their initial values after learning each task.

5.2 Stable versus Plastic networks

Here, we verify the significance of the training regime in continual learning performance (Sec 3.) and demonstrate the effectiveness of the stability techniques (Sec 4.) in reducing the forgetting.

Each row of Figure 3, represents one of three related concepts for each training regime on each dataset. First, the top row shows the evolution of accuracy on validation sets of each task during the continual learning experience. For instance, the blue lines in this row show the validation accuracy of task 1 throughout the learning experience. In the Middle row, we show the twenty sharpest eigenvalues of the curvatures of each task. In the bottom row, we measure the ℓ_2 distance of network parameters between the parameters learned for each task, and the parameters learned for subsequent tasks.

Aligned with our analysis in Section 3, we show that in contrast to the *plastic* regime, the *stable* training reduces the catastrophic forgetting (Fig. 3 (Top)) thanks to (1) decreasing the *curvature* (Fig. 3 (Middle)) and (2) shrinking the change of parameters (Fig. 3 (Bottom)).

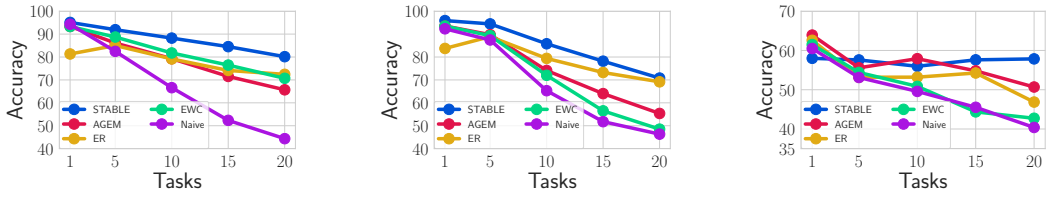
5.3 Comparison with other methods

In this experiment, we show that the stable network is a strong competitor for various continual learning algorithms. In this scaled experiment, we increase the number of tasks from 5 to 20, and provide results for Split CIFAR-100, which is a challenging benchmark for continual learning algorithms. The episodic memory size for A-GEM and ER-Reservoir is limited to be one example per class per task (i.e., 200 examples for MNIST experiments and 100 for CIFAR-100), similar to [9, 8]. To have a consistent naming with other studies, in this section, we use the word "Naive" to describe a plastic network in our paper. To evaluate each algorithm, we measure the average accuracy and forgetting (i.e., A_t and F in Sec. 5.1).

Table 2 compares these metrics for each method once the continual learning experience is finished (i.e., after learning task 20). Moreover, Fig. 4 provides a more detailed picture of the average accuracy during the continual learning experience. To show that stable networks suffer less from catastrophic forgetting, we provide a comparison of the first task's accuracy in the appendix.

Table 2: Comparison of the average accuracy and forgetting of several methods on three datasets.

Method	Memoryless	Permuted MNIST		Rotated MNIST		Split CIFAR100	
		Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Naive SGD	✓	44.4 (±2.46)	0.53 (±0.03)	46.3 (±1.37)	0.52 (±0.01)	40.4 (±2.83)	0.31 (±0.02)
EWC	✓	70.7 (±1.74)	0.23 (±0.01)	48.5 (±1.24)	0.48 (±0.01)	42.7 (±1.89)	0.28 (±0.03)
A-GEM	✗	65.7 (±0.51)	0.29 (±0.01)	55.3 (±1.47)	0.42 (±0.01)	50.7 (±2.32)	0.19 (±0.04)
ER-Reservoir	✗	72.4 (±0.42)	0.16 (±0.01)	69.2 (±1.10)	0.21 (±0.01)	46.9 (±0.76)	0.21 (±0.03)
Stable SGD	✓	**80.1 (±0.51)**	**0.09 (±0.01)**	**70.8 (±0.78)**	**0.10 (±0.02)**	**59.9 (±1.81)**	**0.08 (±0.01)**
Multi-Task Learning	N/A	86.5 (±0.21)	0.0	87.3(±0.47)	0.0	64.8(±0.72)	0.0



(a) Permuted MNIST (b) Rotated MNIST (c) Split CIFAR-100

Figure 4: Evolution of the average accuracy during the continual learning experience with 20 tasks

While our stable network performs consistently better than other algorithms, we note that our proposed techniques are orthogonal to other works and can also be incorporated in them, as we show in Appendix C.8

6 Conclusion

In this work, we have revisited the catastrophic forgetting problem from loss landscapes and optimization perspective and identify learning regimes and training techniques that contribute to the forgetting. The analytical insights yielded a series of effective and practical techniques that can reduce forgetting and increase the stability of neural networks in maintaining previous knowledge.

We have studied these techniques through the lens of optimization by studied the wideness of the loss surfaces around the local minima. However, they might have other confounding factors for reducing catastrophic forgetting as well. We call for more theoretical research to further their role in demystifying trading off the stability plasticity dilemma and its effect on continual learning.

Finally, we have empirically observed that these simple techniques proved to be more effective than some of the recent approaches (e.g., regularization based methods, or memory-based methods) but are orthogonal to them in the sense that our practical recommendations and provided insights on loss perspective can be incorporated to them.

Broader Impact

Continual Learning aims for effectively training a model from sequential tasks while making sure the model maintains a reasonable performance on the previous ones. It's an integral part of Artificial General Intelligence (AGI) that reduces the cost of retraining (time, computation, resources, energy) and mitigates the need for storing all previous data to respect users' privacy concerns better. Reducing catastrophic forgetting may potentially risk privacy for data that are explicitly wanted to be forgotten. This calls for more future research into formalizing and proposing continual learning agents that allow the identifiable parts of data to be forgotten, but the general knowledge is maintained. The research presented in this paper can be used for many different application areas and a particular use may have both positive or negative implications. Besides those, we are not aware of any immediate short term negative impact.

Acknowledgment

SIM and HG acknowledge support from the United States National Science Foundation through grant CNS-1750679. The authors thank Anonymous Reviewers, Jonathan Schwarz, Sepehr Sameni, Hooman Shahrokhi, and Mohammad Sadegh Jazayeri for their valuable comments and feedback.

References

[1] 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.

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Online continual learning with no task boundaries. *arXiv preprint arXiv:1903.08671*, 2019.

[3] Pierre Baldi and Peter J Sadowski. Understanding dropout. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2814–2822. Curran Associates, Inc., 2013.

[4] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[5] Mehdi Bennani and M. Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *ArXiv*, abs/2006.11942, 2020.

[6] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

[7] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2018.

[8] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip H. S. Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *ArXiv*, abs/2002.08165, 2019.

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. 2019.

[10] Jinghui Chen and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *ArXiv*, abs/1806.06763, 2018.

[11] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204, 2015.

[12] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

[13] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*, 2019.

[14] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. *ArXiv*, abs/1910.07104, 2019.

[15] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *ArXiv*, abs/1805.09733, 2018.

[16] Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. *ArXiv*, abs/1910.05929, 2019.

[17] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

[18] Hongchang Gao, Jian Pei, and Heng Huang. Demystifying dropout. In *ICML*, 2019.

[19] Rong Ge, Sham M. Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure. *ArXiv*, abs/1904.12838, 2019.

[20] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[21] Noah Golmant, Amir Yao Zhewei, Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigentings: efficient pytorch hessian eigendecomposition, October 2018. URL `https://github.com/noahgolmant/pytorch-hessian-eigenthings`.

[22] Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. *CoRR*, abs/1312.6211, 2013.

[23] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[24] Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *ICLR 2018*, 2018.

[25] Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *arXiv preprint arXiv:1906.05201*, 2019.

[26] David P. Helmbold and Philip M. Long. Surprising properties of dropout in deep networks. *J. Mach. Learn. Res.*, 18:200:1–200:28, 2016.

[27] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

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

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

[31] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[32] Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in sgd. *ArXiv*, abs/1711.04623, 2018.

[33] Stanislaw Jastrzebski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. In *ICLR*, 2018.

[34] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on the optimization trajectories of deep neural networks. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

[35] Ghassen Jerfel, Erin Grant, Thomas L. Griffiths, and Katherine A. Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *NeurIPS*, 2019.

[36] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

[37] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[38] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *ArXiv*, abs/1712.07628, 2017.

[39] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017.

[40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[41] James N Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, and et. al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526, 2017.

[42] Soheil Kolouri, Nicholas Ketz, Xinyun Zou, Jeffrey Krichmar, and Praveen Pilly. Attention-based structural-plasticity. *arXiv preprint arXiv:1903.06070*, 2019.

[43] Matthias Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ale Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *ArXiv*, abs/1909.08383, 2019.

[44] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Neural information processing systems*, pages 4652–4662, 2017.

[45] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[46] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*, 2019.

[47] Zhongyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *ArXiv*, abs/1910.07454, 2019.

[48] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[49] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2017.

[50] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences of the United States of America*, 115 44, 2018.

[51] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[52] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. 1989.

[53] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. In *Front. Psychol.*, 2013.

[54] Seyed Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[55] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and H. Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *ArXiv*, abs/2010.04495, 2020.

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

[57] Cuong V Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091*, 2019.

[58] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. 2018.

[59] Adam Paszke, Sam Gross, Francisco Massa, and et. al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[60] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 7645–7655, 2019.

[61] Sylvestre-Alvise Rebuffi, Alexander I Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2016.

[62] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[63] Amanda Rios and Laurent Itti. Closed-loop gan for continual learning. *arXiv preprint arXiv:1811.01146*, 2018.

[64] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748, 2018.

[65] 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.

[66] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[67] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4535–4544, 2018.

[68] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[69] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *ArXiv*, abs/1711.00489, 2017.

[70] Nitish Srivastava. Improving neural networks with dropout. 2013.

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

[72] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[73] Matthew Wallingford, Aditya Kusupati, Keivan Alizadeh-Vahid, Aaron Walsman, Aniruddha Kembhavi, and Ali Farhadi. In the wild: From ml models to pragmatic ml systems. *ArXiv*, abs/2007.02519, 2020.

[74] Colin Wei, Sham M. Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. volume abs/2002.12915, 2020.

[75] Junfeng Wen, Yanshuai Cao, and Ruitong Huang. Few-shot self reminder to overcome catastrophic forgetting. *arXiv preprint arXiv:1812.00543*, 2018.

[76] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *ArXiv*, abs/1705.08292, 2017.

[77] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent escapes from sharp minima exponentially fast. *arXiv preprint arXiv:2002.03495*, 2020.

[78] Jaehong Yoon, Eunho Yang, Jungtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Sixth International Conference on Learning Representations*. ICLR, 2018.

[79] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continuous learning of context-dependent processing in neural networks. *arXiv preprint arXiv:1810.01256*, 2018.

[80] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR, 2017.

[81] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, pages 8196–8207, 2019.

[82] Mengmi Zhang, Tao Wang, Joo Hwee Lim, and Jiashi Feng. Prototype reminding for continual learning. *arXiv preprint arXiv:1905.09447*, 2019.

Supplementary material

In this document, we present the materials that were excluded or summarized due to space limitation in the main text. It is organized as follows:

Appendix A extends our analysis in Section 3 of the main text regarding the forgetting in continual learning.

Appendix B provides further information for our experimental setup and the hyper-parameters used for each experiment. Note that, in addition to this document, we provide our code with scripts to reproduce the results for each experiment.

Appendix C includes additional experiments and results. More specifically, it includes:

- Additional information about Figure 1 such as accuracy on all tasks.
- Extended version of Table 1 with detailed discussion.
- Additional results regarding the norm of parameters in our first experiment (Section. 5.2)
- Additional results regarding the comparison of the first task accuracy for algorithms in our second experiment (Section. 5.3).
- A **new experiment** in Appendix C.8, where we apply the stability techniques for other methods such as A-GEM and EWC. We demonstrate that these methods can enjoy a performance boost if they use a stable training regime.

A Further analysis on forgetting

In this section, we extend our analysis in Section 3 of the main paper.

Let w_1^* and w_2^* still be the optimal points for the tasks while $\hat{w}_1$ and $\hat{w}_2$ are the convergent or (near-) optimum parameters after training has been finished for the first and second task, sequentially. We usually use Stochastic Gradient Descent (SGD) or its many variants to find a low-error plateau for the optimization procedure. We can define the *forgetting* (of the first task) using the convergent point too:

$$F_1 \triangleq L_1(\hat{w}_2) - L_1(\hat{w}_1). \tag{9}$$

We can approximate $L_1(\hat{w}_2)$ and $L_1(\hat{w}_1)$ around w_1^*:

$$L_1(\hat{w}_1) \approx L_1(w_1^*) + (\hat{w}_1 - w_1^*)^\top \nabla L_1(w_1^*) + \frac{1}{2}(\hat{w}_1 - w_1^*)^\top \nabla^2 L_1(w_1^*)(\hat{w}_1 - w_1^*) \tag{10}$$

$$\approx L_1(w_1^*) + \frac{1}{2}(\hat{w}_1 - w_1^*)^\top \nabla^2 L_1(w_1^*)(\hat{w}_1 - w_1^*), \tag{11}$$

where, $\nabla^2 L_1(w_1^*)$ is the Hessian for loss L_1 at w_1^* and the last equality holds because the model is assumed to converge to critical point or it had stopped in a plateau where gradient's norm vanishes, thus $\nabla L_1(w_1^*) \approx \mathbf{0}$. Similarly,

$$L_1(\hat{w}_2) \approx L_1(w_1^*) + \frac{1}{2}(\hat{w}_2 - w_1^*)^\top \nabla^2 L_1(w_1^*)(\hat{w}_2 - w_1^*). \tag{12}$$

Defining $\Delta w = \hat{w}_2 - \hat{w}_1$ as the relocation vector we compute the forgetting according to Eq. (9):

$$F_1 \approx \frac{1}{2}(\hat{w}_2 - w_1^*)^\top \nabla^2 L_1(w_1^*)(\hat{w}_2 - w_1^*) - \frac{1}{2}(\hat{w}_1 - w_1^*)^\top \nabla^2 L_1(w_1^*)(\hat{w}_1 - w_1^*) \tag{13}$$

$$= \frac{1}{2}((\hat{w}_2 - w_1^*) - (\hat{w}_1 - w_1^*)))\nabla^2 L_1(w_1^*)((\hat{w}_2 - w_1^*) + (\hat{w}_1 - w_1^*))) \tag{14}$$

$$\approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1^*)\Delta w, \tag{15}$$

where in the last equality we used $\|\hat{w}_1 - w_1^*\| \ll \|\hat{w}_2 - w_1^*\|$. Under the assumption that the critical point is a minima (or the plateau we get stuck surrounding a minima), we know that the Hessian needs to be positive semi-definite. We can use this property further to bound F_1 as follows:

$$F_1 = L_1(\hat{w}_2) - L_1(\hat{w}_1) \approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1^*)\Delta w \leq \frac{1}{2}\lambda_1^{max} \|\Delta w\|^2, \tag{16}$$

where λ_1^{max} is the maximum eigenvalue of $\nabla^2 L_1(w_1^*)$. Fig. 5a shows how wider L_1 (lower λ_1^{max}) leads to less forgetting.

Controlling the spectrum of the Hessian without controlling the norm of the displacement can not ensure that F_1 is minimized.

Assume the algorithm has already converged to a plateau where gradient's norm vanishes for L_1 and stopped at $\hat{w}_1$ and is about to optimize L_2. The weights are updated according to $\eta\nabla L_2(w)$ iteratively until a fixed number of iterations, or a minimum (validation) loss, or minimum gradient magnitude is achieved. We show that $\|\Delta w\|$ is related to λ_2^{max} (Refer to fig. 5b for an illustration). Using the triangle inequality we write:

$$\|\Delta w\| = \|\hat{w}_2 - \hat{w}_1\| \geq \|w_2^* - \hat{w}_1\| - \|\hat{w}_2 - w_2^*\|\,. \tag{17}$$

Since $\|w_2^* - \hat{w}_1\|$ is constant we only need to bound $\|\hat{w}_2 - w_2^*\|$. We examine two different convergence criterion.

For the case $L_2(\hat{w}_2) - L_2(w_2^*) \leq \epsilon$ is the convergence criterion, we write the second order Taylor approximation of L_2 around w_2^*:

$$L_2(\hat{w}_2) - L_2(w_2^*) \approx (\hat{w}_2 - w_2^*)^\top \nabla L_2(w_2^*) + \frac{1}{2}(\hat{w}_2 - w_2^*)^\top \nabla^2 L_2(\hat{w}_2)(\hat{w}_2 - w_2^*). \tag{18}$$

Again, $\nabla L_2(w_2^*) = 0$ and the first term in the R.H.S. is dismissed. Moreover the second term in the R.H.S. can be upper bounded by $\frac{1}{2}\lambda_2^{max}\left\|w_2^* - \hat{w}_2\right\|^2$. Therefore, one can write the convergence criterion as

$$\frac{1}{2}\lambda_2^{max}\left\|w_2^* - \hat{w}_2\right\|^2 \leq \epsilon \Longrightarrow \|w_2^* - \hat{w}_2\| \leq \frac{2\sqrt{\epsilon}}{\lambda_2^{max}}. \tag{19}$$

Combining the above with (17) we get:

$$\|\Delta w\| \geq C - \frac{2\sqrt{\epsilon}}{\lambda_2^{max}}, \tag{20}$$

where, $C = \|w_2^* - \hat{w}_1\|$ is constant. Therefore, by decreasing λ_2^{max}, the lower bound on the $\|\Delta w\|$ decreases and the near-optimum w_2^* can be reached in a closer distance to w_1^*. Fig. 5b shows this case.

For the case where $\|\nabla L_2\|(\hat{w}_2) \leq \epsilon$ is the convergence criterion we write the first order Taylor approximation of ∇L_2 around w_2^*:

$$\nabla L_2(\hat{w}_2) - \nabla L_2(w_2^*) \approx \nabla^2 L_2(w_2^*)(\hat{w}_2 - w_2^*). \tag{21}$$

Again, $\nabla L_2(w_2^*) = 0$ and the second term in the L.H.S. is dismissed. Moreover, the R.H.S. can be upper bounded by $\lambda_2^{max}\|w_2^* - \hat{w}_2\|$. Therefore, one can write the convergence criterion as

$$\lambda_2^{max}\|w_2^* - \hat{w}_2\| \leq \epsilon \Longrightarrow \|w_2^* - \hat{w}_2\| \leq \frac{\epsilon}{\lambda_2^{max}}. \tag{22}$$

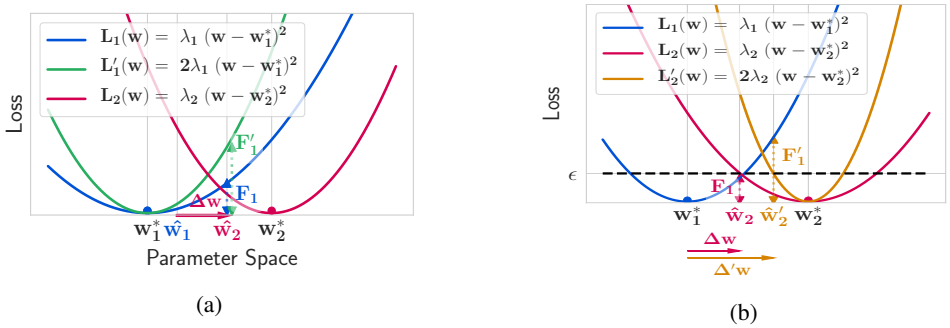
Combining the above with (17) we get:

$$\|\Delta w\| \geq C - \frac{\epsilon}{\lambda_2^{max}}, \tag{23}$$

where, $C = \|w_2^* - \hat{w}_1\|$ is constant. Therefore, by decreasing λ_2^{max}, the lower bound on the $\|\Delta w\|$ decreases and the near-optimum $\hat{w}_2$ can be reached within a smaller distance from $\hat{w}_1$.

In practice, we usually fix the number of gradient update $\eta\nabla L_2(w)$ iterations where the learning rate and the gradient of the loss for task 2, play important roles. First, it's already clear that the smaller the learning rate, the higher the chance of finding $\hat{w}_2$ close to $\hat{w}_1$. More importantly, learning decay helps with wider minima by allowing to start with a large learning rate and imposing an exploration phase that increases the chance of finding a wider minima. Therefore, learning rate decay plays a significant role in reducing catastrophic forgetting.

Second, the magnitude of the update is proportional to $\|\nabla L_2(w)\|$ where is lager for higher curvature not only at the minima but in its neighborhood too. Look at Fig. 5b for a simplistic illustration where we assume that the (near-)optimal points for the low and high curvature achieved lie very close to each other. Again, we can intuitively see that the forgetting is correlated with the curvature around the local minimas.



(a) (b)

Figure 5: **(a)**: For a fixed Δw, the wider the curvature of the first task, the less the forgetting. **(b)**: The wider the curvature of the second task, the smaller $\|\Delta w\|$.

B Experimental Setup Details

B.1 Discussion on our experiment design

We would like to note that one fundamental criterion we take into account in our experiments is to facilitate the verification of our results. We achieve this goal by:

1. Reporting the results on common continual learning datasets used in previous studies: For instance, as mentioned in the main text, the majority of continual learning papers to which we compare our results use Permuted MNIST, Rotated MNIST, and Split CIFAR-100 benchmarks. While we acknowledge the fact that continual learning literature can benefit from benchmarks in other domains beyond computer vision, we agree with the current trend that CIFAR-100 with 20 sequential tasks is challenging enough for continual learning.

2. Using similar architectures with other studies: For our first experiment, we used two-layer MLP on MNIST datasets o 5 tasks. This architecture is used in [14, 74]. For the second experiment on 20 tasks, the Resnet architecture described in the text, was used in [7, 9, 8]. The only change we apply to the architecture was to add dropout layers in residual blocks.

3. Providing metrics for accuracy and forgetting: We believe any continual learning work should provide report both mentioned metrics. We also report these metrics, which are also reported in [7, 8, 9].

4. **Releasing the code.** Please refer to our code repository for further instructions for replicating the results. We have implemented stable SGD with PyTorch [59].

B.2 Continual learning setup

In this section, we review our experimental setup (e.g., number of tasks, number of epochs per task).

In experiment 1 (Section 5.2), our goal was to *elaborate the impact of training regime*. Hence, we found that five tasks and two benchmarks are sufficient for our purpose. Each task in that experiment had 5 training epochs.

In experiment 2 (Section 5.3), we aimed to *demonstrate the performance of stable training regime*, and hence, we chose the setup that is similar to our baselines. We used 20 tasks where each task had 1 training epoch. This setup is used in several studies [7, 9, 8]. We note that we use the task identifiers only in the CIFAR-100 experiment, similar to A-GEM and ER-Reservoir implementations.

In the new experiment in Appendix C.8, the purpose is to *show that the stability techniques can be incorporated into other algorithms* such as A-GEM, EWC, and ER-Reservoir. We use the rotation MNIST with 20 tasks and 1 epoch per task.

Finally, We note that we have used Stochastic Gradient Descent (SGD) with momentum optimizer in all experiments.

B.3 Hyperparameters in experiments

In this section, we report the hyper-parameters we used in our experiments. For other algorithms (e.g., A-GEM, and EWC), we ensured that our hyper-parameters included the optimal values that the original papers reported. We used grid search for each model to find the best set of parameters detailed below[3]:

Parameters for experiment 1

B.3.1 Plastic (Naive)

- initial learning rate: [0.25, 0.1, **0.01**, 0.001]
- batch size: 64

B.3.2 Stable

- initial learning rate: [0.25, **0.1**, 0.01, 0.001]
- learning rate decay: [0.9, 0.75, **0.4**, 0.25]
- batch size: [**16**, 64]
- dropout: [**0.5**, 0.25]

Parameters for experiment 2

B.3.3 Naive

- initial learning rate: [0.25, 0.1, **0.01** (MNIST, CIFAR-100), 0.001]
- batch size: [**10**, 64]

B.3.4 Stable

- initial learning rate: [0.25, **0.15** (CIFAR-100), **0.10** (MNIST), 0.05, 0.01, 0.005]
- learning rate decay: [0.9, **0.85** (CIFAR-100), 0.7, **0.55** (MNIST), 0.5]
- batch size: [**10**, 64]
- dropout: [**0.25** (MNIST), **0.10** (CIFAR-100), 0.0]

B.3.5 EWC

- initial learning rate: [0.25, **0.1** (MNIST, CIFAR-100), 0.01, 0.001]
- batch size: [**10**, 64]
- λ (regularization): [1, **10** (MNIST, CIFAR-100), 100]

B.3.6 AGEM

- initial learning rate: [0.25, **0.1** (MNIST), **0.01** (CIFAR-100), 0.001]
- batch size: [**10**, 64]

B.3.7 ER

- initial learning rate: [0.25, **0.1** (MNIST), **0.01** (CIFAR-100), 0.001]
- batch size: [**10**, 64]

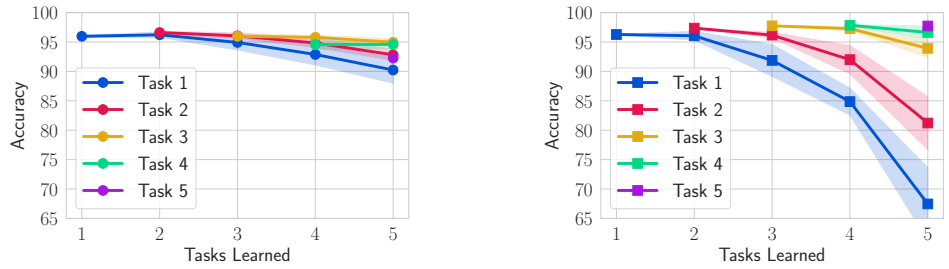
[3]For further instructions for reproducing the results, see our code repository.

C Additional Results

C.1 Detailed accuracy of the networks in Figure 1

Figure 1 in the introduction (Section 1), compares the first task accuracy between the naive SGD and the stable SGD. Here, we provide the accuracy of all tasks for these two training regimes.

Although in the introduction section we did not introduce the "stable training" term, we note that Network 1 (i.e., the stable network), suffers much less from the catastrophic forgetting of previous tasks compared to Network 2 (i.e., the plastic network), thanks to the training regime, and at the cost of a relatively small drop in the accuracy of the current task.



(a) Network 1 (Stable Network)

(b) Network 2 (Plastic Network)

Figure 6: Full results for the comparison of accuracy on all tasks for two networks in Figure 1 of the introduction section.

C.2 Extended version of Table 1 in Section 3: disentangling the stability of different tasks

In this section, we aim to disentangle the role that stable training of the current task 1 plays against stable training of task 2 in decreasing the catastrophic forgetting for task 1.

In table 3, we report the forgetting measure F_1 for four different scenarios: training the first task (with/without) stability techniques and training the second task (with/without) these techniques, respectively. The reported "Stable" networks in this table use dropout probability of 0.25, learning rate decay of 0.4 and a small batch size of 16, while "Plastic" networks do not exploit the dropout regularization and learning rate decay, and set the batch size to 256.

The reported mean and standard deviations are calculated for five different runs with different random seeds. As expected, the case Stable/Stable is the best and Plastic/Plastic is the worst in terms of forgetting. But the interesting observation is the huge difference between Stable/Plastic and Plastic/Stable. This suggests, among other possible explanations, that the wideness of the current task is more important than the wideness of the subsequent tasks for the goal of reducing the forgetting.

Table 3: Disentanglment of forgetting on Permuted MNIST.

Task 1	**Task 2**	**Forgetting (F1)**	$\lambda_1^{\max}$	$\|\Delta \mathbf{w}\|$
Stable	Stable	$1.61 \pm (0.48)$	$2.19 \pm (0.12)$	$73.1 \pm (1.23)$
Stable	Plastic	$4.77 \pm (1.72)$	$2.21 \pm (0.18)$	$174.4 \pm (7.66)$
Plastic	Stable	$12.45 \pm (1.58)$	$7.72 \pm (0.19)$	$74.7 \pm (2.63)$
Plastic	Plastic	$19.37 \pm (1.79)$	$7.73 \pm (0.22)$	$178.4 \pm (8.58)$

C.3 Additional result for experiment 1: comparing norms of weights

In section 4.2, we discussed that dropout pushes down the norm of the weights by regularizing the activations. To support this argument, in Figure 7 we compare the norm of the weights for stable and plastic networks in experiment 1. For the stable network the norm of the optimal solutions after training each task is smaller and this might be the reason for smaller displacement in the sequential

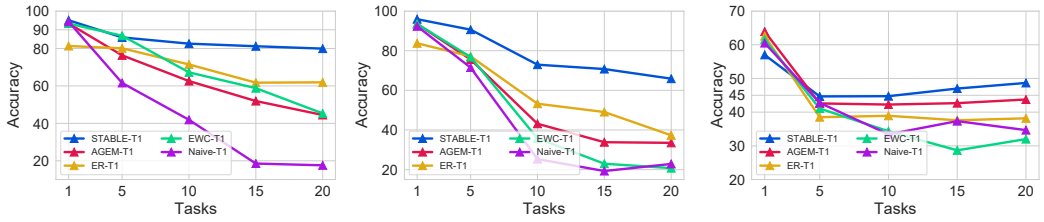
optimizations for tasks. Further analysis and theoretical justification of this phenomena is beyond the scope of the current paper and is left as an interesting future work.

(a) Permuted MNIST (b) Rotation MNIST

Figure 7: Comparing the norm of parameters for networks in experiment 1

C.4 Additional result for experiment 2: comparison of the first task accuracy

In Section 5.3, we provided the plots for the evolution of average accuracy during the continual learning experience. Here, we measure the validation accuracy of the first task during the learning experience with 20 tasks in Figure 8. The figure reveals that the stable network remembers the first task much better than other methods.



(a) Permuted MNIST (b) Rotated MNIST (c) Split CIFAR-100

Figure 8: Evolution of the first task accuracy

C.5 Additional result for experiment 2: CIFAR-100 with more epochs

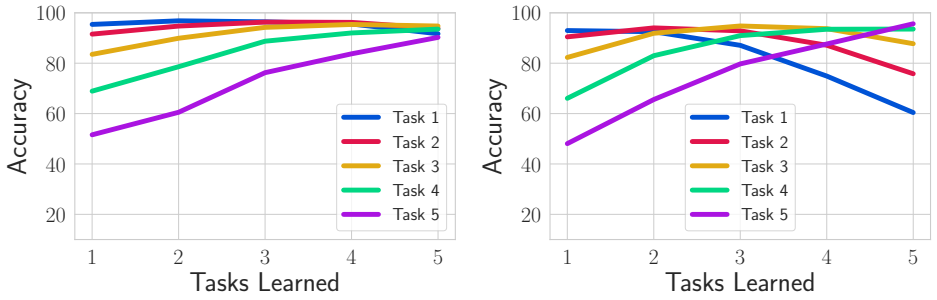
In this experiment, we extend the number of epochs for the CIFAR-100 benchmark. Table 4 shows that the accuracy of stable-SGD increases as the number of epochs per task increases. However, we note that the single epoch setting would be more realistic as it is closer to the online, real-world setups.

Table 4: Comparison of stable-SGD performance on CIFAR-100 benchmark with different number of epochs per task

	Average Accuracy (%)	Forgetting
1 epochs	59.9 ($\pm$ 1.81)	0.08 ($\pm$ 0.01)
5 epochs	64.9 ($\pm$ 0.87)	0.13 ($\pm$ 0.01)
10 epochs	69.7 ($\pm$ 0.74)	0.12 ($\pm$ 0.01)

C.6 Measuring Forward Transfer

Although our main focus in this work is on the catastrophic forgetting and the "backward transfer", we include the forward-transfer on rotation MNIST with 5 tasks. We see the forgetting for stable net is much less by compromising a negligible amount of forward transfer. This is in line with our discussion on stability-plasticity dilemma.



(a) Stable Network

(b) Plastic Network

Figure 9: Comparison of the forward transfer in stable and plastic networks on rotation MNIST with 5 tasks.

C.7 New experiment: comparison with orthogonal gradient descent (OGD)

The idea behind orthogonal gradient descent (OGD) [14] is to change the gradient updates to be perpendicular to the space spanned by the gradient vectors of previous tasks with the assumption that these orthogonal updates will change the output of the network minimally. Bennani and Sugiyama [5] shows that the OGD updates can only be effective across a single task and propose OGD+ that it is robust to catastrophic forgetting across an arbitrary number of tasks with tighter generalisation bounds.

However, in practice, we find that OGD is not a strong competitor to stable-SGD. In Table 5 and Table 6, we compare the accuracy of each tasks once the network trained continually on five tasks of rotated-MNIST and permuted-MNIST, respectively. We directly report the numbers from the OGD paper as our setting was exactly the same. Moreover, we note that the reported numbers for stable-SGD are calculated over five runs, each with different seed.

Method	Task 1	Task 2	Task 3	Task 4	Task 5
OGD	75.6	86.6	91.7	94.3	93.4
Stable-SGD	91.5	92.1	95.02	94.2	90.9

Table 5: Comparison of OGD and stable-SGD on rotated MNIST with 5 tasks

Method	Task 1	Task 2	Task 3	Task 4	Task 5
OGD	79.5	88.9	89.6	91.8	92.4
Stable-SGD	94.2	92.1	92.7	92.3	91.4

Table 6: Comparison of OGD and stable-SGD on Permuted MNIST with 5 tasks

C.8 New experiment: stabilizing other methods

One important question that deserves further investigation is ***"Can other methods benefit from the stable training regime?"***. In this section, we show that the answer is yes.

For the rotation MNIST dataset with 20 tasks, we use similar architecture and hyper-parameters for SGD, EWC, A-GEM, and ER-Reservoir as described in Appendix B. To make these models stable, we add dropout (with the dropout probability 0.25), keep the batch size small, and use a decay factor of 0.65 to decrease the learning rate at the end of each task.

Table 7 shows that by merely stabilizing the training regime, we can improve the average accuracy of EWC, A-GEM, and ER-Reservoir by 12.9%, 15.8% and 9%, respectively. Besides, Figure 10 shows the evolution of the average accuracy throughout the learning experience. As expected, the episodic memory in stable A-GEM and stable ER-Reservoir helps these methods to outperform Stable SGD, which does not use any memory. However, as noted before, stable SGD outperforms these complex methods in their original form when they do not employ stabilization techniques.

Table 7: Stabilizing other methods: the average accuracy and forgetting on rotated MNIST with 20 tasks.

Method	**Average Accuracy**	**Forgetting**
SGD	46.3 (± 1.37)	0.52 (± 0.01)
Stable SGD	70.8 (± 0.78)	0.10 (± 0.02)
EWC	48.5 (± 1.24)	0.48 (± 0.01)
Stable EWC	61.4 (± 1.15)	0.30 (± 0.01)
AGEM	55.3 (± 1.47)	0.42 (± 0.01)
Stable AGEM	71.1 (± 1.06)	0.13 (± 0.01)
ER-Reservoir	69.2 (± 1.10)	0.21 (± 0.01)
Stable ER-Reservoir	78.2 (± 0.74)	0.09 (± 0.01)

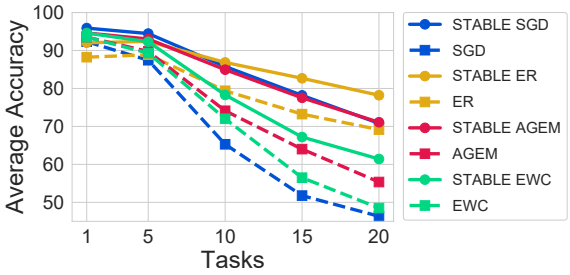


Figure 10: Evolution of average accuracy