CHUNKING: CONTINUAL LEARNING IS NOT JUST ABOUT DISTRIBUTION SHIFT

Thomas L. Lee & Amos Storkey
School of Informatics
University of Edinbrugh
{T.L.Lee-1@sms.,A.storkey@}ed.ac.uk

ABSTRACT

Work on continual learning (CL) has thus far largely focused on the problems arising from shifts in the data distribution. However, CL can be decomposed into two sub-problems: (a) shifts in the data distribution, and (b) dealing with the fact that the data is split into chunks and so only a part of the data is available to be trained on at any point in time. In this work, we look at the latter sub-problem, the *chunking* of data. We show that chunking is an important part of CL, accounting for around half of the performance drop from offline learning in our experiments. Furthermore, our results reveal that current CL algorithms do not address the chunking sub-problem, only performing as well as plain SGD training when there is no shift in the data distribution. Therefore, we show that chunking is both an important and currently unaddressed sub-problem and until it is addressed CL methods will be capped in performance. Additionally, we analyse why performance drops when learning occurs on identically distributed chunks of data, and find that forgetting, which is often seen to be a problem due to distribution shift, still arises and is a significant problem. We also show that performance on the chunking sub-problem can be increased and that this performance transfers to the full CL setting, where there is distribution shift. Hence, we argue that work on chunking can help advance CL in general.[1]

1 INTRODUCTION

How should we update a neural network efficiently when we observe new data? This issue remains an open problem, and is one that the field of *continual learning* (CL) addresses. Many methods (Delange et al., 2021; Parisi et al., 2019; Wang et al., 2023) and settings (Hsu et al., 2018; Antoniou et al., 2020; van de Ven & Tolias, 2019) have been proposed in recent years. Specifically, CL studies settings where a learner sees a stream of chunks of data and where the data distribution for each chunk changes over time. This type of change in the data distribution is commonly known as *task shift* (Caccia et al., 2020).

CL can be decomposed into two sub-problems: (a) learning with a changing data distribution, and (b) only having access to a single chunk of data for learning at any point in time, unable to ever re-access previous chunks. We call this latter sub-problem the *chunking problem*. Current work in CL has focused on realistic settings where both sub-problems are present and where their separation has not been made explicit. This has meant that separating the two sub-problems contributions to the difficulty of CL and to what extent CL methods address each sub-problem has been largely unexplored. Therefore, in this work we investigate the chunking problem, looking at the extent to which it is a factor of CL performance and how well current CL methods deal with it. To do this we formulate and look at the *chunking setting* where we remove the task-shift element of CL but keep everything else the same. We do not propose this setting to be like the real world, where there is often task-shift, but to analyse the chunking sub-problem and its contribution to the more realistic full CL setting with task shift.

Our analysis of the chunking setting establishes a number of findings. First, we show that chunking is responsible for a significant part of the performance difference between CL and offline learning—learning with full access to all the data. Second, our experiments demonstrate that current CL methods do not address the chunking sub-problem, performing comparably to plain SGD training in the chunking setting. These two points suggest that chunking is a significant and unaddressed problem in CL. Additionally, we demonstrate that a large amount of forgetting—the loss of knowledge learnt from previously seen data—occurs in the chunking setting. This casts doubt on the common sentiment that forgetting is caused mainly by task shift (Lee et al., 2021; Ramasesh et al., 2020). Last, we demonstrate

[1]Code is available at https://github.com/Tlee43/Chunking-Setting

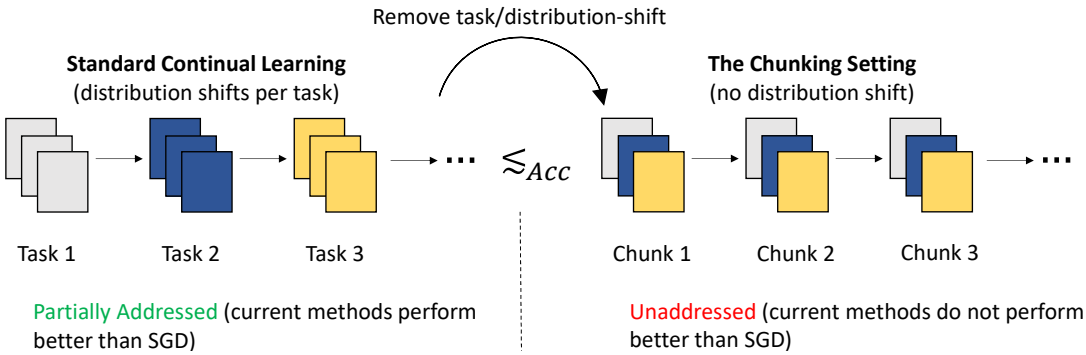

Figure 1: **Standard continual learning versus the chunking setting**. In standard continual learning (CL) a learner sequentially receives chunks of data called tasks and there is a shift in distribution between each task. While in the chunking setting, each chunk of data is identically distributed. CL methods have become better at dealing with task-shift and so partially address standard continual learning. However, we show that current CL methods do not tackle the chunking sub-problem as they perform no better than plain SGD training in the chunking setting. We also find that chunking contributes a significant part of the performance gap between offline learning and CL. This means that performance in the chunking setting is significantly lower than offline learning performance, for commonly used numbers of chunks. Therefore, as the chunking setting provides an approximate upperbound to performance in CL, improving capability in the chunking setting is a necessity to obtain high-performing CL methods.

that we can reduce forgetting and improve performance in the chunking setting, using a per-chunk weight averaging scheme. This performance improvement transfers to the full CL setting——where there is also task shift——establishing that work on the chunking sub-problem has the potential to impact CL in general.

The main contributions of this work are:

- Formulation of the *chunking sub-problem* of CL and demonstrating that it is the reason for a large part of the performance drop between offline learning and CL.
- Analysis of chunking, where we show among other things that current CL methods do not address this sub-problem, performing similarly to plain SGD training.
- Demonstrating that performance in the chunking setting can be improved and that this performance transfers to the full CL setting, illustrating how work on chunking can help improve CL in general.

2 Preliminaries and Related Work

Continual learning (CL) is a well-studied problem, with many different settings and methods being proposed (van de Ven & Tolias, 2019; Wu et al., 2022; Mirzadeh et al., 2020; Delange et al., 2021). We focus on classification problems. In this context, the standard CL setting (sometimes called offline CL (Prabhu et al., 2020)), consists of a learner seeing a sequence of tasks. Each *task* consists of a single chunk of data, where the data is drawn i.i.d. from a task-specific data distribution. In practice, this most commonly means that each task consists of all the training data from a subset of classes in the dataset (van de Ven & Tolias, 2019). A learner only views each task once and can only revisit data from previous tasks which it has stored in a limited memory buffer. For example, for CIFAR-10 (Krizhevsky, 2009) a learner might first see all the data for the airplane and ship classes, then see the data from the dog and cat classes and so on, seeing all data from two classes at a time until the learner has seen all the classes. Standard CL is further subdivided into different scenarios depending on the type of distribution shift which occurs and if the learner is told when the task changes or not. Four popular CL scenarios which are all refinements of the standard CL setting are task-incremental, class-incremental, domain-incremental and task-free learning (van de Ven & Tolias, 2019; Aljundi et al., 2018; Wang et al., 2023). Importantly, as these settings contain task-shift they are not the same as the chunking setting we look at, which is a sub-problem of all of them. Additionally, our results show the behaviour of CL methods in the chunking setting is quite different to their behaviour in each of the above standard CL scenarios. This is because, as shown in Figure 1, CL methods improve quite a bit upon plain SGD training in standard CL but perform similarly to it in the chunking setting. This suggest that current CL methods are quite good at dealing with the task-shift sub-problem but do not currently tackle the chunking problem, a key insight of our work.

The chunking sub-problem of CL is closely related to online learning, without task shift (Hoi et al., 2021; Bottou & LeCun, 2003). In both cases the data is observed in the form of a stationary data stream. However, in chunking the data is batched into chunks to match modern neural network learning processes. Straight online learning can be seen as a special case when each chunk consists of one data instance. Furthermore, we investigate the neural network case in contrast to much work in online learning which focuses on the linear case (Hoi et al., 2021). There is recent work on online learning of neural networks, for example Ash & Adams (2020); Caccia et al. (2022); and Sahoo et al. (2017). But, their focus is not on linking or comparing their work to CL and often the settings and assumptions are quite different from CL. Another related line of work to this paper is the study of the *new-instance* setting (Lomonaco & Maltoni, 2017; Prabhu et al., 2023). However, usually in this setting there is task shift (Lomonaco & Maltoni, 2017) and if not, to the best of our knowledge, previous work has not focused on how the setting is related to CL—with task shift. This is unlike this paper which focuses on providing insight into CL and where the chunking setting is deliberately constructed to examine the chunking sub-problem of CL. Also, in addition to the related work discussed in this section, we also discuss how chunking relates to positive transfer in Appendix C and it use in analysing the online CL setting in Appendix D. More generally, there are many areas in machine learning, like federated learning (Zhang et al., 2021), which have challenges similar to that of the chunking problem. Therefore, these areas might also benefit from work looking at chunking and vice versa.

The reason why the chunking sub-problem and its relation to CL has not been thoroughly explored in the CL literature is unclear. We see it as perhaps due to the fact that while the continual learning of neural networks has a long history (de Angulo & Torras, 1995; Polikar et al., 2001; Storkey, 1997; Grossberg, 1988), the current focus has been on realistic settings that contain a large amount of task shift (Delange et al., 2021; Wang et al., 2023). Because of this greater complexity, most work has focused on reducing the negative effects of task shift, leaving the component of performance due to chunking to be overlooked and implicit. Yet decomposing a problem can aid its solution, and indeed, we show that chunking is responsible for a large part of the performance drop from offline learning to CL.

To see if it is possible to improve performance in the chunking setting we consider per-chunk weight averaging and find that it provides significant improvements. There have been many weight averaging approaches proposed for offline learning (Izmailov et al., 2018; Tarvainen & Valpola, 2017). Additionally, there have been weight averaging methods suggested for CL (Lee et al., 2017; 2020; Garg et al., 2023; Stojanovski et al., 2022). So, we are not proposing our per-chunk weight averaging scheme as being a particularly original method. Instead, its use here is new and of interest in that it demonstrates that we can improve performance in the chunking setting and that this performance transfers to the full CL setting with task shift. To the best of our knowledge this has not been demonstrated before.

3 The Chunking Setting

In the *chunking setting*, a learner sees a sequence of chunks $C_1, C_2, \ldots, C_N$, and trains on one chunk of data at a time, where chunks are not revisited. Each chunk of data consists of instance pairs (x, y), with $x \in X$ (e.g. images) and labels $y \in Y$. The data in all chunks are drawn from the same distribution, so there is no distribution shift. Furthermore, in this paper, to control for class imbalance effects we consider a *balanced* chunking setting (henceforth assumed); we constrain each chunk to have as close to the same number of instances for each class in Y as possible. In this way we ensure the results of our experiments are solely due to the effects of limited data availability through chunking and not due to class imbalance. We record results for the case where the chunks are class imbalanced in Appendix H and observe that for our experimental setup class imbalance does not have any significant effect.

In practice, to perform experiments in the chunking setting, consider a class-balanced training dataset of size M and a target chunk size S. First, we randomly reorder the training data for each class, and then arrange all the data into a class-ordered list. Data is then sequentially allocated into $\lfloor M/S \rfloor$ chunks by assigning each element of the list in turn to a chunk, in a cyclical fashion. So, the first data item goes into chunk 1, second into chunk 2 etc., up to an item into chunk $\lfloor M/S \rfloor$, then the next into chunk 1 again and so on. Then we randomly permute the data within each chunk and randomly reorder the chunks themselves. To ensure chunks are fully balanced, in the experiments in this paper we choose chunk sizes so that all chunks are of equal size and contain the same number of data instances for each class. Finally, we reserve a equal-sized portion of data from each class to form a test set which is used to evaluate the accuracy of a method.

The only difference between the chunking setting and the full CL setting is the lack of task shift. Therefore, the chunking setting provides a simple way to analyse and understand the problems caused by chunking. Importantly, we are *not* proposing the chunking setting as being *realistic*, instead we use it to explore the chunking sub-problem which is a component of all the more realistic settings where there is also task-shift. Also, performance in the chunking setting gives an approximate upper bound to CL performance (task-shift just makes things harder), and so without solving this setting CL will never be able to improve beyond current chunking performance.

Table 1: Accuracy of DER++ when using a ResNet18 in the offline, chunking and standard CL class-incremental settings, along with the percentage drop in accuracy from offline learning to CL due to chunking (Chunking Prop.). We split each dataset into 10 tasks following the experimental setup of Buzzega et al. (2020) and Boschini et al. (2022). The table shows, by the bold column, that a significant proportion of the performance drop from offline learning to CL is due to the chunking problem.

Dataset	Offline	Chunking	CL	**Chunking Prop.**
CIFAR-100	$73.72_{\pm 0.115}$	$63.35_{\pm 0.348}$	$53.00_{\pm 0.327}$	$\mathbf{50.05\%}$
Tiny ImageNet	$60.63_{\pm 0.366}$	$50.54_{\pm 0.118}$	$39.02_{\pm 0.97}$	$\mathbf{46.69\%}$

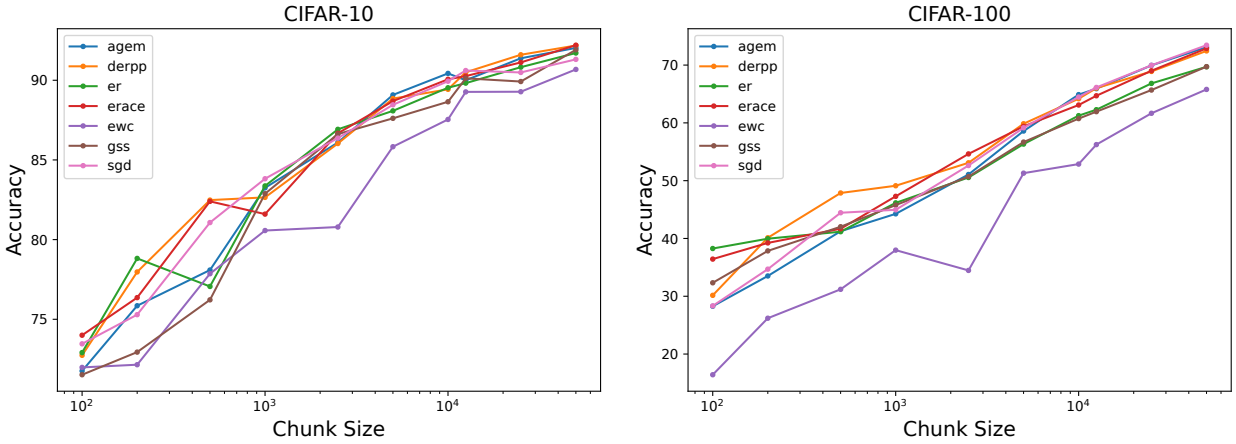

Figure 2: End-of-training accuracy against chunk size on CIFAR-10 and CIFAR-100. Each data point on a curve presents the end-of-training accuracy of a method from a full run with chunks of the size given on the horizontal axis. The plots show that a smaller chunk size leads to a greater performance drop from offline learning (the performance of the right most point in each plot) and that CL methods perform similarly to plain SGD training in the chunking setting.

4 Analysis of the Chunking Setting

To see how much chunking is a factor of the performance in CL, we perform an experiment with DER++ (Buzzega et al., 2020), a popular and highly capable CL method. We find that chunking plays a significant part in the performance drop from the offline setting. The experiment consists of comparing the relative performance drop from offline SGD training to DER++ on the standard CL and chunking settings. For the experiment, we use class-incremental learning for standard CL; which means at test time, the learner has to classify across all the classes (van de Ven & Tolias, 2019) in exactly the same way as the chunking setting. Additionally, following the experimental setup of Buzzega et al. (2020) and Boschini et al. (2022), we use a ResNet18 backbone and a 10 task/chunk split of (a) CIFAR-100 (Krizhevsky, 2009) with a memory size of 2000, and (b) Tiny ImageNet (Wu et al., 2015) with a memory size of 5120. The rest of the experimental details are given in Appendix B. The results are presented in Table 1 and show that the performance drop between offline learning and chunking is 50.05% and 46.69% of the full performance drop from offline learning to CL for CIFAR-100 and Tiny ImageNet, respectively. This indicates that a significant part of the performance drop of CL from offline learning is due to chunking and not due to task/distribution shift. Also, in the real world it is often the case that the hard task shifts commonly used in continual learning do not happen (Bang et al., 2021; 2022; Mi et al., 2020) and instead there are smoother changes between tasks which could reduce the effect of task shift and increase the importance of dealing with chunking.

4.1 Performance in the Chunking Setting

Our results on the chunking setting show that CL methods perform no better than plain SGD training. For instance, Figures 2 and 3 present the performance of state-of-the-art CL methods for different chunk sizes and a memory buffer size of 500 examples on CIFAR-10, CIFAR-100 and Tiny Imagenet, which are commonly used in the CL literature (Delange et al., 2021). We train on each chunk for 50 epochs for CIFAR-10 and CIFAR-100 and 100 epochs for Tiny

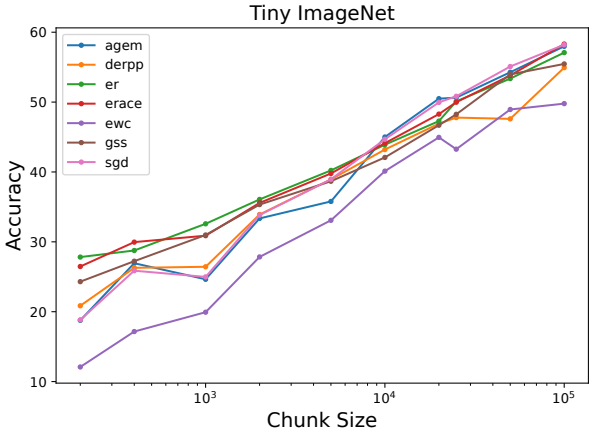

Figure 3: End-of-training accuracy against chunk size for Tiny ImageNet. Each data point on a curve presents the end-of-training accuracy of a method from a full run on Tiny ImageNet with chunks of the size given on the horizontal axis. The plot shows that the smaller the chunk size the greater the performance drop from offline learning (the performance of the right most point in the plot) and that CL methods perform similarly to plain SGD training in the chunking setting.

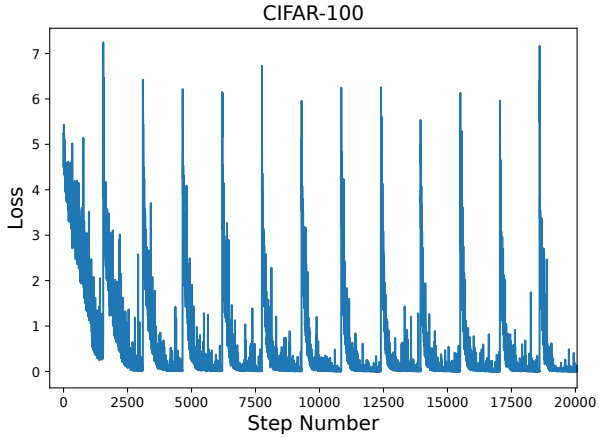

Figure 4: The training loss curve for plain SGD on CIFAR-100 when training on 50 chunks, where we plot the training loss for the first 2000 update steps corresponding to learning on the first 13 chunks. The plot shows that the loss converges for each chunk and hence that the learner does not underfit when training on any chunk.

ImageNet which we found to be give the best or comparable to the best performance (as shown in Appendix I). The full experimental details of this experiment are described in Appendix B. The results displayed in Figures 2 and 3 show that all the CL methods perform roughly the same as plain SGD training. Hence, our results indicate that current CL methods do not tackle the chunking problem and instead have focused on reducing the performance drop due to task shift, as they perform much better than SGD on settings with task shift (Wang et al., 2023). One point to note on this is that the replay methods ER (Chaudhry et al., 2020) and ER-ACE (Caccia et al., 2021) perform better than SGD for very small chunk sizes. This is due to them storing 500 examples in memory and so have an effective chunk size of 500 more data points than SGD, which impacts performance for chunk sizes around and below 500. Additionally, Figures 2 and 3 show that there is a large performance drop as the chunk size decreases. For example, on CIFAR-100 for offline learning when all the data is in one chunk, corresponding to a chunk size of 50000, CL methods get a test accuracy of around 73% but when each chunk consists of 1000 examples they get around 45%. Also, in addition to the results here, we show that the stability gap phenomenon (De Lange et al., 2023) appears in the chunking setting in Appendix E and look at the impact of using pretrained models in the chunking setting in Appendix G.

An important question to ask is why does chunking reduce the performance from offline learning. There are three general possibilities: not integrating all the information a chunk has into the model (underfitting), fully integrating each chunk's information but at the cost of forgetting previous information (forgetting) or a mixture of both. To explore which possibility is true we look in more detail at the case when we train using 50 chunks. We present the training loss curve in Figure 4 of learning on the first 13 chunks for CIFAR-100 and in Figure 5 the test accuracy and accuracy on the training data for the 5^{th}, 20^{th} and 40^{th} chunks evaluated at the end of each chunk, for CIFAR-100 and Tiny ImageNet (see Appendix J for CIFAR-10). The training loss curve in Figure 4 shows that we fit each chunk well as the loss plateaus for each chunk and at a relatively low value. Furthermore, the accuracy curves for each chunks training data in Figure 5 establishes that after training on the chunk the model fits it perfectly, achieving an accuracy of 100%. Hence, we know that the learner fits each chunk well, removing the possibility of underfitting. Figure 5 also shows that after learning on the chunk the accuracy on that chunks data quickly drops back to the level of the test set performance, showing that the learner is forgetting a lot of the chunk's information. Therefore, our results suggest that the performance drop in the chunking setting is due to forgetting. However, not all of a chunk's information is forgotten as the test accuracy improves as the learner sees more chunks. Additionally, our results indicate that forgetting is not only due to task shift, which is commonly assumed in previous work (Lee et al., 2021; Ramasesh et al., 2020), but that it is also due to chunking. It is also useful to question why forgetting is occurring. One potential reason is overfitting, as Figure 5 shows that the learner achieves 100% accuracy on the current chunk. However, whether this means overfitting is happening is a difficult question to answer as a model can achieve 100% training accuracy and still generalise well—as demonstrated by the double decent phenomenon (Nakkiran et al., 2021). If

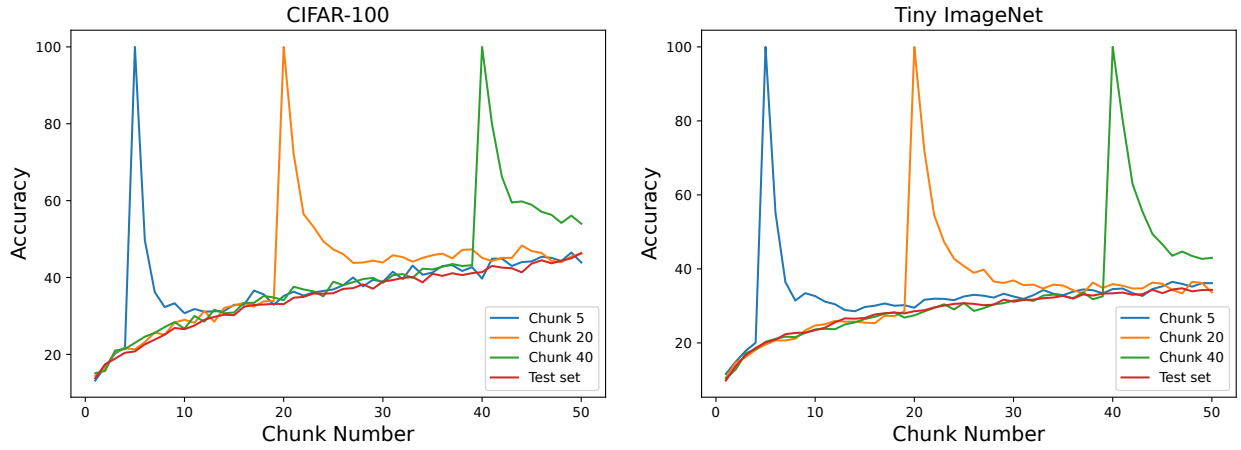

Figure 5: Accuracy at the end of learning on each chunk for the training set of the 5^{th}, 20^{th} and 40^{th} chunks and the test set, for CIFAR-100 and Tiny ImageNet when using plain SGD training. We split the datasets into 50 chunks, corresponding to a chunk size of 1000 and 2000 for CIFAR-100 and Tiny ImageNet, respectively. The plots show that after learning on a chunk the accuracy on that chunk quickly drops to the level of test set performance and hence that the learner quickly forgets a large part of the knowledge of a chunk after learning on it.

overfitting is the cause of the forgetting, it is interesting that even at the start of learning the learner forgets previous chunks. This is because, at the start of learning there is plenty of capacity in the network to both overfit to the current chunk and retain knowledge of previous chunks. So, if this is the case, the current way models are learnt seems to be particularly destructive of previous knowledge.

Our results suggest that forgetting is the reason for the reduced performance in the chunking setting, when compared to offline learning. However, not all the information provided by a chunk is forgotten and if a learner could repeatedly resample chunks it would approach offline performance (as suggested by Figure 5). This fact is standard knowledge for online learning (Bottou & LeCun, 2003) and has recently been shown to be true for CL with task-shift (Lesort et al., 2022). However, unlike standard online learning, due to real-world constraints in the chunking setting and CL it is not possible to resample chunks. Therefore, in these settings we need to be able to fully learn a chunk of data without needing to repeatedly revisit it in the future. This implies that improving chunking performance and reducing forgetting is closely related to improving the efficiency of learning. Hence, we hope that work on improving chunking performance will also improve the general efficiency of learning algorithms and vice versa.

5 Per-Chunk Weight Averaging

To explore if it is possible to improve performance in the chunking setting and whether this performance transfers to standard CL, we look at using per-chunk weight averaging. The reason we look at weight averaging is that by averaging over past weights the learner should better preserve information about previous chunks, reducing forgetting and hence improve performance. Also, in Appendix F we look at chunking when using linear models and show that weight averaging performs well. The simple weight averaging method we look at, *per-chunk weight averaging*, consists of training the model as normal but we additionally store an average of the weights learnt at the end of each chunk. The per-chunk weight average is not used in training but in evaluation is used as the weights of the network. Here we consider the *weights* to be all the parameters of the neural network, including batch normalisation statistics (Ioffe & Szegedy, 2015). More specifically, we look at using in evaluation the mean or an exponential moving average (EMA) of the weights found after training on each chunk up to some chunk k, defined by

$$\boldsymbol{\theta}_k^{MEAN} = \frac{1}{k}\sum_{t=1}^{k}\boldsymbol{\theta}_t \tag{1}$$

$$\boldsymbol{\theta}_k^{EMA} = \alpha\boldsymbol{\theta}_{k-1}^{EMA} + (1-\alpha)\boldsymbol{\theta}_k, \tag{2}$$

where $\boldsymbol{\theta}_t$ is the value of the weights after learning on chunk C_t and for EMA, $\alpha \in [0, 1]$ controls how much weight is given to old versus newly learnt end-of-chunk weights.

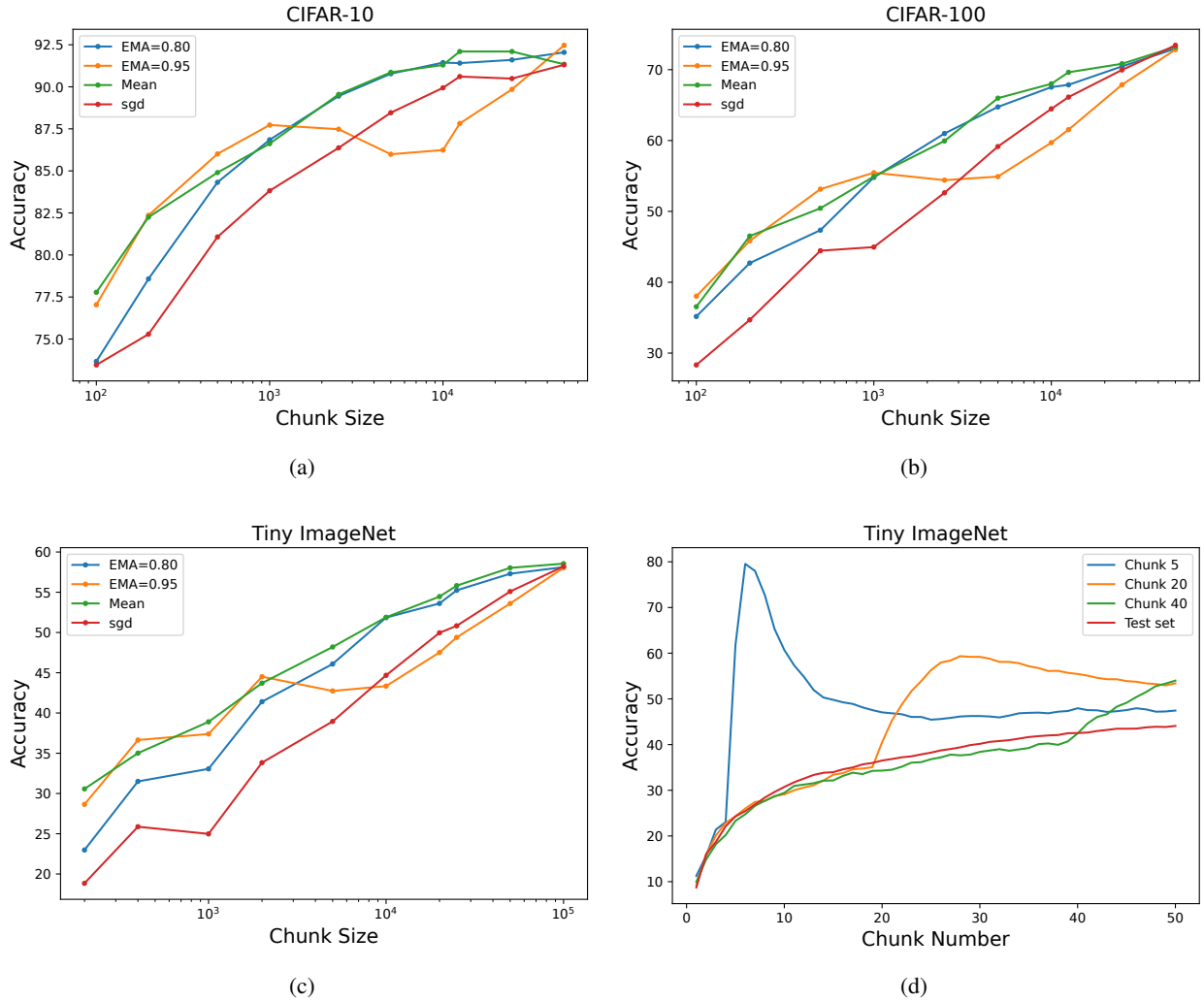

Figure 6: Plots (a), (b) and (c) show the end-of-training accuracy when learning with the given chunk size for CIFAR-10, CIFAR-100 and Tiny ImageNet, where *sgd* is learning without weight averaging and we display EMA results for α=0.8 and 0.95. These plots show that using weight averaging, in particular mean weight averaging, improves performance in the chunking setting. Plot (d) shows when using mean weight averaging the accuracy at the end of learning on each chunk for the training set of the 5^{th}, 20^{th} and 40^{th} chunks and the test set, for Tiny ImageNet with 50 chunks, corresponding to a chunk size of 2000. The plot demonstrates, when compared to Figure 5, that mean weight averaging forgets less than plain SGD training.

To observe whether per-chunk weight averaging improves performance in the chunking setting, we carry out experiments using it in combination with plain SGD training. The reason we only look at plain SGD training and not a CL method is that, as shown in Figures 2 and 3, no CL method looked at performs any better than SGD in the chunking setting. The experimental setup is the same as the previous experiments and is described in Appendix B. The results of the experiments are presented in plots (a), (b) and (c) of Figure 6 and show that it is clear that for all three datasets—CIFAR-10, CIFAR-100 and Tiny ImageNet—using a per-chunk weight average in evaluation increases accuracy. For instance, for the smallest chunk size looked at for each dataset, using mean weight averaging improves accuracy by $+4.32\%$, $+8.22\%$ and $+11.73\%$ for CIFAR-10, CIFAR-100 and Tiny ImageNet, respectively. Additionally, Figure 6 demonstrates that using the mean is better than or comparable to using EMA for nearly all chunk sizes on each dataset. We only display EMA for two α values in the figure but we looked at many more in Appendix K, and selected the two best values to show in Figure 6. So, our results show that using the mean of the weights learnt after learning on each chunk for prediction is an effective way to improve performance in the chunking setting.

Table 2: Accuracy of CL methods in online and standard CL settings when using per-chunk weight averaging (WA-) or not, averaged over 3 runs and where we report the standard error over the runs. We also present results for IMM a CL weight averaging method, which performs worse than using per-chunk weight averaging with any base method. The table indicates that in general using weight averaging improves performance as shown by the positive performance improvement (ΔAcc) when using weight averaging for almost all method/dataset/setting combinations.

		CIFAR-10		CIFAR-100		Tiny ImageNet	
Setting	Method	Class-IL	Task-IL	Class-IL	Task-IL	Class-IL	Task-IL
Online	DER++	$34.76_{\pm 2.20}$	$78.56_{\pm 1.10}$	$6.73_{\pm 0.26}$	$41.21_{\pm 1.34}$	$5.48_{\pm 0.21}$	$30.95_{\pm 0.11}$
	WA-DER++	$33.46_{\pm 0.72}$	$81.97_{\pm 0.25}$	$12.34_{\pm 0.19}$	$52.34_{\pm 0.43}$	$8.53_{\pm 0.05}$	$39.32_{\pm 0.55}$
	ΔAcc ($\uparrow$)	-1.30	$+3.41$	$+5.61$	$+11.13$	$+3.05$	$+8.37$
	ER	$36.19_{\pm 1.19}$	$81.89_{\pm 0.92}$	$8.45_{\pm 0.45}$	$44.14_{\pm 1.31}$	$5.56_{\pm 0.21}$	$27.23_{\pm 0.65}$
	WA-ER	$39.59_{\pm 0.60}$	$84.27_{\pm 0.37}$	$14.01_{\pm 0.23}$	$50.66_{\pm 0.77}$	$7.77_{\pm 0.09}$	$34.26_{\pm 0.33}$
	ΔAcc ($\uparrow$)	$+3.40$	$+2.38$	$+5.56$	$+6.52$	$+2.21$	$+7.03$
	AGEM	$16.82_{\pm 0.61}$	$70.70_{\pm 1.92}$	$4.70_{\pm 0.51}$	$29.56_{\pm 1.93}$	$3.93_{\pm 0.22}$	$20.53_{\pm 1.30}$
	WA-AGEM	$22.59_{\pm 1.04}$	$72.37_{\pm 3.03}$	$10.73_{\pm 0.35}$	$44.68_{\pm 0.58}$	$9.06_{\pm 0.47}$	$34.44_{\pm 0.59}$
	ΔAcc ($\uparrow$)	$+5.77$	$+1.67$	$+6.03$	$+20.39$	$+5.13$	$+13.91$
	GSS	$27.33_{\pm 1.26}$	$81.28_{\pm 1.47}$	$7.93_{\pm 0.16}$	$49.95_{\pm 0.24}$	$5.59_{\pm 0.11}$	$36.00_{\pm 0.49}$
	WA-GSS	$35.03_{\pm 0.50}$	$84.51_{\pm 0.41}$	$8.40_{\pm 0.32}$	$54.68_{\pm 0.28}$	$4.82_{\pm 0.06}$	$42.69_{\pm 0.38}$
	ΔAcc ($\uparrow$)	$+7.70$	$+3.23$	$+0.47$	$+4.73$	-0.77	$+6.69$
	IMM	$23.90_{\pm 1.40}$	$68.84_{\pm 0.77}$	$3.39_{\pm 0.29}$	$20.65_{\pm 0.55}$	$1.03_{\pm 0.08}$	$9.69_{\pm 0.29}$
Standard	DER++	$53.18_{\pm 0.87}$	$88.90_{\pm 0.30}$	$16.26_{\pm 1.22}$	$58.92_{\pm 0.36}$	$11.08_{\pm 0.38}$	$34.26_{\pm 0.32}$
	WA-DER++	$49.88_{\pm 1.63}$	$93.25_{\pm 0.33}$	$23.46_{\pm 1.48}$	$72.46_{\pm 1.08}$	$12.39_{\pm 0.93}$	$49.51_{\pm 0.69}$
	ΔAcc ($\uparrow$)	-3.30	$+4.35$	$+7.20$	$+13.54$	$+1.31$	$+15.25$
	ER	$40.01_{\pm 0.81}$	$89.79_{\pm 0.75}$	$11.78_{\pm 0.34}$	$57.80_{\pm 1.02}$	$8.36_{\pm 0.16}$	$31.72_{\pm 0.46}$
	WA-ER	$56.49_{\pm 0.87}$	$94.28_{\pm 0.17}$	$24.24_{\pm 0.64}$	$70.07_{\pm 0.29}$	$12.31_{\pm 0.19}$	$46.71_{\pm 0.33}$
	ΔAcc ($\uparrow$)	$+16.48$	$+4.49$	$+12.46$	$+12.27$	$+3.95$	$+14.99$
	AGEM	$20.19_{\pm 0.28}$	$85.80_{\pm 1.18}$	$9.35_{\pm 0.01}$	$46.99_{\pm 0.26}$	$8.15_{\pm 0.05}$	$24.76_{\pm 0.62}$
	WA-AGEM	$38.87_{\pm 2.83}$	$92.06_{\pm 0.61}$	$18.05_{\pm 0.68}$	$65.23_{\pm 0.61}$	$10.42_{\pm 0.32}$	$42.75_{\pm 0.25}$
	ΔAcc ($\uparrow$)	$+18.68$	$+6.26$	$+8.70$	$+18.24$	$+2.27$	$+17.99$
	GSS	$30.91_{\pm 1.02}$	$86.08_{\pm 0.35}$	$10.74_{\pm 0.10}$	$50.30_{\pm 0.28}$	$8.30_{\pm 0.01}$	$27.55_{\pm 1.04}$
	WA-GSS	$51.58_{\pm 1.14}$	$93.75_{\pm 0.43}$	$14.78_{\pm 0.57}$	$69.20_{\pm 0.35}$	$6.13_{\pm 0.07}$	$46.57_{\pm 1.16}$
	ΔAcc ($\uparrow$)	$+20.67$	$+7.67$	$+4.04$	$+18.90$	-2.17	$+19.02$
	IMM	$33.42_{\pm 1.97}$	$89.91_{\pm 1.19}$	$12.28_{\pm 1.33}$	$43.46_{\pm 2.00}$	$4.91_{\pm 0.58}$	$22.28_{\pm 0.73}$

To analyse why per-chunk weight averaging improves performance, we look at how well it preserves the information of past chunks. To do this, as in Figure 5, we measure, for per-chunk mean weight averaging, the test accuracy and the accuracy on the training data of the 5^{th}, 20^{th} and 40^{th} chunks at the end of learning on each chunk, when using 50 chunks. The results are shown in plot (d) of Figure 6 for Tiny ImageNet and for CIFAR-10 and CIFAR-100 in Appendix J. By comparing these results to the ones when using the final weights for evaluation, shown in Figure 5, we see that when using per-chunk mean weight averaging more information is preserved from previous chunks. This is because using it gives higher accuracy on the training data from previous chunks than the test set long after that chunk was trained on. While, when using the final weights for evaluation this is not the case, as after learning on a chunk the accuracy on the training data of that chunk drops quickly down to around the test set accuracy. This suggests that part of the reason per-chunk weight averaging performs well is that it forgets less than plain SGD training in the chunking setting.

5.1 Application to Continual Learning

While per-chunk weight averaging improves performance in the chunking setting, it is also important to see how this translates to the full CL setting, so that we can see how work on the chunking setting can impact CL in general. To do this we perform experiments using mean weight averaging in class and task incremental learning (van de Ven & Tolias, 2019), the two most common CL scenarios, using four standard well-performing methods: DER++ (Buzzega

et al., 2020), experience replay (ER) (Chaudhry et al., 2020), AGEM (Chaudhry et al., 2019) and GSS (Aljundi et al., 2019). The difference between class and task incremental learning is that at test time for task-incremental learning each method only predicts which class a data instance is between the classes of that data instance's task, while for class-incremental learning the method has to classify between all classes seen. As in common with the rest of this work and many works on continual learning (Delange et al., 2021; Buzzega et al., 2020), we use CIFAR-10, CIFAR-100 and Tiny ImageNet as the datasets for this experiment, splitting CIFAR-10 into 5 tasks each containing the data of 2 classes and splitting CIFAR-100 and Tiny ImageNet into 10 tasks each consisting of the data of 10 classes for CIFAR-100 and 20 for Tiny ImageNet. In addition to performing experiments using the standard CL setting (described in Section 2), we also present results for online CL (Mai et al., 2021). *Online CL* is the same as standard CL but the leaner sees the data for each task as a sequence of mini-batches each of which is used to update the learner only once and is not revisited. For standard CL, methods can repeatedly iterate over the data of a task, in our experiments for each task we use 50 epochs for CIFAR-10 and CIFAR-100 and 100 epochs for Tiny ImageNet, like previous work (Buzzega et al., 2020) and we set the memory size to be 100 examples for all experiments.

The results of the experiments on per-chunk mean weight averaging in CL are presented in Table 2 and demonstrate that in almost all cases it improves performance. For example, in the standard CL setting using per-chunk mean weight averaging improves performance on average by $+6.39\%$, $+11.11\%$, $+12.02\%$ and $+11.36\%$ for DER++, ER, AGEM and GSS, respectively. While in the online CL setting it improves performance on average by $+5.05\%$, $+4.52\%$, $+8.82\%$ and $+3.68\%$ for DER++, ER, AGEM and GSS, respectively. However, for class-incremental learning with DER++ on CIFAR-10 and GSS on Tiny ImageNet per-chunk mean weight averaging does worse than using the final learnt weights. But, as a method will have access to both options when using per-chunk mean weight averaging by validating the performance of each option it should be possible to pick the better one, avoiding any accuracy loss. For completeness, in Appendix L we also perform experiments with per-chunk EMA weight averaging in CL, showing that it performs worse than using the mean, like in the chunking setting. So, in summary, we have shown that per-chunk mean weight averaging improves performance in the chunking setting and that, in general, this improvement transfers to CL, showing that work on the chunking sub-problem can impact CL research as a whole.

We also perform experiments in standard and online CL with IMM (Lee et al., 2017) a popular weight averaging method for CL. IMM uses Fisher information to compute a weighted average of parameter weights. The results are presented in Table 2 and show that per-chunk weight averaging applied to any of the CL methods looked at outperforms IMM. So, while the main point of looking at per-chunk weight averaging is to demonstrate that performance can be improved in the chunking setting and that this performance transfers the standard CL setting; we also show that compared to IMM it is an effective weight averaging method for CL.

6 Conclusions

In this work we have looked at chunking, a sub-problem of continual learning (CL). We have presented results evidencing that it is responsible for a large part of the performance drop between offline and CL performance. Our results also reveal that current CL methods do not tackle the chunking problem, having comparable performance to plain SGD training in the chunking setting. Therefore, we have shown that chunking is a currently unaddressed problem which contributes a significant amount to the difficulty of CL. Additionally, we have demonstrated that a large amount of forgetting happens in the chunking setting, which casts into doubt the common belief that forgetting is caused mainly by task shift (Lee et al., 2021; Ramasesh et al., 2020). We also showed that performance on the chunking sub-problem can be improved—using per-chunk weight averaging. Furthermore, we demonstrated that this increased performance in the chunking setting transfers to the full CL setting, indicating that future work on chunking has the possibility of improving CL as a whole.

Acknowledgments

This work was kindly supported by ARM and EPSRC through an iCASE PhD scholarship.

References

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-Free Continual Learning. *arXiv preprint arXiv:1812.03596*, 2018.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient Based Sample Selection for Online Continual Learning. In *Proceedings of the 33rd Conference on the Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019.

Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining Benchmarks for Continual Few-shot Learning. *arXiv preprint arXiv:2004.11967*, 2020.

Jordan Ash and Ryan P Adams. On Warm-Starting Neural Network Training. In *Proceedings of the 34th Conference on the Advances in Neural Information Processing Systems*, pp. 3884–3894, 2020.

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow Memory: Continual Learning with a Memory of Diverse Samples. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8218–8227, 2021.

Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online Continual Learning on a Contaminated Data Stream with Blurry Task Boundaries. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9275–9284, 2022.

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-Incremental Continual Learning into the Extended DER-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5497–5512, 2022.

Léon Bottou and Yann LeCun. Large Scale Online Learning. In *Proceedings of the 17th Conference on the Advances in Neural Information Processing Systems*, 2003.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *Proceedings of the 34th Conference on the Advances in Neural Information Processing Systems*, pp. 15920–15930, 2020.

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 *Proceedings of the 10th International Conference on Learning Representations*, 2021.

Lucas Caccia, Jing Xu, Myle Ott, Marcaurelio Ranzato, and Ludovic Denoyer. On Anytime Learning at Macroscale. In *Proceedings of the 1st Conference on Lifelong Learning Agents*, pp. 165–182, 2022.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, and David Vázquez. Online Fast Adaptation and Knowledge Accumulation (OSAKA): A new Approach to Continual Learning. In *Proceedings of the 34th Conference on the Advances in Neural Information Processing Systems*, pp. 16532–16545, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual Learning in Low-rank Orthogonal Subspaces. In *Proceedings of the 34th Conference on the Advances in Neural Information Processing Systems*, pp. 9900–9911, 2020.

V.R. de Angulo and C. Torras. On-Line Learning with Minimal Degradation in Feedforward Networks. *IEEE Transactions on Neural Networks*, 6(3):657–668, 1995.

Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual Evaluation for Lifelong Learning: Identifying the Stability Gap. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Saurabh Garg, Mehrdad Farajtabar, Hadi Pouransari, Raviteja Vemulapalli, Sachin Mehta, Oncel Tuzel, Vaishaal Shankar, and Fartash Faghri. TIC-CLIP: Continual Training of CLIP Models. *arXiv preprint arXiv:2310.16226*, 2023.

Stephen Grossberg. Nonlinear Neural Networks: Principles, Mechanisms and Architectures. *Neural Networks*, 1(1): 17–61, 1988.

Tyler L Hayes and Christopher Kanan. Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 220–221, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online Learning: A Comprehensive Survey. *Neurocomputing*, 459:249–289, 2021.

Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. In *Proceedings of the 3rd Continual Learning Workshop, at the 32nd Conference on the Advances in Neural Information Processing Systems*, 2018.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging Weights leads to Wider Optima and better Generalization. *arXiv preprint arXiv:1803.05407*, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, and Agnieszka Grabska-Barwinska. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Preprint*, 2009.

Janghyeon Lee, Donggyu Joo, Hyeong Gwon Hong, and Junmo Kim. Residual Continual Learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 4553–4560, 2020.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Proceedings of the 31st Conference on the Advances in Neural Information Processing Systems*, 2017.

Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual Learning in the Teacher-Student Setup: Impact of Task Similarity. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6109–6119, 2021.

Thomas L Lee and Amos Storkey. Approximate Bayesian Class-Conditional Models under Continuous Representation Shift. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, pp. 3628–3636, 2024.

Timothée Lesort, Oleksiy Ostapenko, Diganta Misra, Md Rifat Arefin, Pau Rodríguez, Laurent Charlin, and Irina Rish. Challenging common assumptions about catastrophic forgetting. *arXiv preprint arXiv:2207.04543*, 2022.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Beyond not-Forgetting: Continual Learning with Backward Knowledge Transfer. In *Procceding of the 36th Conference on the Advances in Neural Information Processing Systems*, pp. 16165–16177, 2022.

Vincenzo Lomonaco and Davide Maltoni. Core50: A New Dataset and Benchmark for Continuous Object Recognition. In *Proceedings of the 1st Conference on Robot Learning*, pp. 17–26, 2017.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online Continual Learning in Image Classification: An Empirical Survey. *arXiv preprint arXiv:2101.10423*, 2021.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained Visual Classification of Aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Fei Mi, Lingjing Kong, Tao Lin, Kaicheng Yu, and Boi Faltings. Generalized Class Incremental Learning. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 240–241, 2020.

Thomas Minka. Bayesian Linear Regression. Technical report, Microsoft Research, 2000.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the Role of Training Regimes in Continual Learning. In *Proceedings of the 34th Conference on the Advances in Neural Information Processing Systems*, pp. 7308–7320, 2020.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep Double Descent: Where Bigger Models and More Data Hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12): 124003, 2021.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Foundational Models for Continual Learning: An Empirical Study of Latent Replay. *arXiv preprint arXiv:2205.00329*, 2022.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A review. *Neural Networks*, 113:54 – 71, 2019.

Francesco Pelosin. Simpler is Better: off-the-shelf Continual Learning Through Pretrained Backbones. *arXiv preprint arXiv:2205.01586*, 2022.

R. Polikar, L. Upda, S.S. Upda, and V. Honavar. Learn++: An Incremental Learning Algorithm for Supervised Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. GDumb: A Simple Approach that Questions our Progress in Continual Learning. In *Procceding of the 16th European Conference on Computer Vision*, pp. 524–540, 2020.

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 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3698–3707, 2023.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of Catastrophic Forgetting: Hidden Representations and Task Semantics. *arXiv preprint arXiv:2007.07400*, 2020.

Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online Deep Learning: Learning Deep Neural Networks on the Fly. *arXiv preprint arXiv:1711.03705*, 2017.

Zafir Stojanovski, Karsten Roth, and Zeynep Akata. Momentum-Based Weight Interpolation of Strong Zero-Shot Models for Continual Learning. *arXiv preprint arXiv:2211.03186*, 2022.

Amos Storkey. Increasing the Capacity of a Hopfield Network Without Sacrificing Functionality. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, pp. 451–456, 1997.

Antti Tarvainen and Harri Valpola. Mean Teachers are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results. *Proceedings of the 31st Conference on the Advances in Neural Information Processing Systems*, 2017.

Naftali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. In *Proceedings of the 2015 IEEE information theory workshop*, pp. 1–5. IEEE, 2015.

Gido M van de Ven and Andreas S Tolias. Three Scenarios for Continual Learning. *arXiv preprint arXiv:1904.07734*, 2019.

Martin J Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. 2019.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *arXiv preprint arXiv:2302.00487*, 2023.

Ross Wightman, Hugo Touvron, and Herve Jegou. ResNet Strikes Back: An Improved Training Procedure in timm. In *Proccedings of the Workshop on ImageNet: Past, Present, and Future, at the 35th Conference on the Advances in Neural Information Processing Systems*, 2021.

Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny Imagenet Challenge (cs231n), http://tiny-imagenet.herokuapp.com/. Technical report, Stanford, 2015.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. Pretrained Language Models in Continual Learning: A Comparative Study. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A Survey on Federated Learning. *Knowledge-Based Systems*, 216:106775, 2021.

A Limitations

While we have aimed to be thorough in our analysis of the chunking problem, like any work there are limitations to what is presented in the paper. We list a number of limitations here. First, this work is mainly empirically based so it would be interesting to see what theoretical approaches can tell us about the chunking sub-problem and its relation CL. For example, the performance in the chunking setting is an approximate upper-bound to CL performance, as it a reduced and simpler setting due to removing the task-shift element. However, in this work we did not make the nature of the upper-bounding exact and so it would be useful to see if a theorem could be derived to formally express how chunking performance upper-bounds CL performance. Additionally, it would be interesting to see a theoretical approach to decomposing the difficulty of CL into its chunking and task-shift components. Another limitation is that while mean weight averaging does improve performance in the chunking setting, there is still a large gap between it and offline learning performance, for a small enough chunk size. This means there is space for better methods to be developed which better solve the chunking problem and hence hopefully also the full CL setting with task shift. Last, we only look at supervised CL in this work however the chunking problem is also a sub-problem of any self or semi-supervised CL setting. Therefore it would be interesting to see how much chunking contributes the difficulty of those settings and whether CL methods developed for those settings currently tackle the chunking problem.

B Experimental Details

For all of our results we follow the experimental protocol of Buzzega et al. (2020) and Boschini et al. (2022), and use a modification of the CL library *Mammoth* used in those works to run the experiments. Therefore, for all our experiments we use a ResNet18 (He et al., 2016) as the backbone model. Additionally, we utilize augmentations, applying random crops and horizontal flips to images trained on for all the datasets used. To be able to have a fair comparison in our chunking experiments all methods are trained using SGD and with the same number of epochs: 50 epochs for each chunk for CIFAR-10 and CIFAR-100 and 100 for Tiny ImageNet. We use the same mini-batch size for all experiments, which is 32 examples, and for replay CL methods we use 32 as the replay batch size as well. The hyperparameters of methods were found using a grid search on a validation set and are the same as in Buzzega et al. (2020) and Boschini et al. (2022); however, all the results were newly computed by the authors for this work. For all results on the chunking setting a learning rate of 0.1 was used to ensure a fair comparison between methods and chunk sizes. Last, the full list of CL methods evaluated in the chunking setting is: AGEM (Chaudhry et al., 2019), DER++ (Buzzega et al., 2020), ER (Chaudhry et al., 2020), ER-ACE (Caccia et al., 2021), EWC (Kirkpatrick et al., 2017), GSS (Aljundi et al., 2019) and plain SGD training. Plain SGD training is when the neural network is trained with SGD using the standard cross entropy loss, as in previous work (Buzzega et al., 2020; Boschini et al., 2022).

C Positive Transfer and the Chunking Problem

While a large part of the effort in CL has thus far been to reduce forgetting, another key problem is improve the positive transfer capabilities of methods (Lin et al., 2022). *Positive transfer* is the ability of a learner to use the data in the currently accessible chunk/task to to improve its knowledge of previous chunks/tasks and future chunks/tasks to be seen. It has been shown that current CL methods do not perform positive transfer well and that improving positive transfer capabilities will improve the performance of CL algorithms in general (Lee & Storkey, 2024; Lin et al., 2022). While positive transfer has previously been explored between chunks/tasks of differing data distributions, our results on the chunking setting show that current CL methods fail to perform positive transfer well even if the chunks are all sampled from the same distribution. This is because, Figures 2 and 3 show that CL methods perform comparable to plain SGD training in the chunking setting and that Figure 5 shows that plain SGD training is very bad at positive transfer, as it looses instead of gains performance on past chunks when training on additional chunks. Therefore, we propose that improving performance in the chunking setting should lead to methods that are better at positive transfer and vice versa. Additionally, as the chunking setting is much simpler than other CL settings, being a sub-problem of all of them, hopefully by looking at this setting its simplicity will be a benefit when working on improving the positive transfer capabilities of methods.

D The Chunking Problem in Online CL

In Section 4 we show that the chunking problem is a significant challenge in standard CL and here we describe why it is also a challenge for online CL. Unlike this work and past work in standard CL, most work in online CL compare not to offline performance but to an i.i.d upperbound. The *i.i.d upperbound* is plain SGD training run in the online CL setting but all the mini-batches are drawn i.i.d without replacement from the dataset. Therefore, this upperbound

computes the chunking performance of plain SGD for a chunk size of one mini-batch and where a method only takes a single update on each chunk (to the best of our knowledge this has not been explicitly discussed in the online CL literature and the chunking problem has remained unexamined until this work). Interestingly, online CL methods are now getting comparable performance to this upperbound (Caccia et al., 2021; Lee & Storkey, 2024). This suggests that most of the remaining performance to be gained in online CL is in solving the chunking problem. Therefore, there is a large space in the literature to solve the chunking problem and in doing so improve online CL performance beyond this current i.i.d upperbound performance. In this paper we give the first steps towards this by showing we can improve on the i.i.d uppperbound performance by using per-chunk weight averaging (see Figure 6 and Table 2).

E THE STABILITY GAP IN THE CHUNKING SETTING

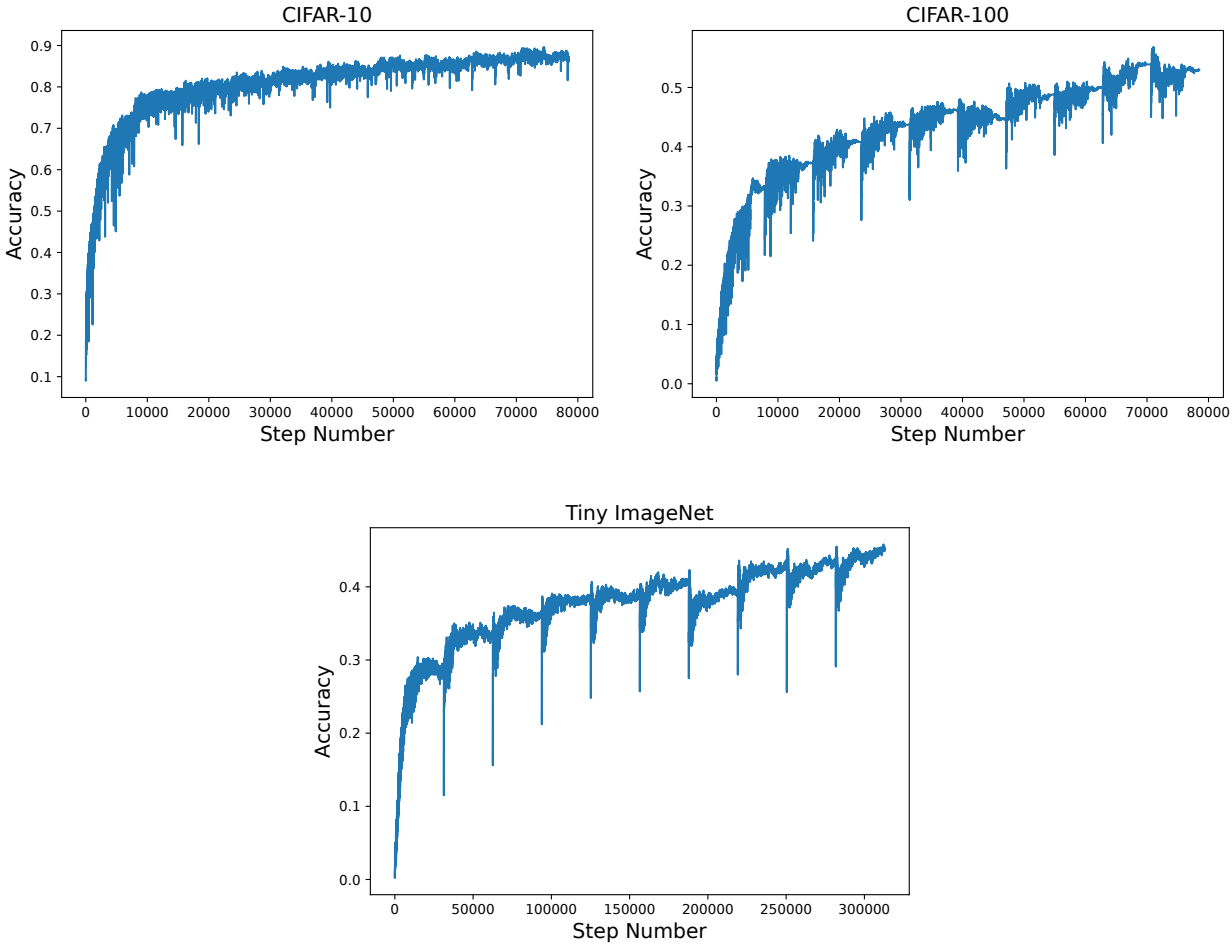

Figure 7: Test accuracy curves for plain SGD training using 10 chunks on CIFAR-10, CIFAR-100 and Tiny ImageNet. Step number refers to the number of parameter updates which have been performed up to that point. The plots, apart from CIFAR-10, show that at the start of each chunk there is a large drop in test accuracy.

The stability gap is a recently discovered phenomena in standard CL (De Lange et al., 2023), we show in this appendix that it also occurs in the chunking setting. The *stability gap* is the phenomena that at the start of learning a new task the accuracy for previous tasks drops quickly by a significant amount and then recovers back to a stable level. This stable level is somewhat lower than the accuracy value before seeing the new task. The reason this is thought to happen (De Lange et al., 2023; Caccia et al., 2021) is that at the start of learning on a new task the performance on it will be poor, leading to large gradients and changes to the weights of the network. This potentially induces a drop in performance on previous tasks as the large weight updates remove information about previous tasks. Finally, as the weights change the performance on data stored in memory from previous tasks will reduce, increasingly impacting the

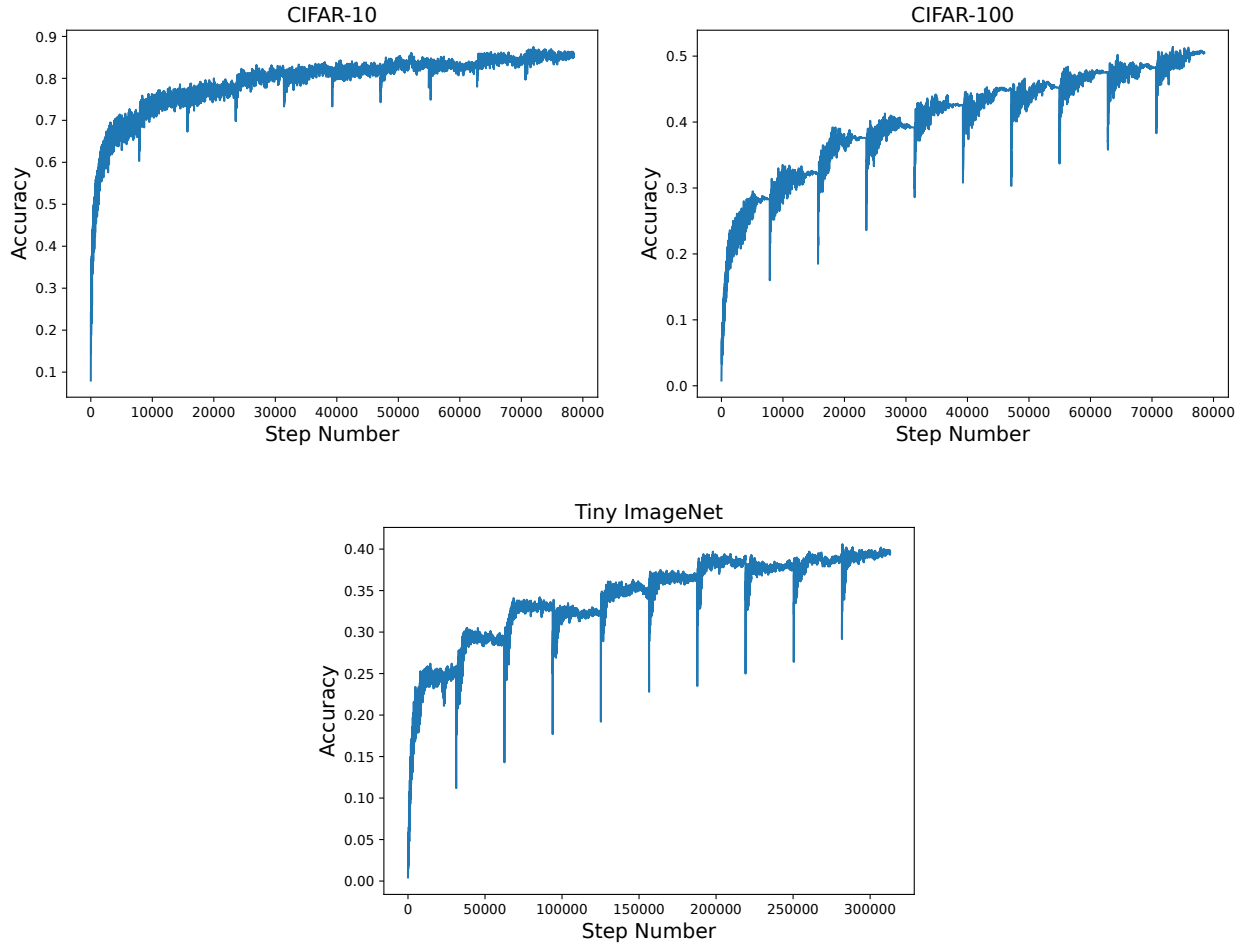

Figure 8: Test accuracy curves for ER using 10 chunks on CIFAR-10, CIFAR-100 and Tiny ImageNet. Step number refers to the number of parameter updates which have been performed up to that point. The plots show that at the start of each chunk there is a large drop in test accuracy.

parameter updates to perform better on previous tasks, recovering a large amount of the performance lost on previous tasks. Given the interest in this phenomena, it is useful to explore if it occurs in the chunking setting and so see if it is only because of task shift or not.

We present experiments on the stability gap in the chunking setting in Figures 7 and 8. The experiments consist of recording the performance on a held-out set of data for each parameter update step when learning on 10 chunks and using plain SGD training or ER. We use the same experimental setup as the rest of our experiments and these experiments are analogous to the ones performed by De Lange et al. (2023) but for the chunking setting instead of standard CL. The results show that the stability gap also occurs in the chunking setting as all bar one of the plots show sudden drops in test performance when the learner starts learning on a new chunk. The plot which does not show sudden drops is for plain SGD learning on CIFAR-10, where we believe the general noise in performance hides the systematic drops in performance at the start of each chunk. So, our results show that the stability gap phenomenon is not only to do with distribution shift. Furthermore, our experiments add light to why the stability gap occurs, where we can rule out distribution shift as being a necessary factor. Instead, these experiments indicate in conjunction with Figure 4—which shows the training loss for plain SGD training—that there is some form of ”overfitting” or compression (Tishby & Zaslavsky, 2015) occurring which induces high losses at the start of learning a chunk. This in turn leads to large update steps. Finally, these large updates potentially makes the network forget previously learnt information, creating the drop in performance at the start of a chunk we see in Figures 7 and 8. Given that the stability gap occurs in the chunking setting a potential good direction to solve it would be to explore it in the setting. This is because, the chunking setting is simpler and easier to reason about than standard CL.

F Analysis of the Linear Case

To theoretically analyse the chunking problem we turn to the linear regression case, where we can leverage closed form solutions. In this case, the naive solution is to perform least squares on each arriving chunk. However, as the least squares problem is convex and so does not depend on the initialised weights, it will fully forget all the past chunks, only using the last chunk to create the predictor. This means that the standard least squares solution to linear regression fails in the chunking setting. Instead a better solution is to use Bayesian linear regression (Minka, 2000). This is because Bayesian linear regression given any particular chunking of the data will return the same predictor and so fully solves the chunking setting. Therefore, it is instructive to see how Bayesian linear regression prevents forgetting. To achieve this we present below the update equations for Bayesian linear regression. The prior on the weights is $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_0)$ and the posterior after seeing all the chunks up to and including the $(k-1)$th is $\boldsymbol{\theta}|C_{1:k-1} \sim \mathcal{N}(\mathbf{m}_{k-1}, \mathbf{V}_{k-1})$. Additionally, for a chunk C_t we define $\mathbf{X}_t$ as its row-wise matrix of data instances and $\mathbf{y}_t$ as its vector of targets. The likelihood is defined by assuming $y|\mathbf{x}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}^T\mathbf{x}, \sigma^2)$. Then, the Bayesian posterior for the kth chunk is

$$\boldsymbol{\theta}|C_{1:k} \sim \mathcal{N}(\mathbf{m}_k, \mathbf{V}_k), \tag{3}$$

$$\mathbf{m}_k = \mathbf{V}_k\mathbf{V}_{k-1}^{-1}\mathbf{m}_{k-1} + \frac{1}{\sigma^2}\mathbf{V}_k\mathbf{X}_k^T\mathbf{y}_k, \tag{4}$$

$$\mathbf{V}_k^{-1} = \mathbf{V}_k^{-1} + \frac{1}{\sigma^2}\mathbf{X}_k^T\mathbf{X}_k. \tag{5}$$

By recursively expanding the $\mathbf{m}_{k-1}$ and $\mathbf{V}_{k-1}$ terms till we reach the prior we have that

$$\mathbf{m}_k = \frac{1}{\sigma^2}\sum_{t=1}^{k}\mathbf{V}_k\mathbf{X}_t^T\mathbf{y}_t, \tag{6}$$

$$\mathbf{V}_k^{-1} = \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\sum_{t=1}^{k}\mathbf{X}_t^T\mathbf{X}_t = \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}_{1:k}^T\mathbf{X}_{1:k}. \tag{7}$$

The equations above show that Bayesian linear regression prevents forgetting by having its posterior mean $\mathbf{m}_k$ be: (a) a sum of the least squares solutions of each chunk and (b) instead of using the chunks unnormalised empirical covariance $\mathbf{X}_t^T\mathbf{X}_t$ in the least squares solutions it uses the running estimate of the weight precision $\mathbf{V}_k^{-1}$. Computing and storing $\mathbf{V}_k^{-1}$ is infeasibly costly for very large systems (e.g. neural networks), taking up $O(\dim(\boldsymbol{\theta})^2)$ space. Therefore, assuming there is only enough memory to store a set of weights a backoff is to use a sum of the least squares solutions to each chunk. This is achieved by *weight averaging*, where at each chunk we perform least squares on that chunk and add it to a running average, which results in the update equation,

$$\mathbf{m}_k = \frac{k-1}{k}\mathbf{m}_{k-1} + \frac{1}{k}(\mathbf{X}_k^T\mathbf{X}_k)^{-1}\mathbf{X}_k^T\mathbf{y}_k. \tag{8}$$

Again, by recursively expanding $\mathbf{m}_{k-1}$ we have that,

$$\mathbf{m}_k = \frac{1}{k}\sum_{t=1}^{k}(\mathbf{X}_t^T\mathbf{X}_t)^{-1}\mathbf{X}_t^T\mathbf{y}_t. \tag{9}$$

Weight averaging gives similar, mean, weights as Bayesian linear regression where instead of using $\mathbf{V}_k$ it uses the per-chunk estimate $\frac{1}{k}(\mathbf{X}_t^T\mathbf{X}_t)^{-1}$ and we divide by k to correctly scale the estimate. Both $\mathbf{V}_k$ and $\frac{1}{k}(\mathbf{X}_t^T\mathbf{X}_t)^{-1}$ are unnormalised estimates of the precision of the data distribution. Therefore, when each chunk is large enough that they are both accurate estimates, we have that $\frac{1}{k}(\mathbf{X}_t^T\mathbf{X}_t)^{-1} \approx \mathbf{V}_k$ for all $t \in \{1, \ldots, k\}$. In this case, weight averaging approximates Bayesian linear regression well and so should not forget that much. More formally, by using a concentration bound on the covariance estimates, we have the following theorem on the approximation error.[2]

Theorem 1 *(proved in Appendix M) Assume that we have k chunks and that each chunk $C_t = \{\mathbf{x}_i \in \mathbb{R}^d | i = 1, ..., S\}$ is sampled i.i.d. from an α-sub-Gaussian distribution (assuming zero mean) with a full rank covariance matrix $\boldsymbol{\Sigma}$. Also, assume bounded random variables such that $\|\mathbf{x}_i\|_2 \leq a_{\mathbf{x}}$ and $\|\mathbf{y}_t\|_2 \leq a_{\mathbf{y}}$. Then, for the Bayesian linear regression model set the prior such that $\mathbf{V}_0 = b\mathbf{I}$ and let $b \to \infty$. Last, denote $\mathbf{m}_{BLR}$ and $\mathbf{m}_{WA}$ as the parameter*

[2] We focus on the goodness of the approximation in the size of each chunk and use simple techniques to give such a bound. With more complicated techniques, it should also be possible to examine the behaviour in the number of chunks, were we believe there should be some convergence to a fixed approximation error (a.s.).

estimates given by Bayesian linear regression and weight averaging, respectively. We then have for universal constants a_1, a_2, a_3 *and for* δ *in the range* $\alpha^{-2}\lambda_d(\mathbf{\Sigma}) > \delta \geq 0$ *the following approximation bound,*

$$\|\mathbf{m}_{BLR} - \mathbf{m}_{WA}\|_2 \leq \frac{2a_{\mathbf{x}}a_{\mathbf{y}}}{\sqrt{S}} \frac{\epsilon(S,\delta)}{(\lambda_d(\mathbf{\Sigma}) - \epsilon(S,\delta))^2}$$

with probability of at least $1 - ka_2e^{-a_3 S \min(\delta,\delta^2)}$. *Defining* $\lambda_d(\mathbf{\Sigma})$ *as the smallest eigenvalue of* $\mathbf{\Sigma}$,

$$\epsilon(S,\delta) = \alpha^2 \left[a_1 \left(\sqrt{\frac{d}{S}} + \frac{d}{S} \right) + \delta \right]$$

and assuming

$$S \geq \frac{\alpha^2 a_1 \left[\alpha^2 a_1 + 2(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta) + \alpha\sqrt{a_1(\alpha^2 a_1 + 4(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta))} \right]}{2(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta)^2} d. \tag{10}$$

Theorem 1 shows formally that the larger the chunk size S the better weight averaging approximates Bayesian linear regression and that in the limit weight averaging converges to Bayesian linear regression (a.s.).

The analysis in this section shows that weight averaging is a reasonable method for the linear case. It greatly improves performance over standard linear regression, which forgets all but the last chunk, and approximates well Bayesian linear regression which fully solves continual learning. Hence, the question arises if this analysis showing weight averaging improves performance also holds true for neural networks. We show in the main paper that empirically this is true (e.g. see Figure 6) but, to the best of our knowledge, it is an open problem to show it theoretically as well.

G The Effect of Pretraining in the Chunking Setting

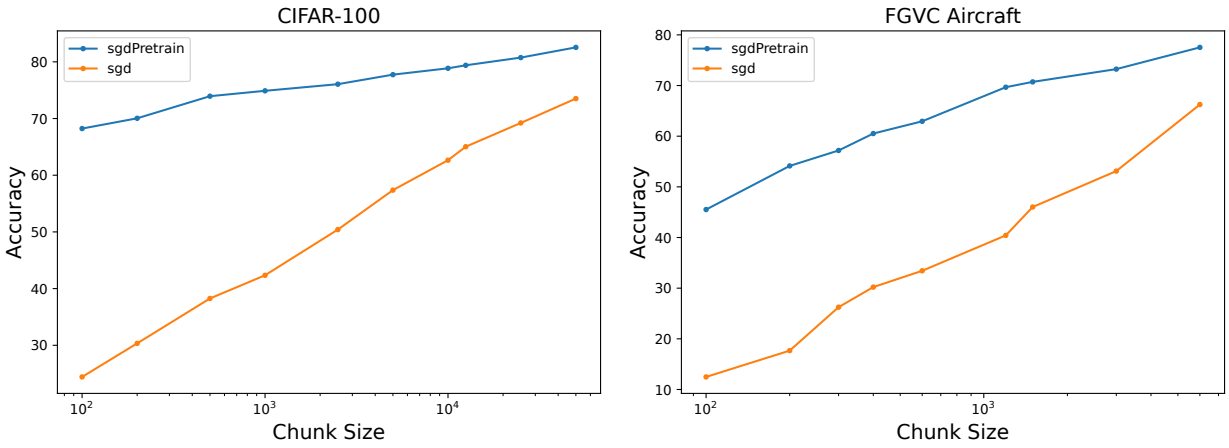

Figure 9: End-of-training accuracy against chunk size on CIFAR-100 and FGVC Aircraft when finetuning a pretrained ResNet18 (sgdPretrain) or training from scratch (sgd), where the pretrained network is pretrained on ImageNet. Each data point on a curve presents the end-of-training accuracy of a method from a full run with chunks of the size given on the horizontal axis. The plots show that using a pretrained network reduces the performance gap from offline learning (the performance of the right most point for each curve) to chunking performance. But, for FGVC Aircraft which is more dissimilar to the pretraining dataset, the narrowing of the performance gap is less.

While pretraining is often not considered in the CL literature it can greatly reduce the performance drop of CL methods compared to offline training (Hayes & Kanan, 2020; Ostapenko et al., 2022; Pelosin, 2022). Therefore, we present here how much pretraining aids in solving the chunking setting. In Figure 9 we record the performance of using a ResNet18 model pretrained on ImageNet (Wightman et al., 2021) compared from learning from scratch, using plain SGD training, for a dataset which is similar to ImageNet, CIFAR-100, and one which is not as similar, FGVC Aircraft (Maji et al., 2013). The reason why we consider FGVC Aircraft to be not as similar to ImageNet as CIFAR-100 is

that CIFAR-100 shares classes with ImageNet while FGVC Aircraft contains classes of different aircraft and so, given that ImageNet only has one aircraft class, FGVC Aircraft a is more fine-grained classification dataset. Figure 9 shows that for CIFAR-100 pretraining greatly reduces the performance drop from offline learning to the chunking setting. For example, when using the pretrained network, there is only a difference of around 10% between seeing the CIFAR-100 as one chunk (offline performance) and seeing chunks of 100 examples each. However, for FGVC Aircraft, while pretraining improves performance, there is still a significant performance gap to offline learning, around a 30% accuracy drop, for small chunk sizes. Therefore, we have shown that pretraining helps solve the chunking problem but for data streams which are dissimilar to the pretraining dataset chunking is still a significant problem.

H Analysis of Stratified Sampling for Chunks

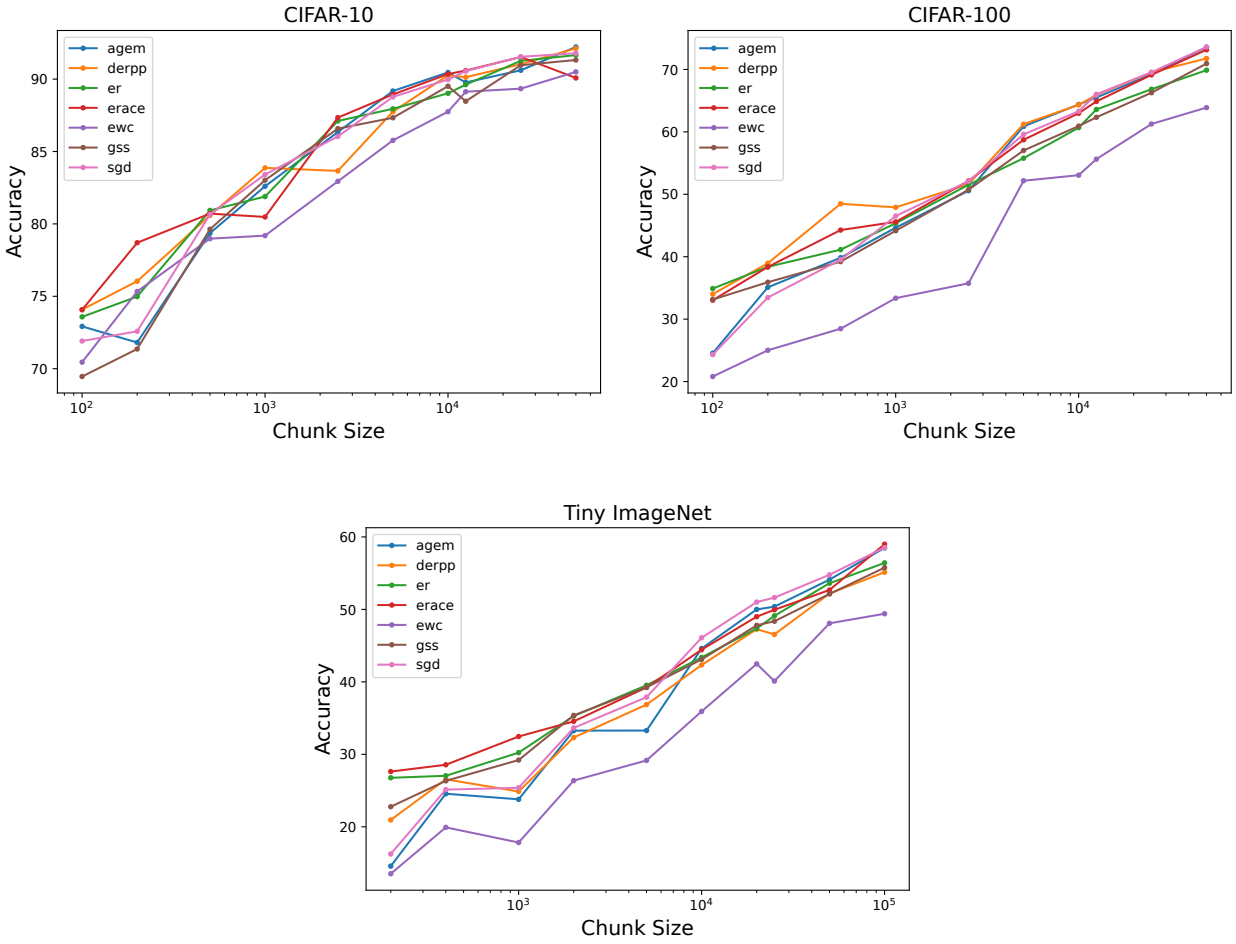

Figure 10: Plots of end-of-training accuracy against chunk size on CIFAR-10, CIFAR-100 and Tiny ImageNet, where the data sampled for each chunk is not constrained to be class balanced. Each data point on a curve presents the end-of-training accuracy of a method from a full run with chunks of the size given on the horizontal axis. The plots show that sampling each chunk without ensuring they are class balanced gives the same trend as when the chunks are class balanced, which are displayed in Figures 2 and 3.

To make sure for small chunk sizes the drop in performance is due to the online availability of data and not class imbalance in a chunk, in the (balanced) chunking setting we stratify sample chunks such that each chunk has an equal amount of data from each class. However, in Figure 10 we look at what happens if the chunks were sampled randomly without ensuring the classes are balanced in each chunk (i.e., we randomly split the dataset into the given number of chunks). The figure shows the same trend as the when the chunks are classed balanced, displayed in Figures 2 and 3, indicating that class imbalance does not affect our findings. However, class imbalance could be a problem for other

datasets/objectives and is for the linear case therefore we assume the chunks are class balanced in the rest of this work and in our formulation of the chunking setting.

I Analysis of Number of Epochs per Chunk

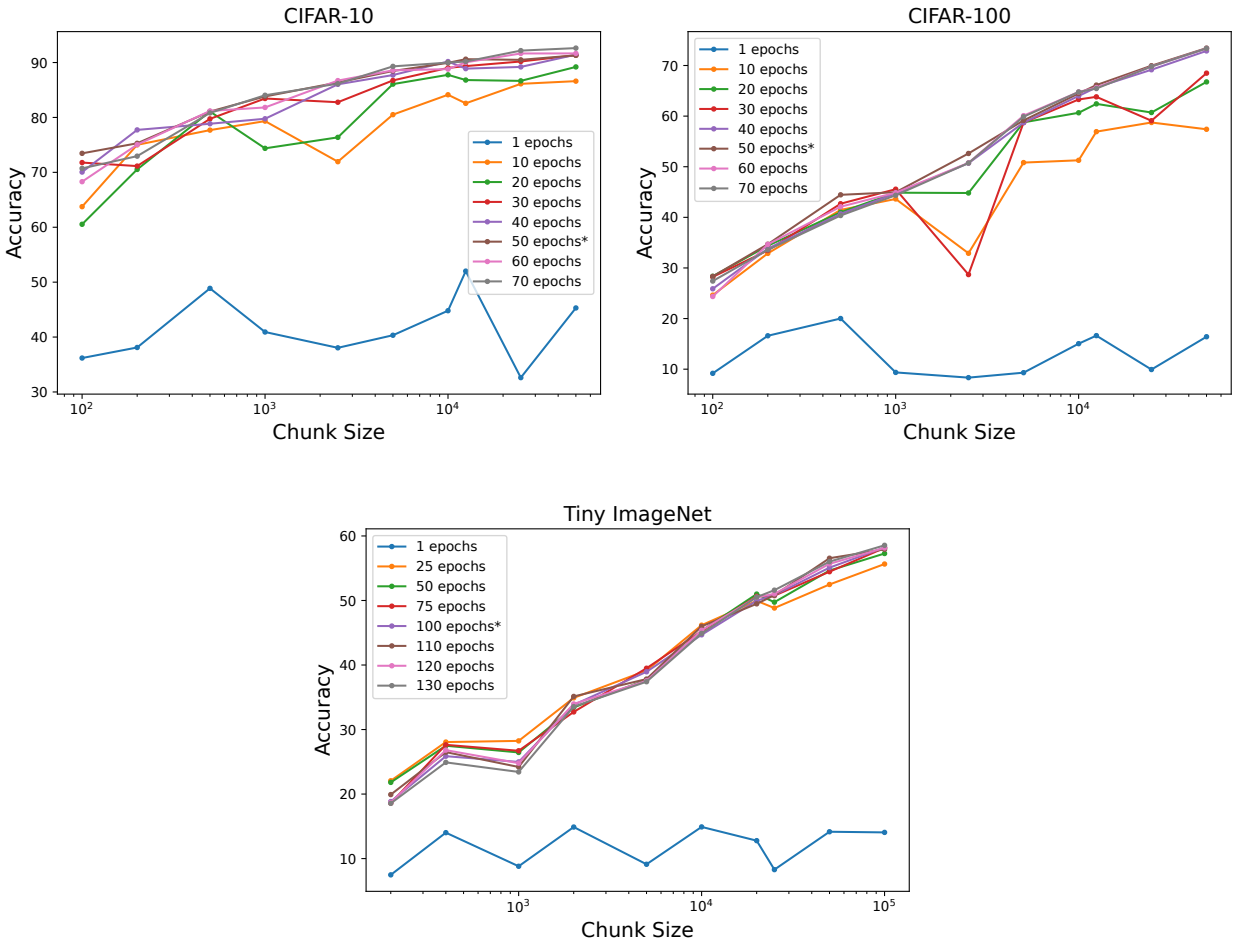

Figure 11: Plots of end-of-training accuracy against chunk size on CIFAR-10, CIFAR-100 and Tiny ImageNet for SGD using different number of epochs per chunk. Each data point on a curve presents the end-of-training accuracy of a method from a full run with chunks of the size given on the horizontal axis. The '*' denotes the number of epochs we use in the rest of the experiments and the figure shows that using our selected values achieves the best, or very similar to the best, accuracy for each chunk size.

Hyperparameters can affect the reason why a method performs badly in the chunking setting. For example, if a method trains for one epoch over each chunk then it probably has bad performance due to underfiting; while, if it is trained for a reasonable number of epochs the reason would be forgetting, as shown in the main paper (e.g. Figure 5). To remove this dependence we report the behaviour when using the hyperparameters which achieve the best accuracy for each chunk size. In Figure 11 we report the effect the number of training epochs used for each chunk has on performance, showing our selected number of epochs is the best or comparable to the best for each chunk size. Interestingly, the best performing number-of-epochs is the same for the full CL setting with task shift (Buzzega et al., 2020; Boschini et al., 2022). This suggests that when fitting the training hyperparameters in CL, we are in part implicitly fitting them to minimise the effect of chunking.

J ADDITIONAL FORGETTING CURVES

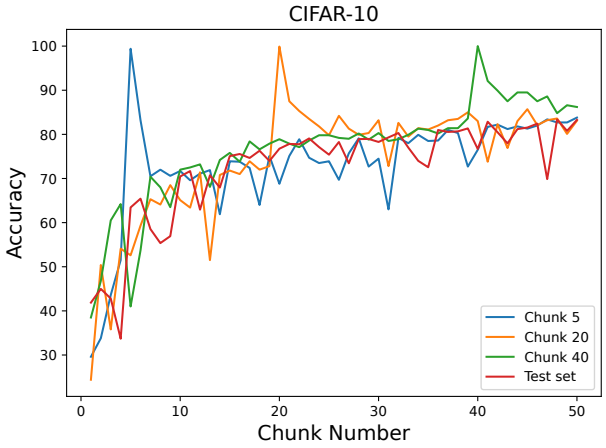

Figure 12: Plot for plain SGD training of accuracy at the end of learning on each chunk for the training set of the 5^{th}, 20^{th} and 40^{th} chunks and the test set, when training on CIFAR-10 with 50 chunks, corresponding to a chunk size of 1000. The plot shows that after learning on a chunk the accuracy on that chunk quickly drops to the level of test set performance and hence that the learner quickly forgets a large part of the knowledge of the chunk after learning on it.

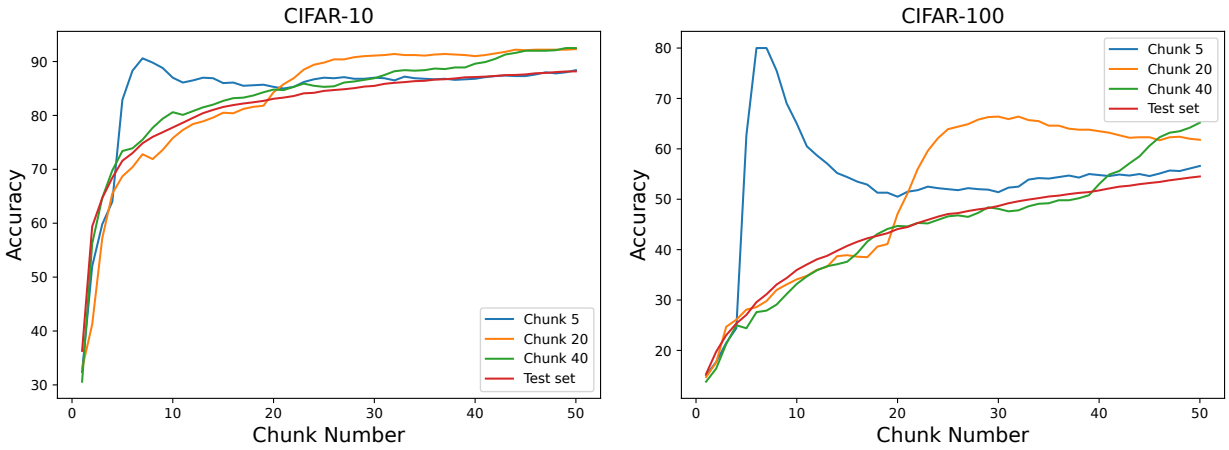

Figure 13: Plots showing when using mean weight averaging the accuracy at the end of learning on each chunk for the training set of the 5^{th}, 20^{th} and 40^{th} chunks and the test set, when training on CIFAR-10 and CIFAR-100 with 50 chunks, corresponding to a chunk size of 1000 for both datasets. The plots demonstrate that mean weight averaging forgets less than plain SGD training, whose analogous plots are displayed in Figures 5 and 12.

K EXPERIMENT ON DIFFERENT WEIGHTINGS FOR EMA

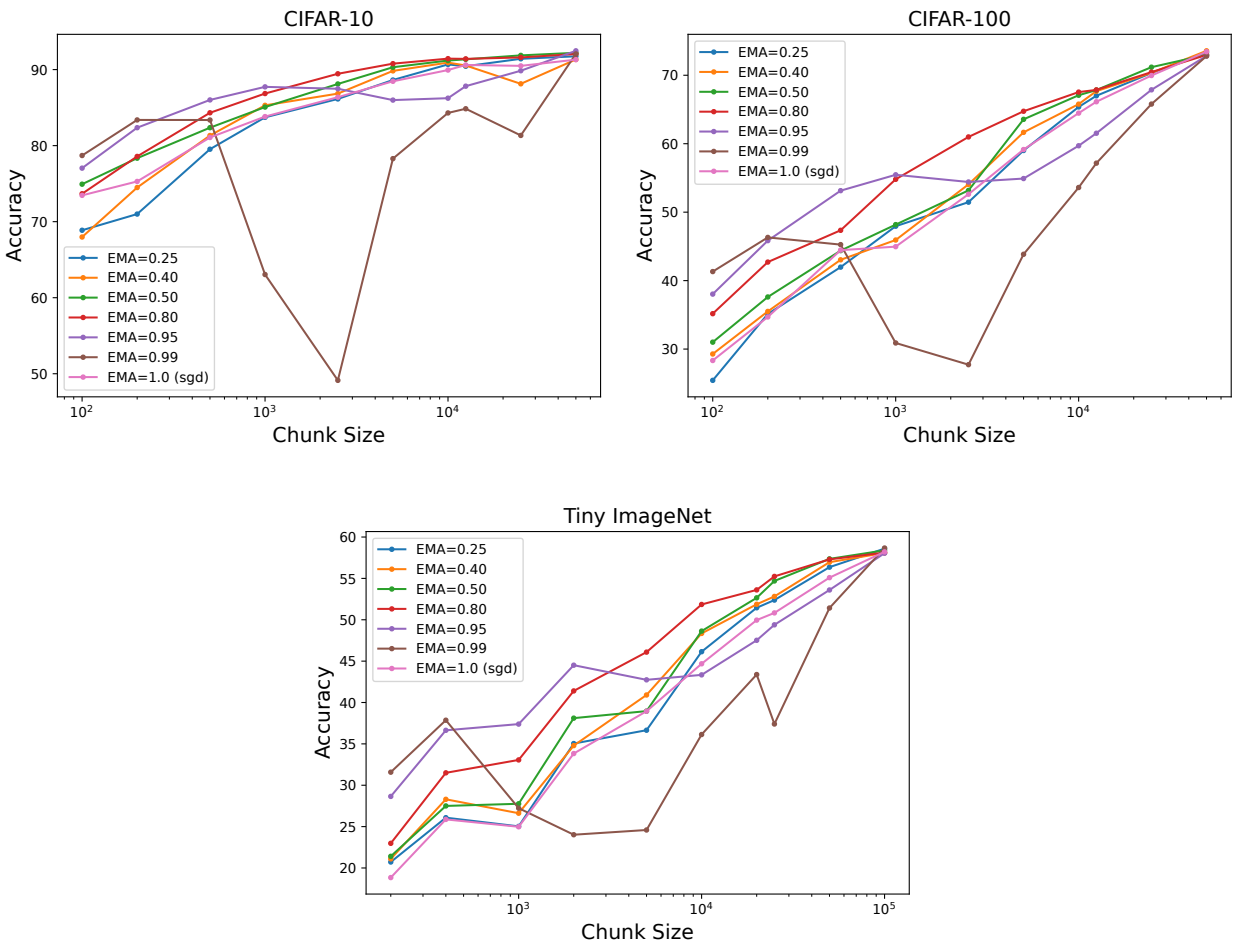

Figure 14: Plots of the end-of-training accuracy when leaning with the given chunk size for different EMA weight values when learning with SGD on CIFAR-10, CIFAR-100 and tiny ImageNet. The plots shows that the EMA values of 0.80 and 0.95 achieve consistently the best or comparable to the best accuracy for all chunk sizes and datasets.

L EXPERIMENT ON THE PERFORMANCE OF PER-CHUNK EMA IN STANDARD CL

Table 3: Accuracy of ER in online and standard CL settings when using per-chunk mean averaging (WA-), per-chunk EMA (EMA-) or neither. The accuracies are averaged over 3 runs and where we report the standard error over the runs. The results show that using per-chunk mean averaging performs better than using per-chunk EMA.

		CIFAR-10		CIFAR-100		Tiny ImageNet	
Setting	Method	Class-IL	Task-IL	Class-IL	Task-IL	Class-IL	Task-IL
Online	ER	$36.19_{\pm 1.19}$	$81.89_{\pm 0.92}$	$8.45_{\pm 0.45}$	$44.14_{\pm 1.31}$	$5.56_{\pm 0.21}$	$27.23_{\pm 0.65}$
	WA-ER	$39.59_{\pm 0.60}$	$84.27_{\pm 0.37}$	$14.01_{\pm 0.23}$	$50.66_{\pm 0.77}$	$7.77_{\pm 0.09}$	$34.26_{\pm 0.33}$
	EMA-ER	$31.92_{\pm 0.45}$	$79.22_{\pm 0.94}$	$8.65_{\pm 0.66}$	$41.72_{\pm 0.88}$	$5.8_{\pm 0.13}$	$28.24_{\pm 0.28}$
Standard	ER	$40.01_{\pm 0.81}$	$89.79_{\pm 0.75}$	$11.78_{\pm 0.34}$	$57.80_{\pm 1.02}$	$8.36_{\pm 0.16}$	$31.72_{\pm 0.46}$
	WA-ER	$56.49_{\pm 0.87}$	$94.28_{\pm 0.17}$	$24.24_{\pm 0.64}$	$70.07_{\pm 0.29}$	$12.31_{\pm 0.19}$	$46.71_{\pm 0.33}$
	EMA-ER	$38.84_{\pm 0.25}$	$89.82_{\pm 0.18}$	$11.68_{\pm 0.08}$	$58.16_{\pm 0.90}$	$8.20_{\pm 0.14}$	$33.76_{\pm 1.72}$

In the main paper we only look at per-chunk mean weight averaging in the CL experiments, here we also report results using per-chunk EMA weight averaging (Table 3). The reason we only looked at per-chunk mean weight averaging in the main paper was that we showed it had better performance in the chunking setting. The results in Table 3 shows that in both online and standard CL it is still the case that per-chunk mean weight averaging performs better. This shows a correspondence between the performances in the chunking and CL settings. One point to note is that we use weighting of 0.8 for EMA in these experiments.

M PROOF OF THEOREM 1

In this section we prove a bound on how well weight averaging approximates Bayesian linear regression. To do this we first state the concentration bound used in the proof, where $||.||_2$ for a matrix denotes the l_2-operator norm.

Theorem 2 (rephrased from Wainwright (2019) (Theorem 6.5 p.166)) *Given a chunk $C_t = \{\mathbf{x}_i \in \mathbb{R}^d | i = 1, ..., S\}$ sampled i.i.d. from a α-sub-Gaussian distribution (assuming zero mean) with covariance matrix $\mathbf{\Sigma}$; there are universal constants a_1, a_2, a_3 such that the empirical covariance matrix $\frac{1}{S}\mathbf{X}_t^T\mathbf{X}_t$ satisfies the bound,*

$$P\left(\frac{1}{\alpha^2}\left\|\frac{1}{S}\mathbf{X}_t^T\mathbf{X}_t - \mathbf{\Sigma}\right\|_2 \leq a_1\left[\sqrt{\frac{d}{S}} + \frac{d}{S}\right] + \delta\right) \geq 1 - a_2 e^{-a_3 S \min(\delta, \delta^2)}$$

for all $\delta \geq 0$.

To allow us to complete the proof we need some assumptions which are given here. We assume that we see k chunks and that each chunk $C_t = \{\mathbf{x}_{t,i} \in \mathbb{R}^d | i = 1, ..., S\}$ is sampled i.i.d. from an α-sub-Gaussian distribution (assuming zero mean) with a full rank covariance matrix $\mathbf{\Sigma}$. We also assume bounded random variables such that $\|\mathbf{x}_i\|_2 \leq a_{\mathbf{x}}$ and $\|\mathbf{y}_t\|_2 \leq a_{\mathbf{y}}$. Additionally, for the Bayesian linear regression model we use a prior such that $\mathbf{V}_0 = b\mathbf{I}$ and let $b \to \infty$. To make the proof easier to follow we denote the running and per-chunk covariances as $\hat{\mathbf{\Sigma}}_{1:k} = \frac{1}{NS}\mathbf{X}_{1:k}^T\mathbf{X}_{1:k}$ and $\hat{\mathbf{\Sigma}}_t = \frac{1}{S}\mathbf{X}_t^T\mathbf{X}_t$, respectively. Furthermore, we denote the projected targets as $\mathbf{y}'_t = \mathbf{X}_t^T\mathbf{y}_t$. Also, let $\mathbf{m}_{\text{BLR}}$ and $\mathbf{m}_{\text{WA}}$ be the parameter estimates given by Bayesian linear regression and weight averaging, respectively, and note to calculate such estimates we assume that the empirical precision matrices exists. In the proof we need a bound on the $\mathbf{y}'_t$ for each chunk which is,

$$\begin{aligned}
\|\mathbf{y}'_t\|_2 &= \|\mathbf{X}_t^T\mathbf{y}_t\|_2 && (11)\\
&\leq \|\mathbf{y}_t\|_2 \max_{\mathbf{v}, \|\mathbf{v}\|_2=1} (\|\mathbf{X}_t^T\mathbf{v}\|_2) && (12)\\
&= \|\mathbf{y}_t\|_2 \|\mathbf{X}_t^T\|_2 && (13)\\
&\leq \|\mathbf{y}_t\|_2 \|\mathbf{X}_t^T\|_F && (14)\\
&\leq a_{\mathbf{x}} a_{\mathbf{y}} \sqrt{S}. && (15)
\end{aligned}$$

Additionally, we need to be able to bound $\|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2$ and $\|\hat{\mathbf{\Sigma}}_t^{-1}\|_2$. This is achieved using Weyl's inequality, where we have for $\lambda_d(\hat{\mathbf{\Sigma}}_t)$ and $\lambda_d(\mathbf{\Sigma})$, the smallest eigenvalues for $\hat{\mathbf{\Sigma}}_t$ and $\mathbf{\Sigma}$, respectively, that

$$\lambda_d(\hat{\mathbf{\Sigma}}_t) \geq \lambda_d(\mathbf{\Sigma}) - \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2. \tag{16}$$

Now, if we were to use Theorem 2 we can ensure the r.h.s. of Eq. 16 is greater or equal to zero, by using a large enough chunk size S. This will be done in the proof and therefore we will have the following bound on $\|\hat{\mathbf{\Sigma}}_t^{-1}\|_2$,

$$\|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 = \frac{1}{\lambda_d(\hat{\mathbf{\Sigma}}_t)} \leq \frac{1}{\lambda_d(\mathbf{\Sigma}) - \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2}. \tag{17}$$

We have an equivalent bound for $\|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2$ using the same argument and using the fact that $\hat{\mathbf{\Sigma}}_{1:k}$ is the mean of the chunk covariances $\hat{\mathbf{\Sigma}}_t$.

Given the above ground work, the proof proceeds as follows,

$$\|\mathbf{m}_{\text{BLR}} - \mathbf{m}_{\text{WA}}\|_2 = \left\| \sum_{t=0}^{k} \frac{1}{kS} \hat{\mathbf{\Sigma}}_{1:k}^{-1} \mathbf{y}'_t - \sum_{t=0}^{k} \frac{1}{kS} \hat{\mathbf{\Sigma}}_t^{-1} \mathbf{y}'_t \right\|_2 \quad \text{(by simplifying from Eqs. 9 and 6)} \tag{18}$$

$$\leq \frac{1}{kS} \sum_{t=0}^{k} \|\hat{\mathbf{\Sigma}}_{1:k}^{-1} \mathbf{y}'_t - \hat{\mathbf{\Sigma}}_t^{-1} \mathbf{y}'_t\|_2 \tag{19}$$

$$= \frac{1}{kS} \sum_{t=0}^{k} \|[\hat{\mathbf{\Sigma}}_{1:k}^{-1} - \hat{\mathbf{\Sigma}}_t^{-1}] \mathbf{y}'_t\|_2 \tag{20}$$

$$\leq \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \max_{\mathbf{v}, \|\mathbf{v}\|_2 = 1} (\|[\hat{\mathbf{\Sigma}}_{1:k}^{-1} - \hat{\mathbf{\Sigma}}_t^{-1}] \mathbf{v}\|_2) \tag{21}$$

$$= \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \|\hat{\mathbf{\Sigma}}_{1:k}^{-1} - \hat{\mathbf{\Sigma}}_t^{-1}\|_2 \quad \text{(by variational def. of the operator norm)} \tag{22}$$

$$= \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \|\hat{\mathbf{\Sigma}}_{1:k}^{-1} (\hat{\mathbf{\Sigma}}_{1:k} - \hat{\mathbf{\Sigma}}_t) \hat{\mathbf{\Sigma}}_t^{-1}\|_2 \tag{23}$$

$$\leq \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_{1:k} - \hat{\mathbf{\Sigma}}_t\|_2 \tag{24}$$

$$= \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 \|(\hat{\mathbf{\Sigma}}_{1:k} - \mathbf{\Sigma}) - (\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma})\|_2 \tag{25}$$

$$\leq \frac{1}{kS} \sum_{t=0}^{k} \|\mathbf{y}'_t\|_2 \|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 [\|\hat{\mathbf{\Sigma}}_{1:k} - \mathbf{\Sigma}\|_2 + \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2] \tag{26}$$

$$\leq \frac{a_{\mathbf{x}} a_{\mathbf{y}}}{k\sqrt{S}} \sum_{t=0}^{k} \|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 [\|\hat{\mathbf{\Sigma}}_{1:k} - \mathbf{\Sigma}\|_2 + \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2] \quad \text{(using Eq. 11)} \tag{27}$$

$$\leq \frac{2a_{\mathbf{x}} a_{\mathbf{y}}}{k\sqrt{S}} \sum_{t=0}^{k} \|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2 \quad \text{(as } \hat{\mathbf{\Sigma}}_{1:k} \text{ is the mean of the chunk covs. } \hat{\mathbf{\Sigma}}_t) \tag{28}$$

$$\tag{29}$$

Now by using Theorem 2 to bound each $\|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2$, the union bound, and Eq. 17 applied to $\|\hat{\mathbf{\Sigma}}_{1:k}^{-1}\|_2$ and each $\|\hat{\mathbf{\Sigma}}_t^{-1}\|_2$, we have that,

$$\|\mathbf{m}_{\text{BLR}} - \mathbf{m}_{\text{WA}}\|_2 \leq \frac{2a_{\mathbf{x}} a_{\mathbf{y}}}{\sqrt{S}} \frac{\epsilon(S, \delta)}{(\lambda_d(\mathbf{\Sigma}) - \epsilon(S, \delta))^2} \tag{30}$$

for δ in the range $\alpha^{-2}\lambda_d(\mathbf{\Sigma}) > \delta \geq 0$ with probability of at least $1 - ka_2 e^{-a_3 S \min(\delta, \delta^2)}$, where

$$\epsilon(S, \delta) = \alpha^2 \left[a_1 \left(\sqrt{\frac{d}{S}} + \frac{d}{S} \right) + \delta \right] \tag{31}$$

and

$$S \geq \frac{\alpha^2 a_1 \left[\alpha^2 a_1 + 2(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta) + \alpha\sqrt{a_1(\alpha^2 a_1 + 4(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta))} \right]}{2(\lambda_d(\mathbf{\Sigma}) - \alpha^2\delta)^2} d. \tag{32}$$

This completes the proof. We note that by using the union bound we remove the convergence behaviour in the number of chunks. A way to solve this problem would be to bound the mean over $\|\hat{\mathbf{\Sigma}}_t^{-1}\|_2 \|\hat{\mathbf{\Sigma}}_t - \mathbf{\Sigma}\|_2$ using more complex methods.