Challenging Common Assumptions about Catastrophic Forgetting and Knowledge Accumulation

Timothee Lesort[12], Oleksiy Ostapenko[123], Pau Rodríguez[3], Diganta Misra[12], Md Rifat Arefin[12], Laurent Charlin[145], Irina Rish[125]

[1]Mila - Quebec AI Institute, [2]Université de Montréal, [3]ServiceNow, [4]HEC Montréal, [5]Canada CIFAR AI Chair

Abstract

Building learning agents that can progressively learn and accumulate knowledge is the core goal of the continual learning (CL) research field. Unfortunately, training a model on new data usually compromises the performance on past data. In the CL literature, this effect is referred to as catastrophic forgetting (CF). CF has been largely studied, and a plethora of methods have been proposed to address it on short sequences of non-overlapping tasks. In such setups, CF usually leads to a quick and significant drop in performance in past tasks. Nevertheless, despite CF, recent work showed that SGD training on linear models accumulates knowledge in a CL regression setup. This phenomenon becomes especially visible when tasks reoccur. We might then wonder if DNNs trained with SGD or any standard gradient-based optimization accumulate knowledge in such a way. Such phenomena would have interesting consequences for applying DNNs to real continual scenarios. Indeed, standard gradient-based optimization methods are significantly less computationally expensive than existing CL algorithms. In this paper, we study the progressive knowledge accumulation (KA) in DNNs trained with gradient-based algorithms in long sequences of tasks with data re-occurrence. We propose a new framework, *SCoLe* (Scaling Continual Learning), to investigate KA and discover that catastrophic forgetting has a limited effect on DNNs trained with SGD. When trained on long sequences with data sparsely re-occurring, the overall accuracy improves, which might be counter-intuitive given our understanding of catastrophic forgetting in CL. We empirically investigate KA in DNNs under various data occurrence frequencies. We show that the catastrophic forgetting usually observed in short scenarios does not prevent knowledge accumulation in longer ones. Moreover, propose simple and scalable strategies to increase knowledge accumulation in DNNs.

1 Introduction

Continual learning (CL) aims to design algorithms that learn from non-stationary sequences of tasks and accumulate knowledge. Classically, the main challenge of CL is catastrophic forgetting (CF) — fast performance degradation on previous tasks when learning from new data. CF is evaluated based on the 0-shot accuracy of past tasks in scenarios with short sequences of disjoint tasks (Lesort et al., 2020; Lange et al., 2019; Belouadah et al., 2021; Hadsell et al., 2020). This evaluation protocol has led our community to the conclusion that new knowledge systematically replaces the previous one, hence that catastrophic forgetting erases past knowledge.

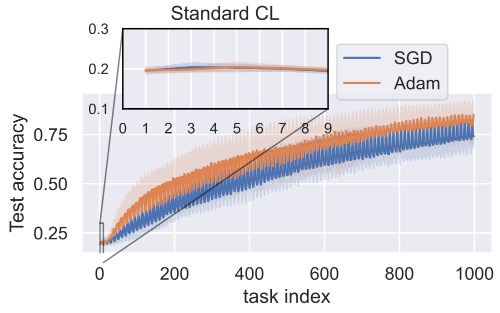


Figure 1: Knowledge accumulation is not observable in standard CL benchmarks (inset top). However, when repeating the sequence of tasks (bottom), it becomes apparent (MNIST, 2 classes per task, averaged over 3 lr and 5 seeds, each task until convergence)

We challenge this idea and step back from classical CL and CF literature to investigate to which extent fine-tuning with SGD under sequential distribution shifts prevents the model from accumulating knowledge.

Our research is driven by a simple observation highlighted in Fig. 1. It shows that a DNN trained with SGD and Adam leads to no overall improvement and first but accumulates knowledge after repeated exposure to a fixed sequence of tasks (even without a separate head per task). This phenomenon contradicts the dominant belief in the CL literature that CF should prevent such accumulation. To investigate this phenomenon further, we demonstrate in Section 2 that a DNN can retain knowledge about one task even after several learning unrelated tasks (which we validate using meta-testing accuracy).Motivated by these results, we investigate the effect of knowledge accumulation in longer sequences of tasks where data is not generated in a fixed pattern. We propose *SCoLe* (Scaling Continual Learning), a framework that allows the generation of long sequences of potentially overlapping tasks (Figure 4). *SCoLe* relaxes the constraint that tasks cannot overlap but constrains the sampling to a subset of classes that changes over time (locally stationary). Samples are non-IID, and learning a task can still lead to CF of previous knowledge. Thus, Scole is designed to reveal whether proper knowledge accumulation occurs in models or if forgetting prevents it. Our findings reveal that when scaled to hundreds or thousands of tasks, deep neural networks (DNNs) trained with stochastic gradient descent (SGD) display a consistent accumulation of knowledge, providing evidence that the impact of catastrophic forgetting is limited. We study the impact of class and task occurrence frequencies on model performance to understand better the knowledge accumulation capability of deep neural networks. We also explore different options to boost knowledge accumulation and benefit from it.

Our contributions are as follows:

- We first show that forgetting does not prevent the transfer to downstream tasks, even several tasks later. We call this effect "knowledge retention", and we show it leads to knowledge accumulation when tasks are reoccurring.
- We propose an experimentation framework "*SCoLe*" (**S**caling **Co**ntinual **Le**arning) to study and assess knowledge accumulation through a long sequence of (reoccurring) tasks on a variety of datasets (MNIST, Fashion-MNIST, KMNIST, CIFAR10, CIFAR100, Tiny-ImageNet200).
- Our study reveals that deep neural networks (DNNs) trained with SGD on a non-stationary distribution are capable of accumulating knowledge, challenging the usual understanding of the impact of catastrophic forgetting.
- We investigate different strategies to improve knowledge accumulation effects, such as gradient masking, frequency replay, optimizing the learning-forgetting trade-off or making models wider.

The code to reproduce results, setups and training is available at `https://github.com/TLESORT/SCoLe`.

2 The Long-Term Effect of Forward Transfer

In the continual learning literature, transfer is a major axis of investigation alongside CF (Lopez-Paz & Ranzato, 2017; Riemer et al., 2019). The transfer can either be forward when already learned tasks help to solve the following ones, or backward when learning new tasks helps with solving past tasks (Lin et al., 2022b). Forward transfer is usually evaluated through the comparison of performance on individual tasks trained IID to their performance obtained in CL streams (Lopez-Paz & Ranzato, 2017; Veniat et al., 2021; Ke et al., 2020; 2021). Recently, one of the major types of transfer investigated in the literature is the transfer from pre-trained models to a sequence of downstream tasks (Ke et al., 2021; Ostapenko et al., 2022; Hayes et al., 2020; Ke et al., 2023). The transfer is usually evaluated through 0-shot performance, but can also be evaluated by probing (Davari et al., 2022; Caccia et al., 2019; Fini et al., 2020). Even if the literature investigating large pre-trained models gives clear insight into forward transfer having effects on all the downstream tasks of a continual learning scenario, forward transfer is usually evaluated by looking at how learning one task influences the following one without investigating explicitly effects on a longer term.

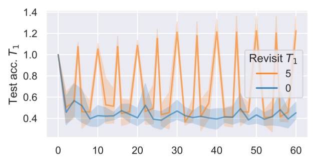


Figure 2: 0-shot test acc. on task 1 while learning distractor tasks and revisiting task 1 at different frequencies (i.e. every 5 tasks or never). This plot suggests that the model completely forgets between two occurrences of task 1.

In this section, we investigate if forward transfer can have a long-term impact beyond the direct next task. From a different perspective, we evaluate if the model can retain knowledge (knowledge retention – KR) over extended periods, and whether it can accumulate it (knowledge accumulation – KA), in non-stationary training regimes.

We design a scenario with one task of interest (task 1) that reoccurs at a regular fixed frequency with intermediate distractor tasks (we run for 5 different seeds, each resulting in task 1 composed of different randomly selected classes). The goal of this setting is to track if the distractor tasks lead to complete forgetting or not, and if not, how catastrophic the forgetting is. The scenario is designed from CIFAR100, the task of interest is a classification task with 5 randomly selected classes. Distractors tasks are built by sampling 5 random other classes and applying pixel-level permutation to the original images (akin to permuted MNIST (Goodfellow et al., 2013)).

We train a model with ResNet architecture, consequently since CNNs usually rely on features related to neighbouring pixels to learn, the pixel permutation of the distractor tasks should be especially perturbing and should lead to forgetting. The model is trained until convergence on each task.

We measure both the test and meta-test performance of the incrementally trained model. Test performance denotes the usual test accuracy directly measured on the trained model (i.e. 0-shot performance) while meta-test performance refers to the accuracy attained by the model if finetuned for a single epoch on task 1 (the task of interest). The meta-test measures how the model is potentially able to transfer knowledge from the past task of interest to a new instance of the same task.

Figure 2, shows a clear catastrophic forgetting behaviour in 0-shot performance. In this setting, distractor tasks make the model completely forget in terms of 0-shot performance, even if task 1 is revisited every 5 tasks. We will see now if the forgetting is that dazzling for meta-testing or if, in fact, the model retains knowledge from the previous task and can still transfer it. If the transfer had limited long-term impact, we would expect the meta-test performance to decrease very fast, while if the transfer had long term impact the meta-test performance should decrease slowly. We can note that long-term transfer is equivalent to knowledge retention. Then, if in addition to knowledge retention, the model can perform knowledge accumulation, the meta-test performance should increase progressively when the task is revisited.

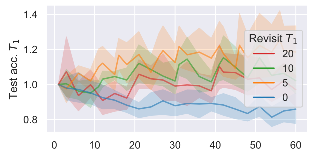


Figure 3: Normalized meta-test accuracy on task 1 while learning distracting tasks and revisiting task 1 at different frequencies. The meta-test accuracy shows that beyond 0-shot forgetting, the model performs knowledge retention and accumulation.

Figure 3 shows that meta-test accuracy actually slowly decreases over time when trained on distractor tasks, and has an upward tendency if task 1 is revisited regularly, with a faster increase for more frequent revisits. This shows that the model can indeed retain past knowledge for prolonged periods and accumulate it, even if the 0-shot test accuracy shows catastrophic forgetting and abrupt drops as in Figure 2. Note: In Figure 3 and Figure 2, we experiment with various sets of classes for task 1, which lead to variability in results depending on the difficulty of this task, to reduce the variance, we normalize accuracies by the task 1 accuracy at its first occurrence.

Summary. In this section, we have demonstrated, through meta-testing, the model's ability to retain knowledge over extended periods, despite experiencing 0-shot catastrophic forgetting. However, in practical settings, the primary concern is achieving satisfactory 0-shot performance for the overall problem, rather than focusing on meta-testing performance. Furthermore, our interest lies in scenarios with high task variability and scalability, where data does not repeat following a fixed pattern and context, and where the number of training stages (tasks) is high (>500 tasks). Motivated by this observation, we introduce the *SCoLe* framework in subsequent sections. This framework addresses the aforementioned requirements and enables us to explore whether knowledge accumulation in deep neural networks (DNNs) could lead to a progressive improvement of 0-shot accuracy in continual learning scenarios.

3 *SCoLe*: A Framework for CL with Long Task Sequences

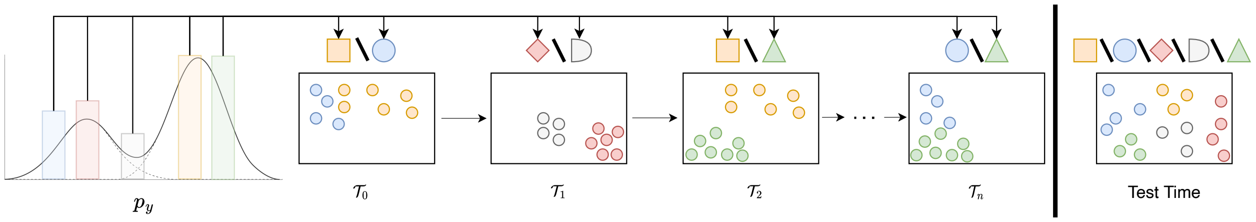


Figure 4: Illustration of *SCoLe* (Scaling Continual Learning) scenario. With 5 classes in total (one per colour) and 2 classes per task. The data is selected randomly based on their label to build tasks dynamically into a potentially infinite sequence. The model is trained on each task and evaluated on the test set containing all possible classes.

To study knowledge accumulation in long sequences of tasks, we propose the *SCoLe* (Scaling Continual Learning) framework. This framework enables the creation of scenarios with an arbitrarily long sequence of tasks. As in classical CL, each task's training set in *SCoLe* contains a subset of a dataset. The evaluation is conducted by measuring the overall performance of this dataset. In this setting, a learning system must accumulate knowledge by sequentially experiencing isolated parts of the data. The key difference from classical CL scenarios is that data in *SCoLe* can reoccur in different contexts.

Modern deep learning systems demonstrate the capability to accumulate knowledge when trained on stationary data. Nonetheless, in situations where data from a shifted distribution is introduced incrementally and does not re-occur, CF hinders the accumulation of knowledge. *SCoLe* proposes to fill the gap between these two settings and to study how knowledge accumulation evolves in non-stationary regimes as re-occurrence becomes more sparse in time. One of *SCoLe* goals is to determine the occurrence frequency regimes in which algorithms can effectively learn. In this paper, the algorithms of interest are how gradient-based algorithms, such as SGD. Intuitively, between two occurrences of a task, DNNs are trained on other tasks and may forget. We measure the test accuracy on the entire test set to track whether DNNs are learning more than forgetting. A model learning consistently more than it forgets will progressively converge to a solution for all tasks. Nevertheless, the trade-off between learning and forgetting depends on many factors, notably the occurrence frequency of the data. Experimenting with *SCoLe* makes it possible to vary the frequency occurrence to understand better the learning dynamics of a given approach.

As noted in Cossu et al. (2021) the reocurrence of data happens naturally in many settings, and continual learning algorithms can benefit from it. One example of a real scenario where data reoccurs is robotics. In a robotics environment, such as a building or a factory, the tasks and concepts may vary through time but regularly reoccur. It is then important to know under which task re-occurrence frequencies algorithms can be successful or not to be able to improve them. Other examples could also be to streaming setups where events can repeat, e.g. newspaper feed, social feed, or stock market... Intuitively, the training environment that fits *SCoLe* are those where the finite world assumption (Mundt et al., 2020) may hold and where training is realized over a long period of time.

Framework. We instantiate *SCoLe* in a classification setting (Figure 4). In *SCoLe*, a task is a period of time where the data distribution is stationary. The task changes with the input distribution. The test set stays fixed. In more detail, each task consists of samples from a random subset of the N total available classes (N is dataset dependent). The agent is a DNN that trains on the current task but evaluates on the test set D_{test} containing the data of all classes. The framework considers scenarios with varying numbers of tasks T and classes per task C.

Formally, the training set D_t for a task t consists of observations (x, y), which are generated like: $y \sim U(S_t)$, $x \sim p(X|Y = y)$. Here, the set $S_t = \{c_i\}_{i=0}^{C-1}$ is sampled without replacement from $P_{SCoLe}(Y)$ and contains classes of task t. $P_{SCoLe}(Y)$ is a probability distribution defined over N classes. We set $P_{SCoLe}(Y)$ at the beginning to define the scenario and set the occurrence frequency of each class.

In the default *SCoLe* scenario, $P_{SCoLe}(Y)$ is the uniform distribution over all N classes $U(0, N-1)$. We can also consider cases where $P_{SCoLe}(Y)$ is non-uniform (Section 5.2) or evolves over time (Appendix H).

SCoLe scenarios can also be built with additional constraints. We experimented with some ideas in the appendix, such as: penalizing reoccurrence of recent classes in Appendix B.1, restricting the number of possible tasks in Appendix B.5, imposing fix sequence of tasks as in Figure 1 but with a random change in the classes in Appendix B.4 or creating cyclic shifts in the distribution of classes in Appendix H.3.

Frequency of Occurrence. One of the key features of *SCoLe* is the ability to study KA as a function of the frequency of occurrence (ν) and determine the frequency range at which algorithms are efficient. Algorithms designed for IID settings are known to be effective learners when ν is very high ($\sim$ every batch). on the contrary, CL algorithms are designed for settings where ν is very low ($\sim$ for some batches and then never again)(Kirkpatrick et al., 2017). Experimenting with *SCoLe* will show performance in the intermediate frequencies of occurrence.

We can compute the expected occurrence frequency ν^*_{class} of one class from $P_{SCoLe}(Y)$, N and C. For example, for a uniform $P_{SCoLe}(Y)$, the expected frequency of occurrence of a class can be calculated with the probability of sampling C elements from a set of N unique elements without replacement and can deduce the frequency of occurrence per task directly: $\nu^*_{class} = 1 - \Pi_{i=0}^{C-1}(1 - \frac{1}{N-i})$ per task ($task^-1$). Then the expected number of tasks between two occurrences of a given class $\tau^*_{class} = \frac{1}{\nu_{class}}$. For example, with CIFAR10 the total number of classes is 10, hence with $C = 2$ and uniform $P_{SCoLe}(Y)$, we have $\nu^*_{class} = 1 - (1 - \frac{1}{10})(1 - \frac{1}{9}) = 0.200\, task^{-1}$ then $\tau_{class} = 5$ tasks.

For the task occurrence frequency, the expected number of tasks between two occurrences of the same task depends on the total number of tasks $\binom{N}{C}$. With CIFAR100 and 2 classes per task, $\tau_{task} = \binom{100}{2} = 4950$. Thus, revisiting exactly the same task in this setup is very rare (every 4,095 tasks). Note for standard CL τ_{task} and τ_{class} are $\inf$. In this paper, the occurrence of frequency is always the expected occurrence frequency (a priori set) and not the empirical frequency (measured by counting the number of occurrences).

Summary. *SCoLe* is a continual learning framework for generating long sequences of tasks with various frequencies of tasks and classes. It is made to study the knowledge accumulation capability of learning algorithms. *SCoLe* scenarios are in between IID scenarios and usual CL scenarios, as the data may reoccur, but the distribution changes through time and forgetting can compromise performance. *SCoLe* can be seen as a bridge between both setups since, depending on the frequency occurrence of classes, we can be closer to one or the other.

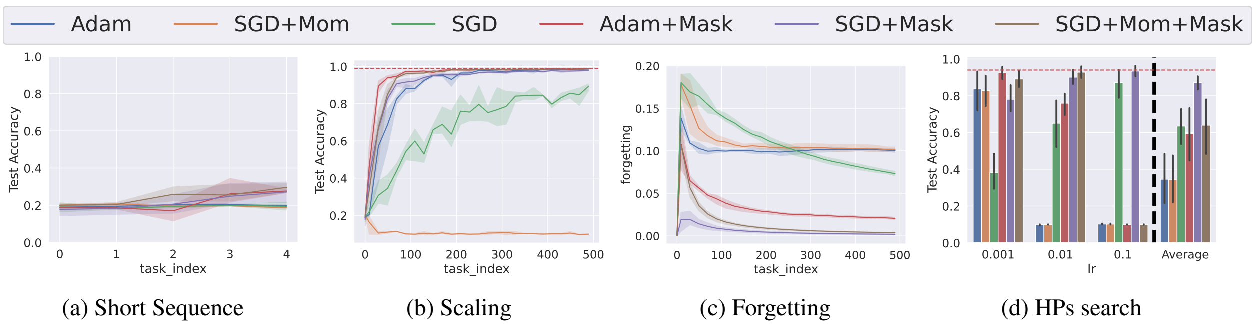


Figure 5: Knowledge accumulation on MNIST over a short sequence of tasks (a), when scaling the number of tasks (b). Estimation of forgetting through time (c). The Hyper-parameters search (d) is realized by averaging performance on scenarios built from MNIST, Fashion MNIST and KMNIST. Results show that masking gradient and removing momentum on SGD consistently increase knowledge accumulation leading to a reduction of forgetting.

4 Knowledge Accumulation

As documented in various classical works on CL, including French (1999), CF typically leads to a decline in performance on previously learned tasks. Hence, we could expect that in any non-IID scenario, forgetting might have a strong effect on performance. Building upon the insights presented in Sec. 2, we investigate knowledge accumulation in a simple *SCoLe* setup. In contrast to Sec. 2, where the task performance where measure independently for other tasks, we now measure the overall 0-shot performance (full test set) set with a single-head architecture.

4.1 Initial Experiments on *SCoLe*

In this first experiment, we evaluate knowledge accumulation in *SCoLe* using MNIST dataset. We use the two most popular optimizers in CL and ML in general: SGD with momentum (Qian, 1999), and Adam (Kingma & Ba, 2014). By default, we kept the optimizer's hyper-parameters of PyTorch (Paszke et al., 2019) and train a small convolutional neural network (c.f. Appendix F).

Inspired by recent papers that show that masking the gradient in the last layer helps classifiers to learn continually (Caccia et al., 2022; Zeno et al., 2018) even without any supplementary memorization process (Lesort et al., 2021a). We also propose to use two simple modifications to SGD (1) removing the momentum, (2) masking the output layer's gradient for classes not currently in the task in the output layer: "gradient masking". More precisely, for masking, we replace outputs of classes that are not present in the current mini-batch by $-1e9$ to prevent backpropagation and to have minimal influence on the softmax activation. On the one hand, intuitively, in the presence of data distribution drifts, momentum (also present in Adam) produces a mixture between the gradient of the previous task and the gradient of the current task, which can create interference in the training process. On the other hand, when training on a subset of classes, the model does not learn anything about other classes. Hence masking the gradient for those outputs avoids interfering with updates for other classes.

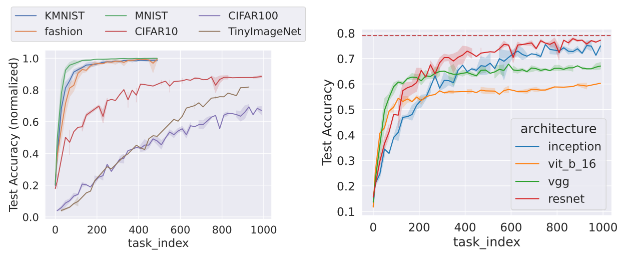
We see in Figure 5a that during the first tasks, no baseline seems to accumulate knowledge, which is in accordance with the common understanding of the CF phenomenon: CF erases past knowledge, and the overall performance depends only on the current task (French, 1999). However, when we scale the number of tasks and let reoccurrences happen, in Figure 5b, we see that performance increases quickly until reaching IID accuracy, at least for most baselines. We also see that forgetting decreases through time in Figure 5c. We note that since classes reoccur, forgetting needs to be measured differently than in usual scenarios (van de Ven & Tolias, 2019) where we track accuracy on all tasks seen so far. We propose to evaluate forgetting within a task by measuring the decrease in accuracy among classes seen so far and not in the current task. This gives us the average performance lost in known classes while learning the current task. The details of the calculation are in Appendix G.

4.2 Robustness of Knowledge Accumulation

Baseline and Hyperparameters: To find the best setup to run our further experiments, we run a small hyperparameter search on the same scenario with MNIST, Fashion-MNIST, and KMNIST. Figure 5d shows the average performance on the three datasets with various learning rates. SGD without momentum and with masking is a stable baseline, as it leads to knowledge accumulation consistently with all learning rates. Hence, We will experiment with this baseline in further experiments. Adam and SGD with momentum can also achieve knowledge accumulation, even without masking; however, our results show it is more sensitive to HPs.

Datasets and Architectures: We investigate if knowledge accumulation happens consistently among datasets and architectures.

We create scenarios on MNIST, Fashion-MNIST and KMNIST with 500 tasks, CIFAR10, CIFAR100 and tiny-ImageNet with 1,000 tasks (3 seeds). We experiment with the baseline SGD+Mask on all of them. For CIFAR100 and tinyImageNet, we set $C = 5$, while for the other datasets $C = 2$. In addition, we train several architectures (Resnet18, Inception, vit_b_16 and VGG) from the torch library and compare them in a default *SCoLe* scenario on CIFAR10 with 2 classes per task. Figure 6a shows the learning curve on the various datasets. We normalize accuracies by the IID accuracy to make curves comparable. The IID test accuracies are: MNIST 99%, Fashion MNIST 89%, KMNIST 94%, CIFAR10 79%, CIFAR100 40%, and miniImageNet 20%. The IID accuracies were obtained with the same models as in *SCoLe* experiments, with Adam with default parameters and without data augmentation. This figure shows that knowledge accumulation occurs consistently in all these datasets. Figure 6b shows that on CIFAR10, knowledge accumulation consistently happens with various types of architectures.



(a) Datasets (b) Architectures

Figure 6: (left) Test acc. on *SCoLe* with T scaled up to 1,000 tasks averaged over MNIST, Fashion-MNIST, KMNIST with 3 seeds. (left) normalized test accuracy with various datasets on Scole (the accuracy is divided by the IID accuracy), and (right) experiment with various architectures on CIFAR10, (2 classes per tasks). - - - line is the best IID performance. The learning rate is set at 0.01.

Summary. In this section, we showed that when data reoccurs through long training sequences, as in *SCoLe*, gradient-based optimization may overcome catastrophic forgetting and accumulate knowledge leading to progressively improved overall 0-shot accuracy. This shows the limited effect that CF might have and the influence of the knowledge accumulation (measured with meta-test probing in Section 2) on 0-shot performance.

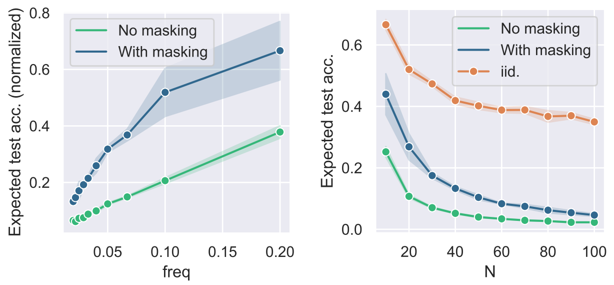
5 Effect of non-stationarity on continual learning with SGD

In the previous section, we investigated knowledge accumulation with one frequency of occurrence per dataset and for all classes. In this section, we explore more in-depth the impact of varying class occurrence frequencies on the ability of DNNs to learn continually. We explore two cases, one where $P_{SCoLe}(Y)$ is uniform; hence the expected frequency of occurrence is the same for all classes and a second where $P_{SCoLe}(Y)$ is not uniform.

5.1 Uniform Occurrence Frequency

In this section, we vary the frequency of occurrence ν^*_{class} by varying the total number of classes N in *SCoLe* scenarios with fixed classes per tasks $C = 2$ and the number of tasks $T = 1000$. Growing N with fixed C, lowers the probability of sampling each class and therefore ν^*_{class}. $P_{SCoLe}(Y)$ is uniform, i.e. ν^*_{class} is the same for all classes.

We use a subset of N classes from CIFAR100 to create various scenarios. Figure 7a shows the accuracy of scenarios with various ν^*_{class}. We normalized the test accuracy by the IID accuracy on the same data to only assess the effect of ν^*_{class} on the knowledge accumulation without the effect of increasing the problem difficulty.

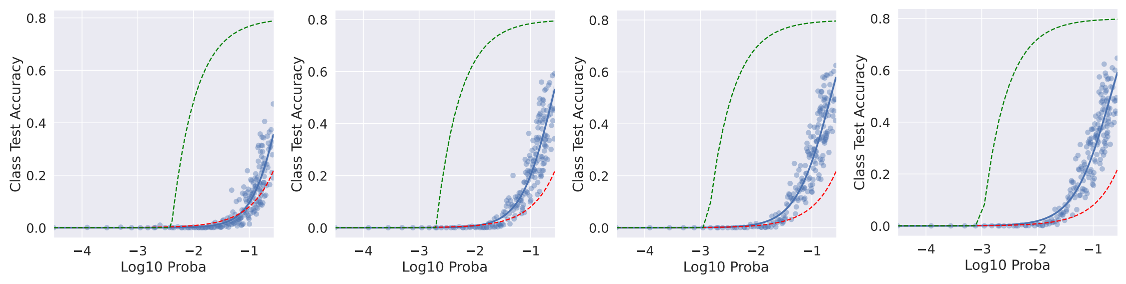


(a) Occurrence Frequency (b) Number of classes

Figure 7: Expected performance depending on ν_{class} (Figure 7a). ν_{class} is influenced by the total number of classes N. To remove the influence of the task difficulty, we normalize by IID accuracy for each number of classes (Figure 7b).

This experiment shows that lowering occurrence frequencies slows down knowledge accumulation. As frequency of occurrence gets lower, further scaling of the number of tasks is necessary to get closer to IID accuracy. We see that for classes appearing more frequently than every 10 tasks (0.1 frequency), the expected normalized performance is higher than 50% of the IID performance. This result demonstrates that SGD-based training can learn without forgetting and accumulate knowledge through a sequence of tasks, in particular when using masking.

5.2 Mixture of Occurrence Frequency



(a) $t \in [0, 250]$ (b) $t \in [250, 500]$ (c) $t \in [500, 750]$ (d) $t \in [750, 10^3]$

Figure 8: Accuracy for classes vs occurrence probability over different task ranges. Looking at the 4 plots, we see that the model increases performance through time for classes with higher occurrence probabilities. Lower bound - - -: Expected performance when the model learns at 80% accuracy for current classes and forgets everything after. Upperbound - - -: Expected performance when the model learns up to 80% each task and never forget after.

In previous experiments, all ν^*_{class} were the same. In this experiment, we give a different ν^*_{class} to each class, and we analyse class performances independently from each other. This setting is built to not depend on the uniform distribution of classes in the results.

In this scenario, we use CIFAR100 dataset, $P_{SCoLe}(Y)$ is not uniform anymore, and each class is given a probability of sampling on a wide range of probability. Then, for each task, we sample 10 classes from $P_{SCoLe}(Y)$ to create tasks. Details on the implementation of how to create a distribution of various probabilities are in Appendix D. In this setting, we estimate $\nu^*_{class} = 1 - \Pi_{i=0}^{C-1}(1 - P_{SCoLe}(Y = class) * \frac{N}{N-i})\ task^{-1}$. We can observe in Figure 8 the performance at different task intervals. Although one might expect to observe improvement across all frequency ranges, the results indicate that only the most frequent classes display enhanced performance. This finding suggests that to improve results on comparable benchmarks, it may be advantageous to prioritize low-frequency classes. Additionally, while the model outperforms the complete forgetting baseline, it still lags significantly behind a model that never forgets.

Summary. In this section, we empirically evaluate how SGD with masking accumulates knowledge with various class occurrence frequencies. The evaluation was conducted both for equal (uniform $P_{SCoLe}(Y)$) and distinct class frequencies. Our experiments detail how well classes are learned and remembered depending on their occurrence frequency. One of our results is that after a significant number of tasks and when using masking, CIFAR100 classes reoccurring more than once every 10 tasks are remembered with over 40% of accuracy.

6 Increasing Knowledge Accumulation in DNNs

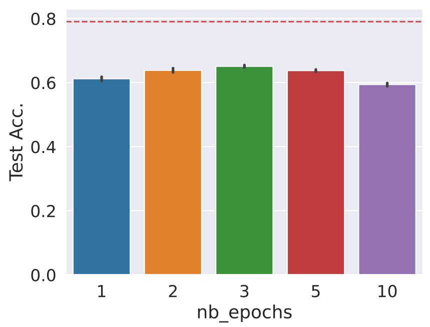
6.1 Learning Forgetting Trade-off

A straightforward approach to minimize the effect of catastrophic forgetting could be to minimize the number of training steps in each task. It would minimise forgetting during a given task, maximize the randomization of data, and get closer to an IID training regime. On the other hand, it could also significantly limit the capability to learn the tasks themselves.

In this section, we investigate this strategy. We introduce the concept of "learning-forgetting trade-off". While learning on a new task, the model increases its knowledge about this new data but also might forget what was learned before. The right learning-forgetting trade-off maximizes overall knowledge and aims to ensure that learning brings more knowledge than forgetting.

Setting: In this experiment, we use the CIFAR10 dataset with $C = 2$ classes per task uniformly sampled with a scenario of $N = 1000$ tasks. We vary the batch size and the number of epochs to control the number of gradient steps per task.

Results: Figure 9 shows that the number of epochs per task or the batch size that maximize the performance are not the ones that minimize the number of gradient steps per task. Growing the number of steps per task might actually improve performance, i.e. the model learns more than it forgets, at least until a certain point, as shown when we increase the



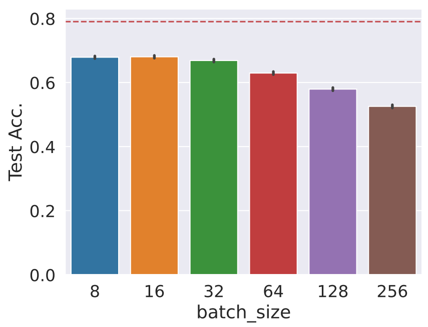


Figure 9: Average test acc. on 1k tasks vs epochs per task (top) and batch size (bottom). Performance may improve with more gradient steps per task but too many gradient steps on each task might lead to performance loss. - - - IID baseline.

number of epochs. If the gradient steps' number is increased too much, the model starts to forget more than it learns. We can note that the ordering of the best number of epochs or best batch size might evolve with the number of tasks. For more detail, we present the result for the same experiments with the test accuracy per task in appendix Figure 11.

This experiment shows the trade-off between learning new information and forgetting previous ones. At scale, algorithms should then find the right amount of new information to learn to maximize this trade-off and the overall performance.

6.2 Model Width

Inspired by Mirzadeh et al. (2021), in this section, we would like to investigate how different widths of resnet models learn on *SCoLe* setup. We are also interested in understanding if making models wider improve performance in all range of frequency of appearance.

We define the bandwidth of SGD as the ν_{class} range that SGD is able to learn without forgetting and accumulating knowledge. We train on the CIFAR100 dataset with SGD (lr=0.1, momentum=0) on one epoch per task. All classes have a different probability of being sampled when building a task as in Section 5.2.

Table 1: Average Performance at different ν_{class} range over the last 100 tasks (over 1000) of various model widths.

Size \ Probabilities	$10^{-4} \leq p < 10^{-3}$	$10^{-3} \leq p < 10^{-2}$	$10^{-2} \leq p < 10^{-1}$	$10^{-1} \leq p < 1$
1	0.0 %	0.2 %	15.4 %	41.4 %
2	0.0 %	0.2 %	18.2 %	43.8 %
4	0.0 %	0.4 %	22.3 %	44.2 %
8	0.0 %	0.7 %	26.4 %	44.5 %
16	0.0 %	1.6 %	32.0 %	45.9 %

We present the results of this experiment on various ν^* ranges in Table 1. As in Mirzadeh et al. (2021), the *SCoLe* setup confirms that wider models perform better. Moreover, interestingly, we see that most of the performance improvement is focused on the intermediate frequency range of $\nu_{class} \in [1e{-}2, 1e{-}3]$.

These results show that wider models improve performance but also increase the bandwidth of SGD with masking.

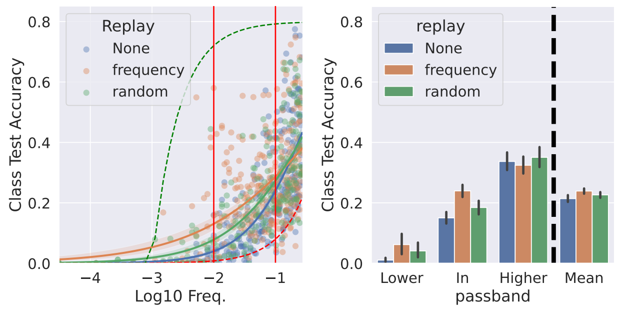
6.3 Frequency Replay

As we have seen in previous sections, SGD training on DNNs leads to knowledge accumulation, a fortiori if the occurrence frequency of classes is high. In this section, we show how to benefit from a better understanding of knowledge accumulation to improve the compute efficiency of a vanilla replay approach.

Approach: Replaying past data is a standard approach to CL. However, replay approaches usually do not take into account the capability of SGD to accumulate knowledge. Replay strategies are either based on random sampling (Rebuffi et al., 2017; Chaudhry et al., 2019) or based on current loss space (Borsos et al., 2020; Aljundi et al., 2019). Therefore, as a proof of concept that we can benefit from both knowledge accumulation and replay mechanisms, we propose to replay based only on the class occurrence frequencies. The data from all classes is saved in the same way, but data from classes that appear frequently are not replayed.

Based on experiments in Section 5.2 and Figure 8, we assume that our model effectively sufficiently accumulates knowledge for classes with $\nu_{class} > 0.1\ tasks^{-1}$ and consequently, we replay other classes. However, to limit the compute cost of our replay mechanisms, we do not replay too rare classes ($\nu_{class} < 0.01\ tasks^{-1}$). The *passband* of our frequency replay mechanism, i.e. the ν_{class} range that is replayed, is $[0.01, 0.1]$. To resume, the frequency replay mechanisms target classes that appear every 10 to 100 tasks.

Implementation Details: Each ν_{class} is estimated by counting occurrences during the scenario. To have an estimate of frequency on several occurrences of one class, the replay process may be triggered after $\tau = 3$ occurrences (c.f. Algorithm 1). The buffer saves $n = 200$ samples per class. For each revisit, $max(n/o, n/20)$ samples are renewed, where o is the number of occurrences of the class so far. We set a minimum of renewable by principle to avoid the replay buffer becoming frozen. While replaying, we oversample the buffer in order to keep a number of samples per class balanced during the task as in Ostapenko et al. (2022).



(a) Frequency Replay. (b) Acc. per frequency range

Figure 10: Class mean performance over the last 100 tasks vs ν^*_{class} (log). In Figure 10a, the vertical red lines are the upper and lower cutoff frequencies of the replay mechanism. Figure 10b plots the average acc. depending on if ν^*_{class} is within the passband of the replay mechanism, lower or higher. Frequency replay improves performance in the desired frequency range and on average.

In this setting, we evaluate the advantage of the frequency replay strategy. We compare frequency replay with simple fine-tuning and random replay. To increase the difficulty of the previous settings and avoid the repetition of the exact same data point, we use TinyImagenet dataset with the augmentations proposed in Hendrycks & Dietterich (2019). Those augmentations simulate common perturbation and corruption in images. At each task, a random augmentation is selected and applied with severity 2. We use a subset of 100 classes of TinyImagenet and train with 10 classes per task to have the same probability distribution as in Section 5.2. In this setting, the compute overload of frequency replay over normal fine-tuning is 21.28%. In comparison, doing a full replay strategy, i.e. replaying all classes seen so far (and not in the current task), would increase the compute by 873% (details in Appendix E). In our experiments, instead of comparing with full replay, we compare the frequency replay with random replay with the same compute budget. The experiments are run on 10 different seeds.

We gathered results in Figure 10. This figure shows the positive effect of frequency replay within the frequency range of interest and overall (Average) compared to fine-tuning and random replay [1]. We observe that in lower frequencies, performances also improve. Indeed, the x-axis is the expected frequency of occurrence and not the empirical frequency of occurrence. If empirically, a low-frequency class appears several times at the beginning of the scenario, it will be “captured” by the replay mechanism and be replayed regularly. The results with this simple replay mechanism illustrate as a proof of concept the impact of better understanding knowledge accumulation in DNNs to improve the scalability and efficiency of continual learning approaches. A related idea to our frequency replay, was proposed in Hemati et al. (2023), where the amount of data stored in the replay buffer depends on the frequency of appearance of classes. However, Their approach is meant to optimize memory footprint while ours is meant to optimize compute footprint.

Algorithm 1 Frequency Replay

```
Input: $\nu_{low}$, $\nu_{high}$, $\tau$, $C_{dict}$, $C_t$, nb_batch.
nb_batch $\leftarrow$ nb_batch $+ 1$
$classes2replay = []$
for $class = 0$ in $C_t$ do
    $C_{dict}[class] += 1$
end for
for $class = 0$ in $C_{dict}.keys() \backslash C_t$ do
    $freq = C_t[class]/$nb_batch
    if $C_t[class] > \tau$ & $freq \in [\nu_{low}, \nu_{high}]$ then
        classes2replay.append(class)
        $C_{dict}[class] += 1$
    end if
end for
Return: classes2replay
```

Note: In classical CL scenarios, classes do not reoccur throughout the sequence of tasks. Nevertheless, frequency replay could still be used. For example, instead of replaying every 10 tasks for $\nu^*_{class} = 0.1$, in a classical setup classes could be replayed every 10 epochs to reduce compute cost while maintaining the advantage of replay. This method was notably used for continual learning for text classification and question answering in de Masson d'Autume et al. (2019).

[1]The p-value of frequency replay results being greater than random replay results is 0.04 in this experiment. With the usual significance level at 5%, it means that the difference is significant.

Summary. Making models wider, replaying based on frequency and/or optimizing the learning-forgetting trade-off strategies are scalable and efficient approaches to improve the knowledge accumulation of SGD. Those results motivate the importance of understanding the dynamics of learning and forgetting in the long run in order to design algorithms that achieve optimal performance-compute trade-offs in practical settings.

7 Related Work

Most scenarios in the continual learning literature study catastrophic forgetting (van de Ven & Tolias, 2019; Lesort et al., 2021b). Such scenarios consist of a sequence of tasks where data appears in a single task. Those settings evaluate whether models can remember tasks that they have seen only once. The no reappearance constraint makes the evaluation of models clear since tasks overlap can not interfere with forgetting. However, they cannot study if approaches such as SGD retain and accumulate knowledge through time. This is the main difference with the present work.

The scenario we propose has some similarities to the CMR (Lin et al., 2022a) and OSAKA (Caccia et al., 2020) frameworks. However, in our setting, we do not evaluate fast adaptation but rather the capacity to learn classes and remember them depending on their occurrence frequency. Moreover, there is no real concept shift in *SCoLe*. That is, $p(y|x)$ is fixed over time. Our evaluation protocol is also similar to the ALMA scenario (Caccia et al., 2021), since we evaluate on a fixed test set. However, in ALMA the data distribution does not drift, while in *SCoLe*, there are drifts between tasks. A scenario with a long sequence of tasks was proposed by Wortsman et al. (2020). Different permutations of pixels for each task are used. However, there is no task re-occurrence, and SGD cannot accumulate knowledge. Stojanov et al. (2019) experimented with scenarios with the re-occurrence of objects and find out that it reduces the effect of forgetting. However, they did not analyse this finding by varying the frequency of occurrence. From a different perspective, since classes might reoccur through time, the scenario can be assimilated to sequences of similar and dissimilar tasks (Ke et al., 2020).

On a similar line as this work, Evron et al. (2022) theoretically showed that for linear regression trained with SGD, knowledge accumulation leads to a reduction of CF when tasks reoccur. Other works investigate the effect of forgetting on weights: Ramasesh et al. (2021) shows that CF impact mostly the weight on the higher layers. Doan et al. (2021); Asanuma et al. (2021) use the modification of weights between two tasks as a proxy to measure the similarity of tasks.

The findings of this paper that DNNs consistently retain and accumulate knowledge could provide an explanation for the observation that self-supervised models are continual learners (Fini et al., 2020), language models are continual learners (Scialom et al., 2022), or that large scale pertaining reduce forgetting (Ramasesh et al., 2022). On the one hand, in settings such as self-supervised learning and language modelling, knowledge accumulation is improved by training with optimised general losses, which increase transfer between tasks and favour the re-occurrence of concepts and features in data. On the other hand, when starting from a model pre-trained on a large task, long-term knowledge retention benefits all downstream tasks, especially in short sequences. Nevertheless, if those results might be connected into common learning and forgetting dynamics, the specificities of each setup remain a factor of variability in results.

8 Conclusion

This paper investigates knowledge retention (KR) and accumulation (KA) in deep neural networks. We show that contrary to what most CL literature shows, catastrophic forgetting has a limited effect on DNNs. Indeed, despite initial catastrophic forgetting observed in early tasks, DNNs still retain knowledge about past tasks even after learning several unrelated tasks and can accumulate knowledge from every training stage. We propose the *SCoLe* framework to create scenarios with various frequencies of class occurrences and study knowledge accumulation at scale. Additionally, to improve knowledge accumulation in DNNs, we experiment with simple and scalable strategies: (1) masking the gradient in the last layer, (2) optimizing the number of training steps per task to maximize the learning-forgetting trade-off, (3) making the model wider, or (4) replaying rare classes. We believe that assessing the efficiency of algorithms at scale with regard to frequency occurrence could greatly improve the evaluation of algorithms for later application into real scenarios.

References

Rahaf Aljundi, Lucas , Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximal interfered retrieval. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 11849–11860. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9357-online-continual-learning-with-maximal-interfered-retrieval.pdf.

Haruka Asanuma, Shiro Takagi, Yoshihiro Nagano, Yuki Yoshida, Yasuhiko Igarashi, and Masato Okada. Statistical mechanical analysis of catastrophic forgetting in continual learning with teacher and student networks. *Journal of the Physical Society of Japan*, 90(10):104001, 2021.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 33:3884–3894, 2020.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.12.003. URL https://www.sciencedirect.com/science/article/pii/S0893608020304202.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptative quantization module. *arXiv preprint arXiv:1911.08019*, 2019. URL https://arxiv.org/abs/1911.08019.

Lucas Caccia, Jing Xu, Myle Ott, Marc'Aurelio Ranzato, and Ludovic Denoyer. On anytime learning at macroscale. *arXiv preprint arXiv:2106.09563*, 2021.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *NeurIPS*, 2020. URL https://arxiv.org/abs/2003.05856.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019. URL http://arxiv.org/abs/1902.10486.

Andrea Cossu, Gabriele Graffieti, Lorenzo Pellegrini, Davide Maltoni, Davide Bacciu, Antonio Carta, and Vincenzo Lomonaco. Is class-incremental enough for continual learning?, 2021.

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16712–16721, 2022.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f8d2e80c1458ea2501f98a2cafadb397-Paper.pdf.

Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1072–1080. PMLR, 13–15 Apr 2021. URL http://proceedings.mlr.press/v130/doan21a.html.

Arthur Douillard and Timothée Lesort. Continuum: Simple management of complex continual learning scenarios. 2021. URL https://arxiv.org/abs/2102.06253.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *Conference on Learning Theory*, pp. 4028–4079. PMLR, 2022.

Enrico Fini, Stèphane Lathuilière, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123730715.pdf.

Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2022.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 13646613. doi: 10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/abs/pii/S1364661399012942.

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

Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020. doi: 10.1016/j.tics.2020.09.004.

Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. REMIND Your Neural Network to Prevent Catastrophic Forgetting. 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123650681.pdf.

Hamed Hemati, Andrea Cossu, Antonio Carta, Julio Hurtado, Lorenzo Pellegrini, Davide Bacciu, Vincenzo Lomonaco, and Damian Borth. Class-incremental learning with repetition. *ArXiv*, abs/2301.11396, 2023.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems*, 33:18493–18504, 2020. URL https://proceedings.neurips.cc/paper/2020/file/d7488039246a405baf6a7cbc3613a56f-Paper.pdf.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/bcd0049c35799cdf57d06eaf2eb3cff6-Abstract.html.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual pre-training of language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m_GDIItaI3o.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the national academy of sciences*, 2017. URL https://www.pnas.org/content/pnas/114/13/3521.full.pdf.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. 2019. URL https://arxiv.org/abs/1909.08383.

Timothée Lesort, Thomas George, and Irina Rish. Continual learning in deep networks: an analysis of the last layer. *arXiv preprint arXiv:2106.01834*, 2021a. URL https://arxiv.org/abs/2106.01834.

Timothée Lesort, Andrei Stoian, and David Filliat. Regularization shortcomings for continual learning. *arXiv preprint arXiv:1912.03049*, 2019.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52 – 68, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.004. URL http://www.sciencedirect.com/science/article/pii/S1566253519307377.

Timothée Lesort, Massimo Caccia, and Irina Rish. Understanding continual learning settings with data distribution drift analysis. *arXiv preprint arXiv:2104.01678*, 2021b.

Bill Yuchen Lin, Sida Wang, Xi Victoria Lin, Robin Jia, Lin Xiao, Xiang Ren, and Wen-tau Yih. On continual model refinement in out-of-distribution data streams. *arXiv preprint arXiv:2205.02014*, 2022a.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Beyond not-forgetting: Continual learning with backward knowledge transfer. *ArXiv*, abs/2211.00789, 2022b.

David Lopez-Paz and Marc-Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6467–6476. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7225-gradient-episodic-memory-for-continual-learning.pdf.

Seyed Iman Mirzadeh, Arslan Chaudhry, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. *arXiv preprint arXiv:2110.11526*, 2021.

Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. 2020. URL https://arxiv.org/abs/2009.01797.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LhY8QdUGSuw.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017. URL https://arxiv.org/abs/1611.07725.

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. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1gTShAct7.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Continual-t0: Progressively instructing 50+ tasks to language models without forgetting. *arXiv preprint arXiv:2205.12393*, 2022.

Stefan Stojanov, Samarth Mishra, Ngoc Anh Thai, Nikhil Dhanda, Ahmad Humayun, Chen Yu, Linda B. Smith, and James M. Rehg. Incremental object learning from contiguous views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. URL https://arxiv.org/abs/1904.07734.

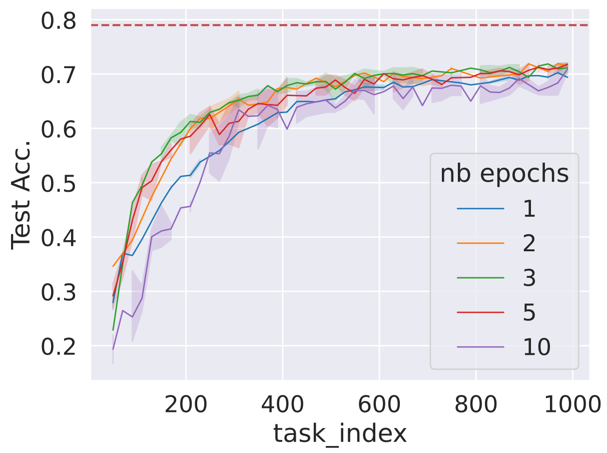
Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient continual learning with modular networks and task-driven priors. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=EKV158tSfwv.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020. URL https://arxiv.org/abs/2006.14769.

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. 2018. URL https://arxiv.org/pdf/1803.10123.pdf.

A Details

A.1 Details on Figure 9



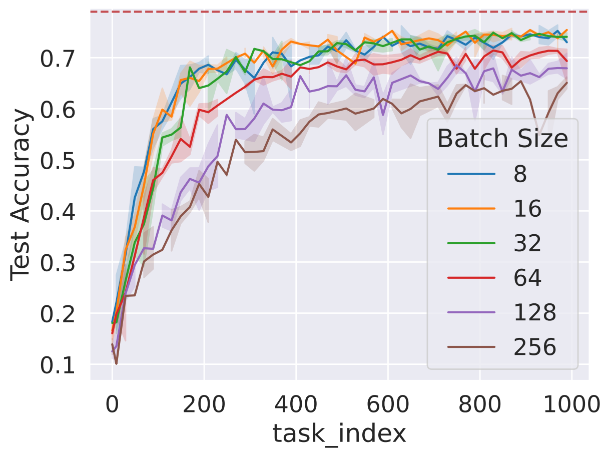


Figure 11: Test acc. on 1k tasks vs epochs per task and batch size. Performance may improve with more gradient steps per task but too many gradient steps on each task might lead to performance loss. - - - IID baseline.

A.2 KL divergence *SCoLe*vs IID.

Measuring divergence to IID.

In IID training, samples are drawn identically and independently from a static distribution. Hence, to break the IID assumption, we use the *SCoLe* framework presented in Section 3, which constrains the sampling to be on a subset of classes that change through time. Hence, samples will not be identically and independently sampled. To estimate how the *SCoLe* framework differs from IID training, we compute the Kullback Leibler divergence between the IID and *SCoLe* distribution and find that for N classes and C classes per task, $D_{KL}\big(p(X,Y|S_t)||P_{iid}(X,Y)\big) = \log\frac{N}{C}$.

For a given task t, the KL-divergence between the data distributions is:

$$\begin{aligned} &D_{KL}\big(p(X,Y|S_t)||p_{iid}(X,Y)\big) \\ &= \sum_{x\in X, y\in Y} p_t(x,y|S_t)\log\frac{p_t(x,y|S_t)}{p(x,y)} \\ &= \sum_{x\in X, y\in Y} p(x|y)p(y|S_t)\log\frac{p(x|y)p(y|S_t)}{p(x|y)p(y)} \\ &= \sum_{y\in Y} p(y|S_t)\log\frac{p(y|S_t)}{p(y)}\sum_{x\in X}p(x|y) \\ &= \sum_{y\in Y}\frac{\mathbb{1}_{\{y\in S_t\}}}{C}\log\frac{N}{C} = \log\frac{N}{C} \end{aligned} \tag{1}$$

Since $P_{SCoLe}(Y))$ is uniform over elements in set S_t, $P_{SCoLe}(Y=y)=\frac{1}{N}$. Since each of the T tasks is equally probable, the expected divergence is as follows:

$$\mathbb{E}[\log\frac{N}{C}] = \sum_{t=1}^{T}\frac{1}{T}\log\frac{N}{C} = \log\frac{N}{C}. \tag{2}$$

This result shows that *SCoLe* data distribution is different from an IID distribution and that we can control the KL divergence by changing N and C. Nevertheless, in the later experiments, instead of referring to the divergence, we will analyse the expected class frequencies ν^*_{class}. Indeed, the frequency of classes is easier to estimate in

A.3 Compute Estimation.

To estimate the compute overload of frequency replay (21.28%) and full replay (873%), we count the number of times any class is replayed. In a sequence of 2500 tasks with 10 classes per task, the finetuning approaches process the data of the equivalent of $2500 \times 10 = 25000$ classes. For frequency replay, the replay is triggered to replay a class 5320 times within the 2500 tasks, while with full replay, it would be 193445 times. Hence the estimated total compute of replay is $(25000+5320) \div 25000 = 1.2128$ time the compute of finetuning, and dense replay $(25000+193445) \div 25000 = 8.23$ times the compute of finetuning.

Samplewise, in total, fine-tuning approaches roughly processed 1.25e7 samples (2500 tasks with 10 classes per task, and approximately 500 samples per task), finetuning with frequency replay processed 1.5e7 while finetuning with full replay would have processed 1.09e8 samples.

To create random selection with the same compute budget, replay is realized by randomly and uniformly selecting a subset of classes in the buffer and replaying the associated data. The number of classes is selected, such that the overall compute growth by 21.28% over simple finetuning.

B Additional Experiments

B.1 Reducing Variance of Expected Time between Repetition

In the original *SCoLe* setup (Section 3), the class distribution is stationary, hence sampling a new task is independent of what happened previously. This phenomenon can lead to a high variance in the number of intermediate tasks between two reoccurrences of a class. In order to reduce this variance, here we introduce a penalty factor γ to penalize the resampling of classes that were sampled recently. Intuitively, we want to make the sampling probability of a particular class proportional to the time passed since it's last occurrence.

We implement this idea in the following way: if a class is sampled in the current task, the probability of sampling this class in the next task is divided by γ. However, by applying this reduction the sum of the vector of probability becomes lower than 1 and a renormalization is applied to fix this. The reduction + renormalization strategy decreases the probability of sampled classes and increase the probability for other classes. One example, if we start with uniform distribution with 5 classes we have a vector of probabilities $p = [0.2, 0.2, 0.2, 0.2, 0.2]$, if we sample two classes $[0, 2]$ with a gamma factor of 2, the unnormalized probability distribution becomes $p = [0.1, 0.2, 0.1, 0.2, 0.2]$ and after normalization $p = [0.125, 0.25, 0.125, 0.25, 0.25]$. When applying this scheme sequentially we maximize the chance of resampling classes that did not appear recently, without changing the average expected time between two occurrences, therefore reducing the variance.

Note that the higher γ is, the closer we get to a fixed sequence of tasks the sequentially reoccurs, as in Figure 1.

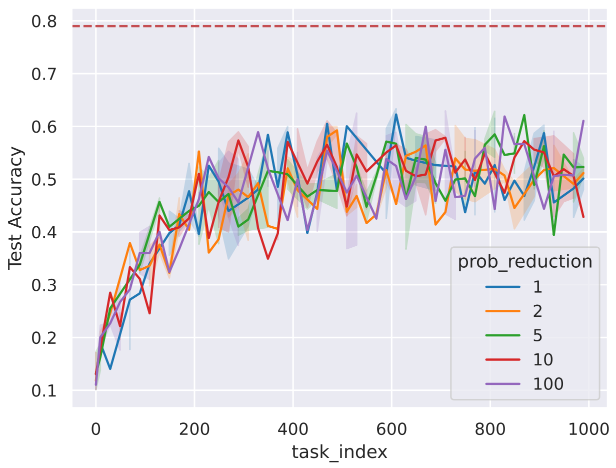


Figure 12: In this experiment, we investigated the effect of penalizing the reoccurrence of recent classes. A value of $\gamma = 1$ indicates no penalization. Surprisingly, we found that this penalization had no discernible impact on knowledge accumulation..

We run experiments with various values of γ, ($\gamma = 1$ is the same as the original *SCoLe* setup) on CIFAR10 dataset with 2 classes per task ($N = 10$, $C = 2$). The results of these experiments are shown in Figure 12, however, the penalization introduced by γ did not seem to influence performance at all. We conclude from this experience that the variance of the expected time between reoccurrences has a low influence on data accumulation, at least when the ratio $\frac{C}{N}$ is quite high.

B.2 Increasing the number of classes per task

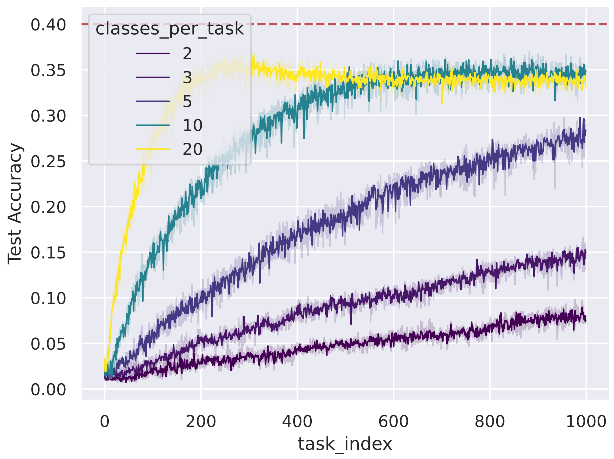


Figure 13: Growing the number of classes per task within a task in the full CIFAR100 dataset.

Figure 13, shows that our training framework also works when the number of classes per task increases. The more classes in the task, the faster the learning curve is. Here, we sample tasks from the entire CIFAR100 dataset.

B.3 Making the model deeper

In these experiments, we train on CIFAR10 with binary tasks and classes sampled uniformly. Figure 14 shows that depth has a low impact in our experiments.

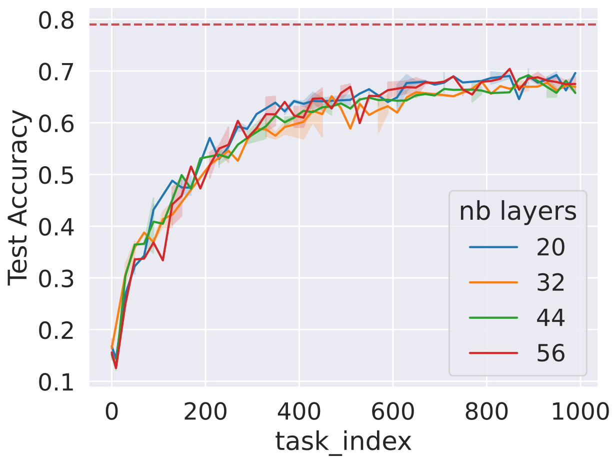


Figure 14: Growing the number of layers in the resnet model. - - - - - line represent IID performance with resnet22.

B.4 Structuring the Task Sequence

Setting: In this experiment, we want to evaluate the role of randomization of classes within tasks. In other words, we try to answer the following question: Is it important that classes are randomly sampled when building tasks? For this,

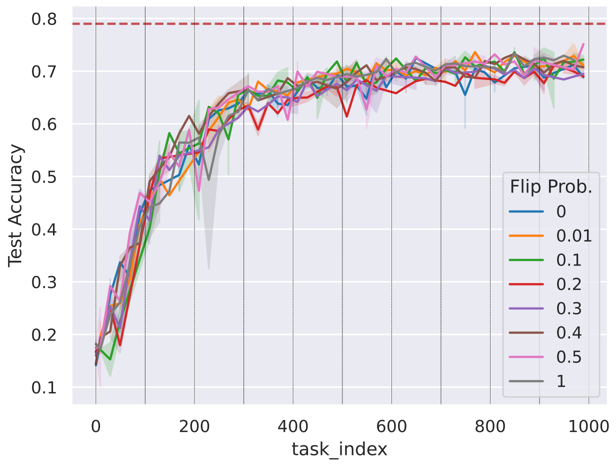


Figure 15: Comparison of several sequences of tasks with a default structure modified by a random flip of classes at each task. The scenario created with CIFAR10, 2 classes per task. The randomization of tasks is not critical for knowledge accumulation.

we start with a fixed sequence of binary classification tasks. This sequence is built so that all possible pairs of classes exist and occur only once. By default, training is achieved by repeating training on this fixed sequence of tasks until the end of the full sequence of tasks. We compare this baseline with the same sequence of tasks, but for each task, we set the probability p that each class is flipped by random to another class. The classes of the initial sequence of tasks is $[0, 1] \rightarrow [0, 2] \rightarrow [...] \rightarrow [1, 2] \rightarrow [1, 3] \rightarrow [8, 9] \rightarrow [0, 1] \rightarrow [...]$. The values of p are $0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1$. We can see that $p = 1$ is the same as in the default scenario.

Results: The results presented in Figure 15 show that having a fixed sequence of tasks instead of a randomized one does not reduce knowledge accumulation in our setting.

B.5 Limiting the Possible Pairs of Classes in Tasks

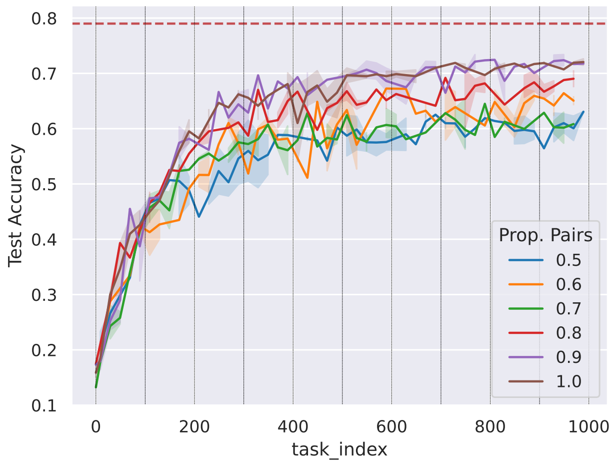


Figure 16: Comparison of *SCoLe* scenario when selecting only a subset of all possible pairs of classes within tasks. We vary the proportion of pairs kept and plot test accuracy.

Setting: In this experiment, we want to evaluate how important it is that all possible pairs of classes exist within the full sequence of tasks. For this, we start from the list of all possible tasks and select only a subset of them. When building the task sequence, we only select pairs of classes from the selected list.

Results: The results presented in Figure 16 show that the presence of all possible pairs of classes within the sequence of tasks plays an important role. In fact, without replay to learn discriminative features between two classes, classes must be on the same task (Lesort et al., 2019).

C SCENARIO IMPLEMENTATION

```
# num_classes: the total number of classes
# classes_per_tasks: number of tasks per class (2 by default)
# probability: vector defining probability of sampling each class for a task (
                                            Uniform by default)
# nb_epochs: epochs of training per task (1 by default)

import numpy as np
from continuum.scenarios import ClassIncremental
from continuum.datasets import CIFAR10

scenario = ClassIncremental(CIFAR10(config.data_dir, train=True), nb_tasks=
                                            nb_classes)
test_set = CIFAR10(config.data_dir, train=False).to_taskset()

for task_index in range(num_tasks):
    classes = np.random.choice(np.arange(num_classes), p=probability, size=
                                            classes_per_tasks, replace=
                                            False)

    # create taskset with only selected classes
    taskset = scenario[classes]
    for epoch in range(nb_epochs):
        # train the model on "taskset" data
        [...]
    # test the model on the full test set
    [...]
```

Figure 17: Pseudo-Code using continuum (Douillard & Lesort, 2021) to control the distribution imbalance in classes.

The implementation presented in Figure 17 proposes a static version of the scenario. However, the "probability" distribution can be modified through the task sequence to create never-ending drifts or cyclic drifts in the class distribution or simply to change the balance of the class distribution.

D MIXTURE OF CLASS FREQUENCY

To change the entropy of the class distribution, we start with a uniform vector of probabilities u. For each class, u gives the probability of this class being sampled for a task. To create an imbalance in class probabilities, we slightly modify this vector using $u' = u - \frac{1}{C^2} * numpy.arange(C)$, with C the number of classes. The modification is quadratic with the number of tasks that cover a wide range of occurrence frequency. To increase the imbalance in the distribution, we multiply u' by itself d times. The complete experimentation tests the values of $0, 1, 2, 5$ and 10. Note that $d = 0$ means that the distribution is uniform. Appendix D shows the Python implementation and the probability vectors for each d).

E COMPUTE

This project was realized with the use of internal clusters. Each run was performed with a single GPU, mostly NVIDIA GeForce RTX 2070, Quadro RTX 8000, and Tesla V100-SXM2-32GB. The total amount of time required for the runs is 202 days.

F ARCHITECTURES NEURAL NETWORK FIRST EXPERIMENT

The architecture of the convolutional neural network used in **??**.

```
# num_classes: the total number of classes
# lambda: hyper-parameter = 1/num_classes
# entropy_decrease: parameter that control the scale of the imbalance

import numpy as np
prob_vec = np.ones(num_classes) / num_classes
% introduction of slight imbalance
prob_vec = prob_vec - (1/num_classes) * np.arange(num_classes) * lambda
prob_vec /= prob_vec.sum()
prob_vec = prob_vec**entropy_decrease / (prob_vec**entropy_decrease).sum()

# we shuffle the vector so for each experiment the imbalance is not the same
np.random.seed(config.seed)
np.random.shuffle(prob_vec)

for task_indef in range(num_tasks):
    selected_classes = np.random.choice(np.arange(num_classes), p=prob_vec,
                                        size=classes_per_task, replace=
                                        False)
    # the we can create the task and train the model
    [...]
```

Figure 18: Pseudo-Code to create imbalance frequency of appearance.

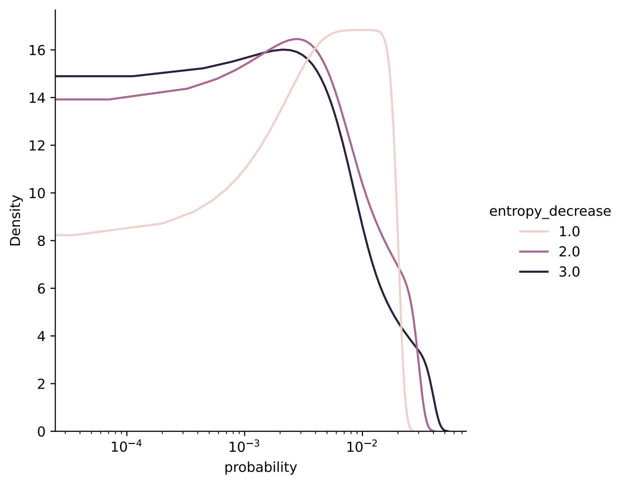


Figure 19: Density distribution of log probabilities of sampling for various entropy decrease factors.

G FORGETTING

In long task sequences, it becomes difficult and costly to track the forgetting on all tasks seen so far, hence we estimate forgetting by looking locally at how learning a new task makes the model forget the one just before. We calculate local forgetting, which is the amount of forgetting in a task-induced by learning the next task. Note: we only compute forgetting with non-overlapping classes between two tasks.

$$F_{local}(t) = \frac{1}{N-C} \sum_{j \notin \mathcal{Y}_t} A_{t,y=j} - A_{t-1,y=j} \tag{3}$$

With $\mathcal{Y}_t$ the set of classes in task t and $A_{t,y=j}$ the accuracy realized on class j at task t. $F_{local}(t)$ averages the forgetting generated in classes that are not in the current task. Then, total forgetting F averages local forgetting on tasks seen to far and is computed as:

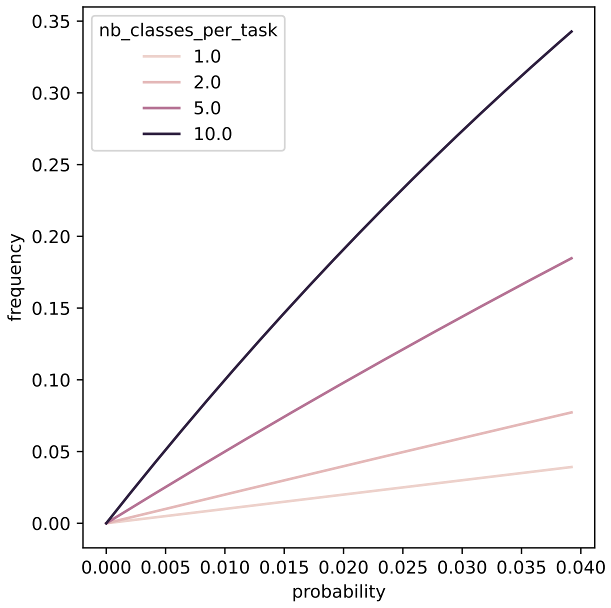


Figure 20: Probility of occurrence vs frequency of occurrence for a various number of classes per task. The number of classes per task grows the frequency of occurrence.

```
import torch.nn as nn

relu = nn.ReLU()
conv1 = nn.Conv2d(1, 10, kernel_size=5)
conv2 = nn.Conv2d(10, 20, kernel_size=5)
maxpool2 = nn.MaxPool2d(kernel_size=2)
fc1 = nn.Linear(320, 50)
head = nn.Linear(50, 10)

# Forward pass with pytorch
# x dimension is [1,28,28]
x = relu(maxpool2(conv1(x)))
x = relu(maxpool2(conv2(x)))
x = relu(fc1(x))
x = head(x)
```

Figure 21: Pseudocode describing the architecture used for experiments in **??**

$$F = \frac{1}{T-1}\sum_{i=1}^{T} F_{local}(t) \tag{4}$$

H EFFECT OF LONG-TERM DISTRIBUTION SHIFTS

In the experiments presented so far, we sampled classes for each task from the same distribution $p(S_t; C)$. Here we create *SCoLe* scenarios with a shift over time in the class distribution and assess the accumulation of knowledge under long-term shifts — shifts that persist for over several hundreds of tasks. Those shifts may result in very large or infinite τ_{class} for some classes. We evaluate knowledge retention and interference by designing three different shift patterns and evaluate models with increasing width to access whether increasing the width can slow down forgetting (Mirzadeh et al., 2021).

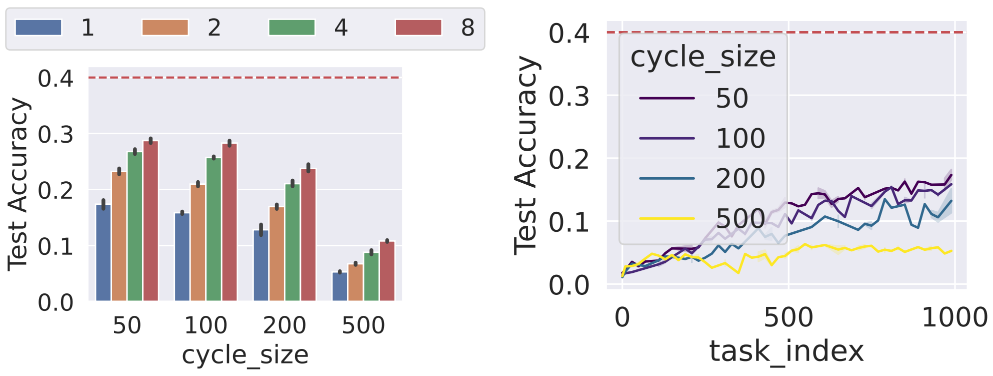


Figure 22: Cyclic Shifts Experiments: (left) shows the test accuracy averaged over the 10 last tasks for all cycle and model sizes. (right) the effect of cyclic shifts for the smaller model.

H.1 Knowledge retention with long-term class shift

We assess the capabilities of deep neural networks to maintain correct prediction on classes that stop appearing. This evaluation of knowledge retention is strict in comparison to related works (Fini et al., 2022; Davari et al., 2022) that evaluate knowledge retention by linear probing latent representation. In our setup, knowledge retention in lower layers is not sufficient, and the model also has to maintain knowledge from feature extraction in lower layers to the decision boundaries in the last layer.

Setting: We train on a scenario with $C = 2$ and uniform $p(Y|S_t)$ on CIFAR10. We start the training with all classes, and after 500 tasks, we remove half the classes from the class distribution. In contrast to standard CL, where distribution shift is usually caused by adding classes, here we remove classes. In such a scenario, the forgetting behaviour should be smooth because the error on remaining classes should be low. **Results: ??** (left) shows that even if no new data is introduced to the learner, that is, no interference with old knowledge is possible, the model can still forget when a subset of already learned classes is no longer trained on. Interestingly, in this setup forgetting is slow and not catastrophic, and knowledge persists during several hundreds of tasks. Moreover, we clearly observe that growing the width of the model increases knowledge retention to the point that it looks like the model perfectly memorized removed classes for the maximum model size.

H.2 Knowledge retention with class substitution

Similarly to the previous section, we investigate a setting with an abrupt shift in the class distribution, however instead of removing classes (shrinking the domain of $p(S_t; C)$), we replace existing classes with new (shift the domain of $p(S_t; C)$). This allows us to investigate the interference and forgetting dynamics that both cause performance drop in this setting. The goal is to assess whether observed knowledge retention can help to slow down forgetting.

Setting: First, 500 tasks are generated from the first 5 classes of CIFAR10 (first period), and the second 500 tasks are generated from the remaining 5 classes (second period). There is then no overlap between the classes in the first and second periods. Also here we test models with various widths. **Results:** The results in **??** (right) show that this sudden class shift creates disturbances in the training process as in classical CL scenarios. Moreover, in the second period of training, the model struggles more to learn the tasks than during the first period, meaning the first period does not provide good initialization for later tasks and that the forward transfer is limited in such a regime. This result corroborates the results of Ash & Adams (2020), who witnessed a similar phenomenon in a transfer setting. Similarly to the previous section, observe that wider models can better resist catastrophic forgetting. Still, this difference appears less clear, and it could be due to the better knowledge retention of first classes.

H.3 Cyclic Shifts

Here we aim to assess whether knowledge retention in DNNs observed previously persists when we fix τ_{class} to be equal for all classes but varying number of expected SGD updates $\overline{u}_{class}$ before repetitions. To this end, we let the distribution $p(S_t; C)$ follow a cyclic pattern.

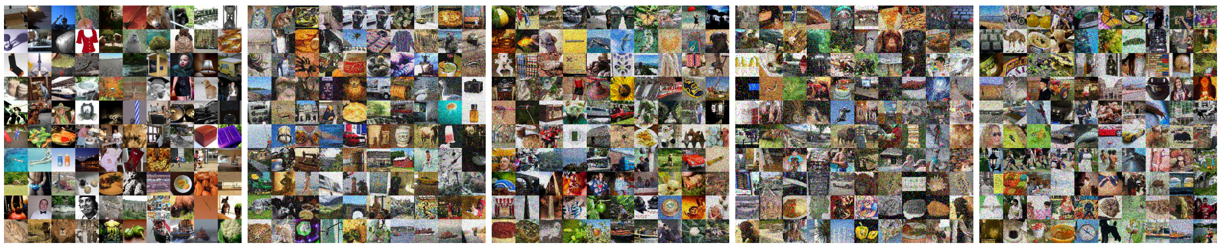

(a) 0 - Original (b) 1 - gaussian_noise (c) 2 - shot_noise (d) 3 - impulse_noisee (e) 4 - speckle_noise

Figure 23: Example of data augmentation for experiments in Section 6.3. The data augmentation is here realized with severity 4 for better visual effect. However, experiments were conducted with severity 1 in order to not make images unrecognizable.

Setting: We define a shifting window of W classes as the domain of $p(S_t; C)$ for $\frac{N}{\lambda}$ tasks after which it is shifted by one class or more depending on N and λ. Here λ is the cycle size, i.e. W is exactly the same for every λ tasks. For example, if the class subset W_t before is $[0, 1, 2]$, after a shift, it will become $[1, 2, 3]$. After λ tasks, it will be again $[0, 1, 2]$. We use the full CIFAR100 dataset ($N = 100$) with 5 classes per task, the window size is $W = 10$, and the cycle size is $\lambda \in [50, 100, 200, 500]$, higher λ leads to higher $\overline{u}_{class}$. We choose CIFAR100 because it has more classes than CIFAR10 and allows the creation of shifts over a longer period. In this experiment, when a class leaves the subset window, it needs $\lambda - W$ tasks to return leading to equal τ_{class} for all classes.

Results: We can see in Figure 22 that an increased shift period makes training harder, i.e., better learning happens if classes reoccur more frequently. Even with the largest period, the model still progresses systematically over time, as seen in Figure 22 (bottom). Consistently with other experiments, wider models result in better performance, as shown in Figure 22 (top). These experiments show that SGD-trained DNNs are also capable of long-term retention and can still accumulate even if they do not see some classes for a long period of time.

Conclusion. Forgetting still happening in long sequences of tasks with long-term distribution shifts is expected. However, experiments in this section show that models are capable of long-term knowledge retention, enabling knowledge accumulation even when data is not seen for a long time. Our results are in line with the findings of Mirzadeh et al. (2021): Wider models forget less in incremental scenarios. Further, we have also shown that the widest models are capable of almost perfect knowledge retention and that with long-term distribution shifts such as cyclic shifts, deep neural networks can accumulate knowledge. They can memorize and reuse knowledge from classes not seen since more than 200 hundred tasks.

I Experiments with Data augmentation

I.1 Samples

I.2 Supplementary Experiments

In most of our experimentation, when a class reappears, the exact same data is fed to the model. In this experiment, we want to investigate the influence of this. We compare training with exact same data and with data modified by random data augmentation.

Setting: We use the augmentations proposed in Hendrycks & Dietterich (2019) that simulate common perturbation and corruption proposed in images. At each task, a new augmentation is selected and applied with severity 1. We selected our augmentation among, “no corruption”, “gaussian noise”, “shot noise”, “impulse noise”, “speckle noise”, “gaussian blur”, “defocus blur”, “motion blur”, “zoom blur”, “fog”, “snow”, “spatter”, “contrast”, “brightness”, “saturate”, “elastic transform” and “glass blur”. They minimized the chance of having the same exact data several times, and the augmentation applied to the data is variate and significant. We use a subset of 50 of the TinyImagenet dataset with 5 classes per task.

Results: Our results are presented in Figure 24. It shows that the augmentation applied to each image to minimize the chances of having the exact same images does not compromise the accumulation of knowledge. Moreover, Figure 25 shows that when growing the number of epochs per task, augmentation can improve performance over exact data repetition.

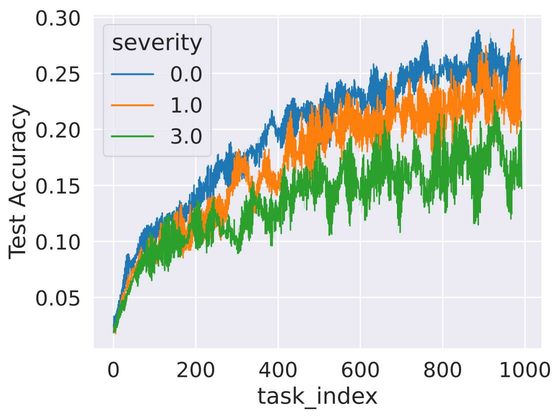


Figure 24: TinyImagenet 50/5, 1 epoch per task: Addition of random augmentation at each task with various severity. The augmentation slows down knowledge accumulation but does not prevent it.

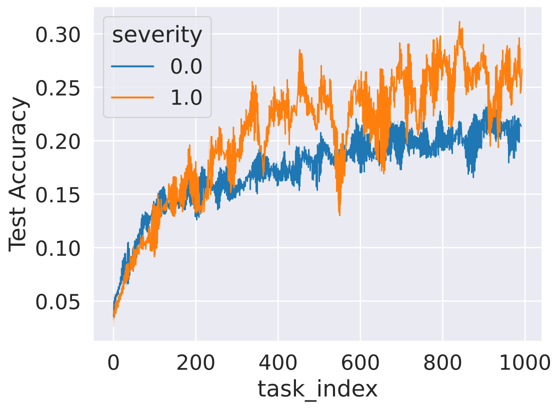


Figure 25: TinyImagenet 50/5, 5 epochs per task: when growing the number of epochs per task, a slight data augmentation improves results over exact data repetition.