SAME ACCURACY, TWICE AS FAST: CONTINUAL LEARNING SURPASSES RETRAINING FROM SCRATCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning aims to enable models to adapt to new datasets without losing performance on previously learned data, often assuming prior data is no longer available. However, in many practical scenarios, both old and new data are accessible. In such cases, good performance on both datasets is typically achieved by abandoning the model trained on the previous data and re-training a new model from scratch on both datasets. This training from scratch is computationally expensive. In contrast, methods that leverage the previously trained model are worthy of investigation as they could significantly reduce computational costs. Our evaluation framework quantifies the computational savings of such methods while maintaining or exceeding the performance of training from scratch. We identify key optimization aspects – initialization, regularization, data selection, and hyper-parameters – that can each contribute to reducing computational costs. For each aspect, we propose effective first-step methods that already yield substantial computational savings. By combining these methods, we achieve up to 2.7x reductions in computation time across various computer vision tasks, highlighting the potential for further advancements in this area.

1 INTRODUCTION

In modern deep learning, the available data tends to expand and evolve over time, often resulting in a model already trained on a large dataset (‘*old data*’) to be further adapted to perform well also on a new, smaller dataset (‘*new data*’). This new data typically introduces slightly different distributions, such as additional classes, new domains, or corner cases. A common approach is to retrain the model from scratch using both the old and new datasets, but as model and dataset sizes increase, it becomes computationally expensive. This highlights the need for more efficient methods that can continuously train models without the cost of full retraining.

A large part of continual learning research has tried to approach this problem in a resource-constrained setting, aiming to reach the highest possible performance on both old and new data under some constraints (Verwimp et al., 2024). Such constraints can lead to suboptimal performance, in part due to catastrophic forgetting (French, 1999). In e.g. industry applications, it is often undesirable to sacrifice performance to save computational costs, and retraining from a randomly initialized model (‘*from scratch*’) is preferred over using older models (Huyen, 2022). This is a wasteful approach; there is a model available that performs well on the old data, so why not use it? In practice, it has been shown to be difficult to continue to train previously trained models, even without storage constraints on past examples (Ash & Adams, 2020).

In this paper, we focus on the cost of training a model on new data, when a model that performs well on the old data is available. The cost of training this old model is treated as a sunk cost (Kahneman & Tversky, 1972), its training happened in the past and the price for this training has already been paid. In contrast, future costs can be reduced or mitigated, and we show that using old models can be an effective way of doing so. Typically, in computational problems there is both a memory and computational aspect, but when models get large, the computational cost tends to outweigh memory costs. For example, the hard disks to store ImageNet21k are about 50 times cheaper than training a large vision transformer on the same data once (see Appendix A.1 for details of this estimate).

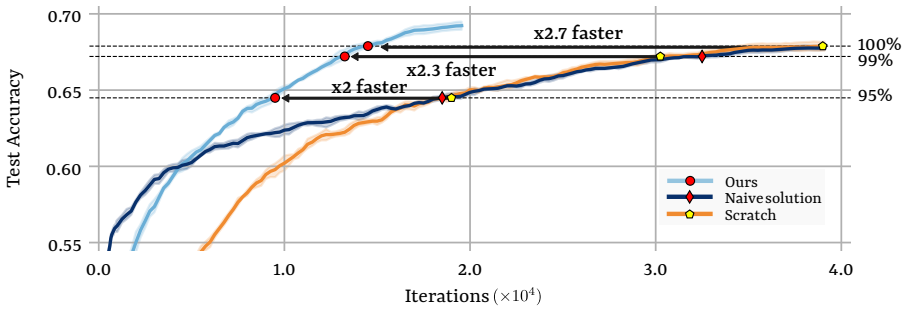

Figure 1: Test accuracy on CIFAR100 (70+30) with a model pre-trained on 70 classes. The 'scratch' method starts from random initialization, while the 'naive' approach uses the pre-trained model without modification. 'Ours' modifies the optimization process (see Section 3) and matches the scratch performance with 2.7× lower computational cost.

While there are good reasons to work in memory-restricted settings (Verwimp et al., 2024), in this paper the focus is on reducing the computational cost when new data becomes available. Particularly, we investigate techniques that can make a difference when an old model is available, instead of general optimization techniques like e.g. mixed precision calculations (Micikevicius et al., 2018).

Instead of the wasteful from-scratch approach, we propose to focus on continuous solutions, which leverage the existing model when new data is available. The simplest of these solutions is to continue training the model on the full dataset, using the previous model's weights for initialization. While this solution uses pre-trained initialization, it converges at a similar speed and often underperforms compared to training from scratch, offering little computational gain. Figure 1 illustrates the limitations of this naive approach.

Here, we introduce an evaluation framework to measure the computational gain of continuous methods. In this framework, we assume access to a previously trained model on the old data, and both the entire new and the old datasets. The goal is to improve the efficiency and accuracy by leveraging this previously trained model, compared to training from scratch on the entire dataset, as well as naive continuous training. We measure the advantage of an approach by the reduction in computational cost required to reach the same accuracy, on the full dataset, as a model trained from scratch.

In Section 3, we start from the canonical SGD update rule and show that all of its aspects – the initialization, the objective function, the sampling process, and the hyper-parameters – can significantly reduce computational costs. We outline and evaluate several strategies – inspired by the literature – that act on these aspects. Each of them presents a promising avenue for future research to accelerate the convergence of continuous models. In Section 4, we show that while these methods individually reduce computational costs, their effects are to some extent complementary; combining them yields even greater reductions. Finally, we further demonstrate that these methods improve training efficiency across a variety of image classification datasets, multi-task settings, and domain incremental scenarios, highlighting the robustness and potential of our method to reduce the computational burden of re-training machine learning models.

1.1 Our contribution

- We propose a novel continual learning evaluation framework, allowing models full access to previous data and measuring their computational cost rather than only their accuracy. This approach shifts the research focus towards more practical, real-world challenges.
- We discuss the different areas in the optimization process that can be explored to accelerate convergence. Each area is broad enough to be the focus of a different method.
- For each area, we introduce a first-step method based on the literature, which already significantly enhances the learning speed of continual models.
- We demonstrate that improvements in these areas are complementary, meaning advancements in one area do not preclude further gains in others.
- Through an empirical study, we show that our combined methods consistently accelerate convergence across multiple datasets and tasks.

1.2 Related Work

Continual learning. Continual learning concerns cases where data is not available all at once (see e.g. (De Lange et al., 2021; Wang et al., 2024) for surveys). Most studies have focused on settings with strong constraints on the amount of old data that can be stored (Verwimp et al., 2024). Replay methods (Chaudhry et al., 2019; Buzzega et al., 2020) aim to use this stored data as effectively as possible. However, even with replay mechanisms, learning new data often leads to (catastrophic) forgetting of previously learned examples. Other approaches modify the model architectures (e.g. Yan et al., 2021) and regularization losses (e.g. Li & Hoiem, 2017) to reduce forgetting. In contrast, some works have looked at settings where computational cost is restricted rather than memory costs. In these cases, standard replay outperforms other methods (Prabhu et al., 2023). Later works (Smith et al., 2024; Harun et al., 2023) improved replay techniques in these settings. Our work, however, imposes no memory restrictions and instead focuses on accelerating the learning compared to models trained from scratch, a challenge that is in some cases harder than outperforming standard replay on a continuous model, see Section 3.1.

Faster optimization. At its core, machine learning involves solving a challenging and computationally expensive optimization problem, and many approaches have been proposed to streamline this process (Sun et al., 2019). First-order methods, such as stochastic gradient descent (SGD)(Robbins & Monro, 1951), are the most widely used, where the full gradient is typically approximated using small batches. However, SGD can suffer from slow convergence, especially in high-variance settings(Johnson & Zhang, 2013), which can be mitigated by techniques like Nesterov momentum (Sutskever et al., 2013). Adaptive approaches such as AdaGrad (Duchi et al., 2011) and Adam (Kingma, 2014) help by adjusting the learning rate dynamically, though explicit learning rate scheduling often enhances their performance (Loshchilov & Hutter, 2022; Smith & Topin, 2019). Second-order methods, despite their promise, are hampered by the computational expense of estimating the Hessian (Martens, 2016). Goyal et al. (2017) also highlights the intricate relationship between batch size and learning rate in neural network optimization. Additionally, regularization techniques like batch normalization (Ioffe & Szegedy, 2015) and weight normalization (Salimans & Kingma, 2016) can further improve convergence. These considerations become especially important when optimizing from a pre-trained model, as is the case in this work, rather than from random initialization (Narkhede et al., 2022).

Warm start. Starting from non-random initialization is most commonly used in transfer learning, where a pre-trained model (typically on ImageNet) serves as a starting point to kick-start training downstream tasks (Zhuang et al., 2020). Contrary to our work, these works are often not concerned with performance on the pre-training (i.e. old) data. While beneficial, pre-training may hurt downstream performance (Zoph et al., 2020). This may be explained by a loss of plasticity in trained networks (Dohare et al., 2024; Abbas et al., 2023). Ash & Adams (2020) showed that when continuously training on the same data source, lower performance is reached than when starting from scratch. Gupta et al. (2023); Parmar et al. (2024) study how to continually pre-train NLP models which is strongly related to our setting. They examine learning rate scheduling but do not consider loss of plasticity, regularization, and data selection as done in this paper.

2 Problem Description

We consider the following setup: an existing dataset, denoted as $\mathcal{D}_{old} = (\mathcal{X}_{old}, \mathcal{Y}_{old})$, where $\mathcal{X}_{old}$ represents the input examples and $\mathcal{Y}_{old}$ the corresponding labels. A model $f_{old} : \mathcal{X}_{old} \to \mathcal{Y}_{old}$, has already been trained on this dataset. We are then provided with a new data, $\mathcal{D}_{new} = (\mathcal{X}_{new}, \mathcal{Y}_{new})$, which may include new classes. The objective is to get a model $f : (\mathcal{X}_{old} \cup \mathcal{X}_{new}) \to (\mathcal{Y}_{old} \cup \mathcal{Y}_{new})$ that performs well on both the new and the old datasets, $\mathcal{D} = \mathcal{D}_{old} \cup \mathcal{D}_{new}$, with minimal computational cost.

To ensure a fair comparison between models, we use the same architecture when comparing the computation costs of different training methods. This, and keeping a fixed batch size, allows us to quantify computational cost through the number of training iterations. Let f^i represent the model f after i training iterations. We define the speed s of a model f in achieving a target accuracy, a, as

the first iteration in which f reached or surpassed that accuracy. Formally:

$$s(f, a) = \min_i \left\{ i \in \mathbb{N} : \frac{1}{|\mathcal{D}|} \sum_j^{|\mathcal{D}|} \mathbb{1}[f^i(x_j) = y_j] \geq a \right\} \tag{1}$$

We use the relative speed-up L_r when comparing the performance of different models against a baseline model. The baseline model $f_{scratch}$ is trained from scratch on the combined dataset $\mathcal{D}$ and achieves an accuracy of $a_{scratch}$. Then we can define L_r as:

$$L_r(f, f_{scratch}) = \frac{s(f_{scratch}, a_{scratch})}{s(f, r/100 \cdot a_{scratch})} \tag{2}$$

or the relative number of iterations that a model f requires to reach the same (L_{100}) or a fraction of (e.g. L_{99}) the final accuracy of a model trained from scratch. For instance, $L_{99} = 2$ would indicate that model f attains an accuracy of $0.99 \cdot a_{scratch}$ with a 2 times lower computational cost than was required to train the full model from scratch to attain $a_{scratch}$, or equivalently, half the number of iterations. To reduce notation complexity, we will simply use L_r in the remainder of the paper when the models can be inferred from context. Note that this measure only works when each iteration has the same computational cost, but the idea can easily be extended to when this is not the case.

2.1 Implementation details

Datasets. We conducted experiments on a variety of image classification datasets, including CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100, subsets of ImageNet (ImageNet-100 and ImageNet-200) (Deng et al., 2009), and Adaptiope (Ringwald & Stiefelhagen, 2021). For continuous training, each dataset was divided into disjoint subsets, and training proceeded in a cumulative manner. For example, in CIFAR-100 (80+10+10), classes are split into three groups: 80 classes, followed by 10, and 10. The model was trained sequentially, first on the 80 classes, then on the combined 80 + 10, and finally on all 100 classes. Class splits were randomized, where the specific seeds are available in the attached code.

Training and baselines. Unless otherwise specified, we used ResNet-18 with a cosine annealing scheduler (Loshchilov & Hutter, 2016), the Adam optimizer, and a learning rate of 0.001. All models are trained with standard cropping and horizontal flipping augmentations. Additional model and hyperparameter details can be found in the attached code and Appendix A.2. The *scratch* baseline indicates a model that is trained from a random initialization on both old and new data together. The *naive* baseline represents a continuous model that simply continues training from the old model, without any modification.

Experimental details. Every experiment shown in this paper is repeated five times. The results shown are the averages of these experiments, accompanied by their standard error in plots. The results presented are always obtained by using the combined test sets of the old and the new datasets.

3 Method

In deep learning, optimization is typically performed using variants of stochastic gradient descent, where model parameters θ_i are updated by subtracting an estimate of the gradient of the objective function, based on a small batch of samples. The standard minibatch SGD update rule is:

$$\theta_{i+1} = \theta_i - \frac{\eta}{N} \sum_{j \in B_i} \nabla L(\theta_i, (x_j, y_j)) \tag{3}$$

where η is the learning rate, L is the objective function, and B_i a batch of N examples sampled from the full dataset $\mathcal{D}$. All components of this update – model initialization (θ_0), batch composition, learning rate adjustments, and modifications to the objective function – play a critical role in the optimization trajectory and convergence speed. For other optimizers like e.g. Adam (Kingma,

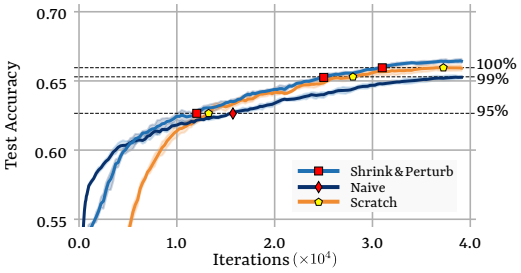

Figure 2: **Initialization**. Naive continuous training is slower and less accurate than retraining from scratch. Re-introducing plasticity with shrink-and-perturb improves both speed and accuracy, surpassing scratch training.

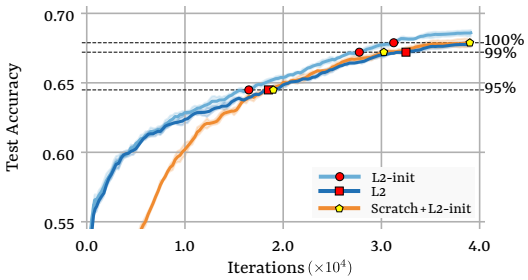

Figure 3: **Objective function**. Regularizing the objective function with $L2$-losses is beneficial in both from scratch and continuous learning, yet the latter outperforms the former when using $L2$-init regularization.

2014), the simple average here would be replaced by a momentum-based average, but the idea remains the same. In the following sections, we explore how each of these elements can significantly accelerate continuous model training. CIFAR-100 (70+30) serves as a case study for comparing various strategies.

3.1 Initialization

When training models from scratch, careful random initialization offers several advantageous properties (Narkhede et al., 2022). However, during continuous training, many of these advantages are lost as training does not start from a random set of weights. Several works have shown that this potentially leads to reduced plasticity, i.e. not being able to learn new information as fast and accurately as a model trained from scratch (Ash & Adams, 2020; Dohare et al., 2023). This issue is similarly observed in continuous training, as shown in Figure 2. The naive benchmark is both worse and converges slower than retraining from scratch.

To restore plasticity during warm-start training without distribution shifts, Ash & Adams (2020) introduced the 'shrink and perturb' method. Rather than using the previously trained model's weights directly, the weights are shrunk by a factor α and combined with a small portion of randomly initialized parameters θ_{random}. The resulting initialization θ_{init} is computed as:

$$\theta_{init} = \alpha\theta_{old} + \beta\theta_{random} \tag{4}$$

In our experiments, we used $\alpha = 0.4$ and $\beta = 0.001$ without tuning, as proposed by the original authors. However, a broad range of α and β values yielded qualitatively similar results (see Appendix A.3). In Figure 2, we compare ResNet-18 models trained continuously on CIFAR-100 (70+30) with and without the shrink-and-perturb method. The results show that re-introducing plasticity can not only accelerate convergence but also potentially improve final accuracy compared to both scratch training and continuous training without it.

3.2 Objective function and regularization

An alternative approach to tackle the reduced plasticity problem is to prevent it from arising in the first place by changing the objective function L. Dohare et al. (2023) proposed maintaining plasticity through continual backpropagation, while Kumar et al. (2023) introduced a simplified version that uses an $L2$ regularizer towards the initial random weights θ_0, rather than towards the origin as is typically done in $L2$ regularization. In our setting, this involves modifying the objective function L to include this $L2$-init regularizer:

$$L_{reg}(\theta_i, (x_j, y_j)) = L(\theta_i, (x_j, y_j)) + \lambda\|\theta_i - \theta_0\|^2 \tag{5}$$

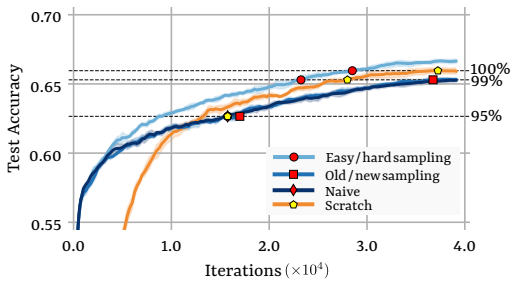

Figure 4: **Batch composition**. 'Old / new' sampling balances old and new examples in each batch, unlike the naive baseline, which uses proportional sampling. 'Easy / hard' sampling reduces the inclusion of the easiest and hardest old examples, significantly improving performance. (Naive and 'old/new' results nearly overlap.)

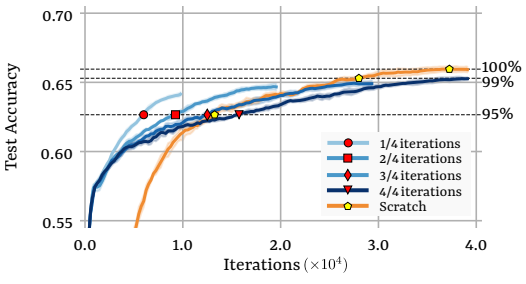

Figure 5: **Hyperparameters**. Shortening the learning rate scheduler allows for faster convergence but at the cost of lower final accuracy. On its own, changing the scheduler does not reach the required accuracy, yet when combined with the other aspects, it becomes important (see Section 4.1)

In our experiments, we used $\lambda = 0.01$ without tuning, as proposed by the original authors. However, a broad range of λ values yielded qualitatively similar results (see Appendix A.4). In Figure 3, we train ResNet-18 models on CIFAR-100 (70+30) and compared the results of models trained from scratch with those trained continuously, both with the $L2$-init regularizer. The results indicate that adding this regularization term accelerates the convergence of the continuously trained models, more so than it improves models trained from scratch. Since regularization can also benefit models trained from scratch, we use this baseline here, as well as the more standard $L2$ regularization (i.e. $\theta_0 = \mathbf{0}$) which is not as effective as $L2$-init regularization.

3.3 Batch composition

Estimating the full gradient of a dataset is expensive in the deep learning case, as many iterations are required to reach a good solution. To overcome this, minibatches containing a small part of the entire dataset are used to estimate the gradient. Typically, examples are sampled from the full dataset with uniform probability. In continual learning, it is a common practice to balance examples from old and new data in a batch (Rolnick et al., 2019), which either increases or decreases the sampling probability of old data depending on whether there is either more or less old than new data.

In Figure 4, we show that having an equal number of new and old examples ('old / new sampling') does not improve the results compared to the naive approach, which balances old and new examples in a batch according to their ratio in the full dataset (i.e. 70% old examples and 30% new ones in this example). Katharopoulos & Fleuret (2018) show that the optimal sampling distribution is proportional to the gradient norms of the individual examples. In Appendix A.7, we show that at the very start of training the gradient norms of the old examples are indeed smaller, but after about 100 iterations, there is no noticeable difference on the class level, which explains the results.

While the ratio of old and new examples does not have an immediate effect, not all data in the replay memory is equally useful. Often in continual learning, access to the old dataset $\mathcal{D}_{old}$ is limited to a small fraction. In our case, the replay memory has infinite capacity. We build on the work of Hacohen & Tuytelaars (2024), who propose a sampling strategy that reduces the importance of very easy and very difficult examples in the memory. More formally, they define learning speed ls of a sample x_j as the epoch e in which sample is classified correctly:

$$ls(x_j, y_j) = \frac{1}{E} \sum_{i=1}^{E} \mathbb{1}[f^i(x) = y] \tag{6}$$

with E the total number of epochs and f^i the model at epoch i.

The learning speed is used to order the old examples from easy to hard, given that the necessary information is recorded during training of the old model (For more details, see Appendix A.5).

Using this order, we reduce the sampling probability of the 10% easiest and highest examples to one-tenth of the other examples. In the easy examples, there is no information left, and the hardest ones are never learned, and thus as useful. The results in Figure 4 show that this approach ('easy / hard sampling') is helpful and speeds up training in the continuous case. In Appendix A.5 we show that this is robust to the exact hyperparameters and that, while sacrificing some convergence speed, the easiest and hardest examples can be removed entirely.

3.4 Learning rate scheduling

The learning rate controls the step size in stochastic gradient descent, directly influencing convergence speed. In deep learning, the learning rate typically starts high to allow faster progress toward optimal solutions and decreases gradually to avoid overshooting (Zeiler, 2012). Common scheduling strategies include 'multistep' scheduling (Zagoruyko & Komodakis, 2016), which reduces the learning rate at set intervals, and cosine annealing (Loshchilov & Hutter, 2017), which decays it more smoothly over time.

When training a model continuously, since the model is already trained on parts of the data, a more aggressive learning rate scheduling can accelerate convergence. While the learning rate still needs to decrease over time, this can happen more rapidly over fewer iterations, as the model has already learned key information. As shown in Figure 15 in the Appendix, the loss curve for the continuous model reaches a plateau much faster compared to scratch training, indicating that the model approaches a local optimum more quickly. This supports the need for a faster reduction in the learning rate (Zeiler, 2012).

We train ResNet-18 models on CIFAR-100 (70+30 split), using different lengths of cosine learning rate schedulers. The most aggressive scheduler used only 25% of the total iterations, while others used 50% and 75%. The learning curves, depicting the mean test errors from these experiments, are shown in Figure 5. The results indicate that more aggressive schedulers lead to faster convergence of the continuous model to a performance level comparable to the scratch solution. However, overly aggressive scheduling hurts the final performance, as the networks fail to achieve 100% of the scratch model's final accuracy. On its own, reducing the learning length is not helpful, yet when combined with the other aspects, it becomes important (See Section 4.1). Similar qualitative results were observed with multistep scheduling, see Appendix A.6. For the remainder of this paper, any adjustments to the learning rate scheduler for specific methods are explicitly mentioned, and a comparison to the unmodified variant is provided.

4 Results

In the previous section, we explored several key aspects of SGD optimization, and each can serve as the foundation for methods that reduce the computational cost of continuous training. By leveraging insights from various works in the literature, we proposed a method for each aspect, demonstrating the effectiveness of each method individually.

In this section, we begin by showing that these methods are complementary, with their combination further accelerating convergence beyond what is achieved by applying them separately. We show that although the same techniques can lead to better convergence when training from scratch, the benefit is larger in the continuous setting. We then extend this combined approach to various datasets, scenarios, and data splits, demonstrating its applicability across image classification tasks.

4.1 Ablations

In the previous section, we introduced several methods aimed at speeding up the convergence of continuous models, each targeting a different aspect of the optimization process. We now examine whether these improvements are complementary or if the benefits of one method negate the effectiveness of others. Table 1 presents results from training five ResNet-18 models and compares all combinations. While not fully cumulative, each combination offers benefits, either in convergence speed or in final accuracy. Additionally, we applied the same techniques to a model trained from scratch, to rule out the possibility that the techniques are *only* improving overall learning capabilities.

Table 1: Ablation results of the four different aspects studied in Section 3 under '*continuous*' on CIFAR100 (70+30). All speed-ups are relative to the model trained from scratch without any modification, '/' indicates the required accuracy was not reached. Each of the aspects individually establishes a speed-up, and combining them improves the results further, albeit not fully cumulative. On the right hand side of the table, '*scratch*' shows the result of a model trained from scratch using the same techniques. Some of them improve the training speed, but to a lesser extent than in the continuous case.

				Continuous			Scratch		
Initialization	**Regularization**	**Data**	**Scheduler**	**Max Acc**	L_{99}	L_{100}	**Max Acc**	L_{99}	L_{100}
				65.26	/	/	65.92	×1.33	×1.00
✓				66.43	×1.49	×1.20	65.06	/	/
	✓			68.60	×1.94	×1.66	67.90	×1.69	×1.51
		✓		66.65	×1.60	×1.31	66.61	×1.54	×1.31
			×0.25	64.15	/	/	63.72	/	/
✓	✓			68.91	×2.19	×1.82	68.01	×1.75	×1.54
✓		✓		66.76	×2.92	×2.53	66.61	×1.54	×1.31
	✓	✓		68.86	×1.96	×1.69	68.30	×1.67	×1.49
✓	✓	✓		**69.47**	×2.19	×1.84	68.31	×1.69	×1.52
✓	✓	✓	×0.25	68.26	×**5.73**	×**5.32**	63.74	/	/

The results show that while some optimization aspects have an effect on the scratch models as well (most notably regularization), their influence is always stronger in the continuous case. This is especially clear when shortening the learning rate scheduler, which allows the continuous model to learn faster, yet greatly reduces the final accuracy of the from-scratch model. These results indicate that the proposed techniques are effective in leveraging the benefits of having a model trained on the old data available. Future research targeting different aspects of the continuous learning process could yield further gains in training speed, as multiple optimization strategies can be effectively integrated to speed up the learning.

4.2 Multiple tasks

We used CIFAR100 (70+30) as a case study in Section 3, adding 30 new classes. In many continual learning scenarios, new data is not only added once, but repeatedly. In Figure 6 we show that applying the same aspects on consecutive tasks continues to work as in the one-task case. For each task, we also train a model from scratch on all classes available up to this point. In Figure 6, the yellow dots indicate when the continuous model reaches the same performance (L_{100}) as the from-scratch model for that task. For every task $L_{100} > 1$ and gets progressively larger, indicating that the continuous model benefits more than it has trained for longer in the past.

These results highlight an important insight: in total, the continuous models have trained for more iterations than the models that are trained from scratch, which also explains why their final accuracy may surpass that of training from scratch. However, when accumulating past costs, the total cost of the continuous model is significantly lower than the sum of costs for models trained from scratch for every task.

4.3 Domain adaptation

All previous experiments had new classes in the 'new' data, often referred to as class-incremental learning in continual learning (De Lange et al., 2021). Here, we use the Adaptiope dataset (Ringwald & Stiefelhagen, 2021) to show that our method also works when new data contains no new classes, but the same classes from a different domain. In particular, the 'old' data consists of 123 categories of product images from a shopping website and the 'new' data are real-life images of the same products captured by users. The objective is to achieve strong performance across both domains, ideally with faster convergence than re-training the model from scratch on both datasets.

Figure 7 shows how our method outperforms both training from scratch and the naive continuous approach when including a new domain. The relative speed-up compared to the naive baselines is smaller than in the class-incremental case, but the final accuracy is considerably improved.

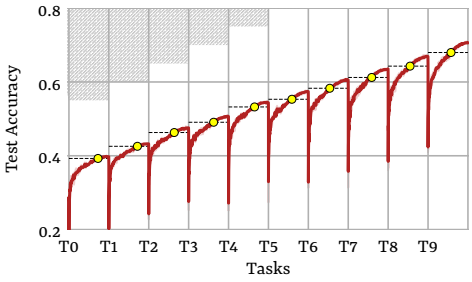

Figure 6: Test accuracy on the full CIFAR-100 when training a continuous model with our method on CIFAR100 $(50 + \sum_{i=1}^{10} 5)$. Yellow dots mark the L_{100} iteration, where the continuous model outperforms the from-scratch model. The shaded area indicates maximum possible performance based on data availability (e.g., 55% during the first task $50 + 5$).

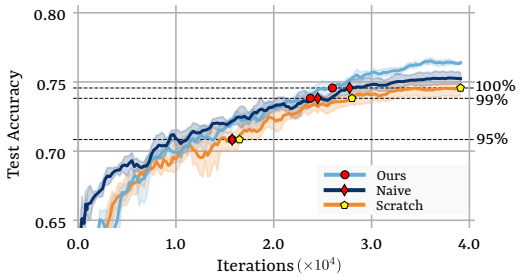

Figure 7: Domain adaptation results using the Adaptiope dataset. The 'old' data contains *product images*, the 'new' data has *real life* images. The test accuracy is on both domains. Although naively continuing to train is nearly as fast as our method in this setting, our final test accuracy is considerably higher.

Table 2: Results of our method across various datasets and splits. We report final test accuracy and relative speedup to reach similar performance compared to the scratch solution. The multiplier after the algorithm name indicates learning rate scheduling aggressiveness (e.g., Ours $(\times 0.5)$ means the minimum learning rate is achieved at half the iterations of Ours $(\times 1)$). '/' indicates that the accuracy level was not reached. Table (a) presents results on different datasets, while Table (b) provides an in-depth analysis of CIFAR-100 with various class splits.

(a) More image classification benchmarks

Dataset	Algorithm	Max Acc	L_{99}	L_{100}
CIFAR10 (8+2)	Scratch	83.32	×1.55	×1
	Continuous	81.37	/	/
	Ours (×1)	83.95	×1.83	×1.29
	Ours (×0.5)	**84.33**	×3.61	×2.92
Adaptiope-PI (100+23)	Scratch	74.59	×1.14	×1
	Continuous	74.93	×1.17	1.02
	Ours (×1)	**77.51**	×1.55	×1.42
	Ours (×0.5)	76.75	×2.36	×2.24
ImageNet-100 (80+20)	Scratch	62.34	×1.18	×1
	Continuous	62.21	×1.23	/
	Ours (×1)	**67.41**	×2.16	×1.97
	Ours (×0.5)	66.26	×3.39	×3.09
ImageNet-200 (180+20)	Scratch	55.16	×1.18	×1
	Continuous	53.5	/	/
	Ours (×1)	**59.62**	×1.69	×1.64
	Ours (×0.5)	58.54	×2.84	×2.75

(b) Different ratios of old and new classes.

Dataset	Algorithm	Max Acc	L_{99}	L_{100}
CIFAR100	Scratch + L2-init	67.90	×1.29	×1.00
	Scratch (×0.5)	67.14	/	/
90+10	Ours (×1.0)	69.42	×1.56	×1.41
	Ours (×0.5)	69.07	×2.95	×2.65
	Ours (×0.25)	68.42	×5.05	×4.47
70+30	Ours (×1.0)	69.47	×1.61	×1.45
	Ours (×0.5)	68.77	×2.84	×2.56
	Ours (×0.25)	68.26	×4.74	×4.34
50+50	Ours (×1.0)	69.09	×1.50	×1.35
	Ours (×0.5)	68.42	×2.70	×2.37
	Ours (×0.25)	67.57	×4.34	/
30+70	Ours (×1.0)	68.81	×1.49	×1.31
	Ours (×0.5)	68.10	×2.56	×2.30
	Ours (×0.25)	66.93	/	/

4.4 Other Datasets and Scenarios

In Section 3, we demonstrated the effectiveness of each method using the CIFAR-100 (70+30) dataset for consistency across experiments. Table 2a extends these results to a range of datasets, including CIFAR-10, ImageNet-100, ImageNet-200, and the product images of Adaptiope. For each dataset, we compare the performance of models trained from scratch, a naive continuous model, and a continuous model using our full method. We report the relative speed-up to reach 99% and 100% of the scratch-trained model's accuracy, along with the final accuracy. Across all the different datasets we tested, our method consistently enhances the speed of convergence both for the L_{99} and L_{100} cases, often also surpassing the final accuracy of the scratch model.

Table 2b presents results for different class split ratios, including CIFAR-100 splits of (90+10), (70+30), (50+50), and (30+70). Our methods consistently accelerate training, with speed-up being more significant when the second data split is smaller. This aligns with intuition: when the ini-

tial 'old' data is larger, the old model already has more knowledge, requiring fewer updates when exposed to new data, compared to training from scratch, which treats both splits equally.

5 Conclusion

Throughout this paper, we have shown that the computational cost that is paid to train models on old data should not be treated as something that is lost, but rather as a starting point for further training whenever new data becomes available. We studied the four main components of the traditional SGD update rule – initialization, objective function, data selection and hyperparameters – and showed that each of them individually can contribute to faster convergence on old and new data combined when starting from an old model. These aspects are thoroughly tested to be robust and effective across a wide range of scenarios. Our proposals should be seen as starting points to work further towards solutions that are even more effective in re-using old models, and saving computational costs across the board in machine learning development and applications.

References

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 620–636. PMLR, 2023.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

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

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

Google. Pricing — Cloud Storage — Google Cloud — cloud.google.com. https://cloud.google.com/storage/pricing, 2024a. [Accessed 01-10-2024].

Google. GPU pricing — Compute Engine: Virtual Machines (VMs) —no Google Cloud — cloud.google.com. https://cloud.google.com/compute/gpus-pricing, 2024b. [Accessed 01-10-2024].

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.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to re-warm your model? In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL https://openreview.net/forum?id=pg7PUJe0Tl.

Guy Hacohen and Tinne Tuytelaars. Forgetting order of continual learning: Examples that are learned first are forgotten last. *arXiv preprint arXiv:2406.09935*, 2024.

Md Yousuf Harun, Jhair Gallardo, Junyu Chen, and Christopher Kanan. Grasp: a rehearsal policy for efficient online continual learning. *arXiv preprint arXiv:2308.13646*, 2023.

Chip Huyen. Real-time machine learning: challenges and solutions, Jan 2022. URL https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html#towards-continual-learning. Online; accessed 29-August-2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Daniel Kahneman and Amos Tversky. Subjective probability: A judgment of representativeness. *Cognitive psychology*, 3(3):430–454, 1972.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *Proceedings of The 3th Conference on Lifelong Learning Agents*, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2022.

James Martens. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

John C McCallum. Price and performance changes of computer technology with time. https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage, 2023.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322, 2022.

Nvidia. NVIDIA A100 GPUs Power the Modern Data Center — nvidia.com. https://www.nvidia.com/en-us/data-center/a100/, 2024. [Accessed 01-10-2024].

Jupinder Parmar, Sanjev Satheesh, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv preprint arXiv:2407.07263*, 2024.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3698–3707, 2023.

Tobias Ringwald and Rainer Stiefelhagen. Adaptiope: A modern benchmark for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 101–110, 2021.

Maxime Rio. Tech Insights: A behind-the-scenes look at rolling out new GPU resources for NZ researchers — nesi.org.nz. https://www.nesi.org.nz/case-studies/tech-insights-behind-scenes-look-rolling-out-new-gpu-resources-nz-researchers, 2023. [Accessed 01-10-2024].

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

James Seale Smith, Lazar Valkov, Shaunak Halbe, Vyshnavi Gutta, Rogerio Feris, Zsolt Kira, and Leonid Karlinsky. Adaptive memory replay for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3605–3615, 2024.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019.

Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, et al. Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*, 2024.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3014–3023, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76, 2020.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33: 3833–3845, 2020.

A Appendix

A.1 Cost estimation

To train a machine learning model there is both a memory (storage) and compute cost. Here we will work out an example for training a Vision Transformer (ViT) (Vaswani, 2017) on ImageNet 21k. This dataset is about 1.13 terabytes. Cloud storage for common providers averages around 0.023$ per GB each month, which would be around 30.13$ per month (Google, 2024a). However, local storage is much cheaper than cloud storage. Today hard disk storage is about 11$ per TB (McCallum, 2023), which can last for 10 years or more.

ViT models were trained on 8 Nividia P100 gpus for 3.5 days (Vaswani, 2017), which cost $1.46 per hour on Google Cloud, which would be $981 in total for the entire training (Google, 2024b). Buying the GPUs would be much more expensive, with prices for a single A100 GPU above $10.000 (Nvidia, 2024). Modern A100 GPUs are about 3.5 times faster on real-life work loads (Rio, 2023), but more expensive to rent. At 4$4.05 per hour, the total price would still be $777.5.

A.2 Hyperparameters

Unless specified otherwise, all experiments are trained with the Adam optimizer (Kingma, 2014), with default settings of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and no weight decay. The starting learning rate is equal to 0.001 and the batch size is consistently 128 in all experiments. Unless specified differently, a cosine scheduler is used where the minimal learning rate $1e-6$ is reached after 39.100 iterations, which is equal to 50 epochs using the complete CIFAR100 dataset. All experiments use cropping and random horizontal flip augmentations.

A.3 Initialization

Figure 8 shows different values of α in the shrink and perturb update rule. In genreal, shrinking more gives better results, although it slows down learning results at the start of learning. Shrinking enough is necessary and not doing so may lead to suboptimal accuracies because of loss of plasticity. In Figure 9 different values of β are tested, which add various levels of noise to the initialization. There is no visible effect of the size of this parameter, which is roughly in line with (Ash & Adams, 2020).

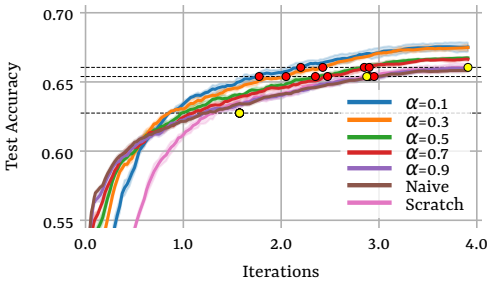

Figure 8: Grid test of the α (shrink) hyperparameter in the shrink and perturb update rule.

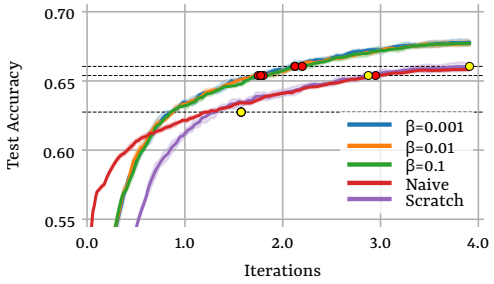

Figure 9: Grid test of the β (perturb) hyperparameter in the shrink and perturb update rule

A.4 Regularization

Figure 10 shows the influence of the λ parameter in the $L2$-init regularization, which controls the strength of the regularization. A too low value will not have any effect, while setting this value too high, will lead to slower than necessary learning speeds.

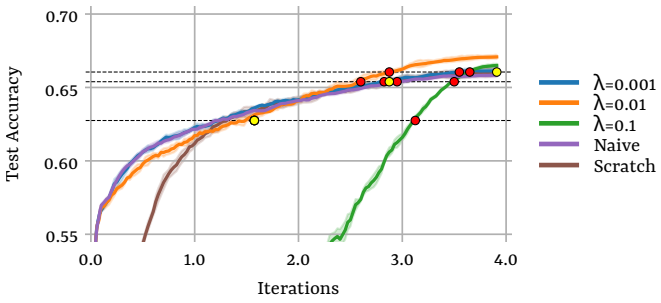

Figure 10: Ablation of the λ parameter in the $L2$-init regularization.

A.5 Batch composition

Figure 11 shows how fast examples are learned during the first task of CIFAR100 (70+30). The x-axis shows the epochs, while each row indicates a single sample. The examples on the bottom are green almost from the start of training, these are the ones that are very easy: the model almost has no difficulty learning them, which also means they do not carry a lot of information. The ones on the top are almost completely red: they are never learned. These examples are equally not very useful: they are so hard the model can not learn them anyway (which may be a result of bad labeling, bad images etc.).

In Figure 12, we ablate two of the parameters of the 'easy / hard' sampling process. c indicates what proportion of the easy and hard examples is influenced. e.g. $c = 0.2$ means that 20% procent of the examples is affected, or the 10% easiest and the 10% hardest. r indicates the probability that the easy and hard examples are sampled in a batch, relative to the examples in the middle. e.g. with $r = 0.1$, an easy or hard example is 10 times less likely to be in a minibatch. The result with $r = 0$ shows that we can even get rid of these examples, reducing the memory requirements for this method. When $r = 0$, this becomes the method proposed by (Hacohen & Tuytelaars, 2024), on which this idea is based.

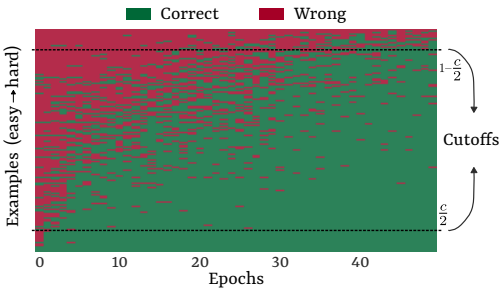

Figure 11: How fast examples are learned during the initial learning phase of training on the old data of CIFAR100 70+30 (i.e. the first 70 classes). Some examples are almost learned instantly (bottom), others never (top). The cutoffs indicate both the 10% easiest and hardest examples.

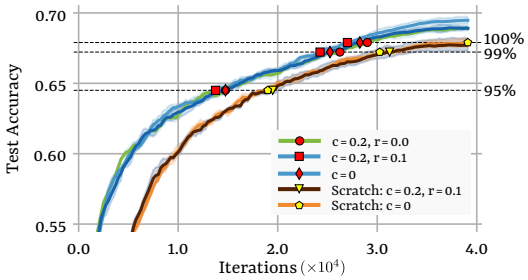

Figure 12: Test accuracy of CIFAR100 on the $70 + 30$ benchmark. Removing the easiest and hardest examples ($r = 0$) barely has an influence. Sampling them with a low probability ($r = 0.1$) allows to learn a little faster on the continuous model, without such an effect on the from scratch model.

A.6 Schedulers

Figure 13 shows an experiment on CIFAR100 (70+30), with a multistep learning rate scheduler (which reduces the learning rate by a fixed factor at fixed steps) rather than a cosine learning rate scheduler. This scheduler reaches the same conclusion, although in this case the iterations where the scheduler kicks in has a much larger influence than in the cosine scheduler case. In the continuous case, the learning rate is kept too high for too long, preventing the model to learn the last details, while the from scratch model is still learning.

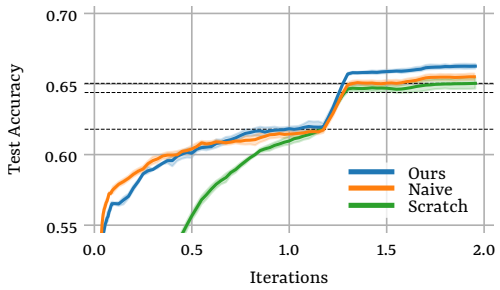

Figure 13: When using multi-step schedulers the qualitative result stays the same, although cosine schedulers are better suited to this task of continuous learning.

A.7 Gradient Norms

Figure 14 shows the average gradient norm of examples of old and new data in the CIFAR100 (70+30) baseline during training of the naive baseline. While at the very start the gradient of new data is considerably higher, this difference has completely disappeared after more than 100 iterations, which is nearly immediately considering 40.000 iterations in total. This explains mostly why it is not useful to oversample new data, which would indicate that new data is more important to learn than remembering the old data, which is not the goal.

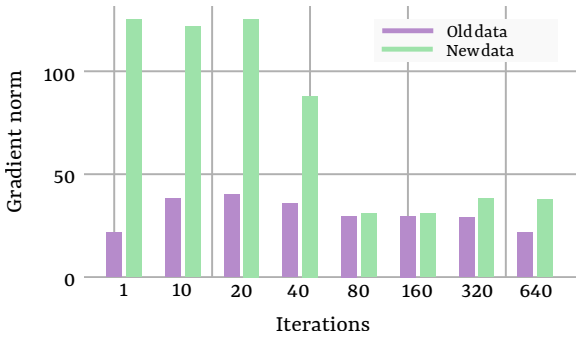

Figure 14: Gradient norms of 'old' and 'new' data during training of the naive baseline on CIFAR100 (70+30).

A.8 Loss plateau

Figure 15 shows how the loss of the naive solution plateaus earlier than when training from scratch, thus allowing to schedule learning rates earlier. However, it is not because such a loss plateau is reached that the final result will be better, it merely indicates that the results won't get any better when training longer than from the point the plateau is reached. In fact, as can be seen in Figure 15, the loss starts to increase again as overfitting takes place. In this sense, it might even be necessary to schedule early enough to obtain the best possible results.

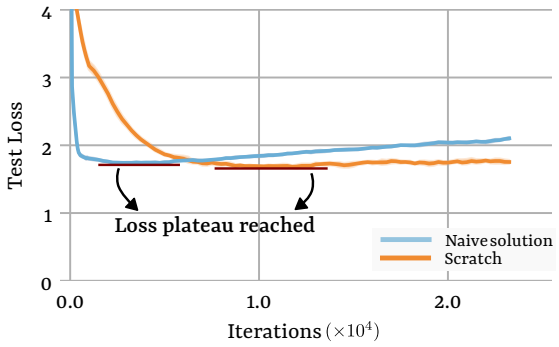

Figure 15: The continuous naive baseline loss plateaus earlier than that of the from scratch baseline.