OVERCOMING THE STABILITY GAP IN CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In many real-world applications, deep neural networks are retrained from scratch as a dataset grows in size. Given the computational expense for retraining networks, it has been argued that continual learning could make updating networks more efficient. An obstacle to achieving this goal is the stability gap, which refers to an observation that when updating on new data, performance on previously learned data degrades before recovering. Addressing this problem would enable learning new data with fewer network updates, resulting in increased computational efficiency. We study how to mitigate the stability gap. We test a variety of hypotheses to understand why the stability gap occurs. This leads us to discover a method that vastly reduces this gap. In large-scale class incremental learning experiments, we are able to significantly reduce the number of network updates needed for continual learning. Our work has the potential to advance the state-of-the-art in continual learning for real-world applications along with reducing the carbon footprint required to maintain updated neural networks.

1 INTRODUCTION

Deep learning is computationally expensive and has an extraordinary carbon footprint (Schwartz et al., 2020). In industrial settings, typically these models are periodically re-trained from scratch as datasets grow in size (Mallick et al., 2022). One avenue toward more efficiently updating networks is continual learning (CL), where the goal is to incrementally train a model to accumulate data in which the model is updated with the new data (Parisi et al., 2019). While most of the CL research community has focused on mitigating catastrophic forgetting (McCloskey & Cohen, 1989), there is a growing body of literature demonstrating its ability to reduce the amount of compute needed to update the network (Ghunaim et al., 2023; Harun et al., 2023a;b; Prabhu et al., 2023). Recently, a major obstacle to this objective has been identified in CL: *the stability gap* (De Lange et al., 2023). As illustrated in Fig. 1, the stability gap refers to the observation that when learning new data, performance on old data decreases dramatically before recovering with additional network updates. This phenomenon has been observed for a variety of CL techniques, including rehearsal (i.e., replay) (French, 1999), which is a widely successful method that involves mixing a portion of old data with new data (Kemker & Kanan, 2018; Hayes et al., 2020; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018b; Rebuffi et al., 2017). Mitigating the stability gap would enable fewer updates to the network providing significant computational improvements during training. In this paper, we seek to understand why the stability gap occurs through a series of hypothesis driven experiments, which results in the development of a new method that requires far fewer network updates during CL.

Catastrophic forgetting occurs when incrementally training from a non-stationary distribution, so CL researchers have focused on scenarios where catastrophic forgetting is greatest, such as class incremental learning (CIL) (Kemker et al., 2018). Typically, in this experimental setting samples arrive in batches where each batch contains classes that are only within that batch. When the model is updated on the new data, accuracy on the classes observed in earlier batches briefly plummets, before gradually recovering when mitigation techniques such as rehearsal are used. The stability gap occurs as each new batch is learned. De Lange et al. (2023) demonstrated the stability gap was reduced when tasks were more similar, which is also when catastrophic forgetting happens to a lesser degree; however, they did not propose methods for mitigating this phenomenon.

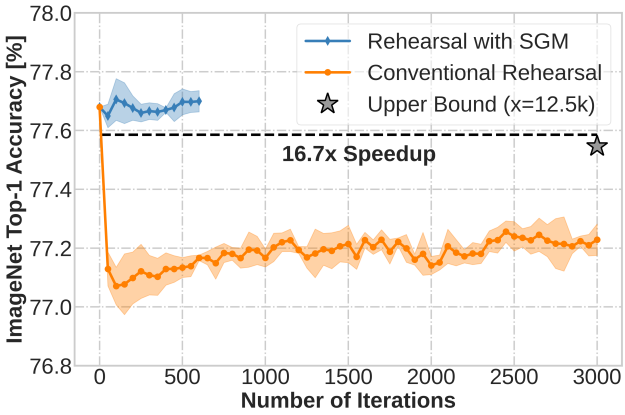


Figure 1: **An overview of *stability gap* phenomenon.** The stability gap is a phenomenon that occurs in CL when learning new data, where accuracy on previously learned data (Y-axis) drops significantly as a function of training iterations when a new distribution is introduced (X-axis). This plot illustrates this behavior for an average rehearsal cycle during CIL, where the network is trained on ImageNet-1K and then learns 365 new classes from Places365-LT over five batches. When rehearsal begins, accuracy on ImageNet-1k for conventional rehearsal drops dramatically before slowly recovering, although it fails to recover the original performance on the old data using 3000 iterations per rehearsal cycle. In this work, we seek to understand why the stability gap occurs through a series of hypothesis-driven experiments, resulting in a mitigation strategy that fully recovers accuracy. Our method, SGM, trains $16.7\times$ faster than offline fine-tuning with the combined 1365 class dataset.

We test two hypotheses for why the stability gap occurs in CIL:

1. ***The stability gap is caused in part due to having a large loss at the output layer for the new classes.*** To test this hypothesis, we study two methods to mitigate having a large loss in the output layer for the new classes. The first method is to initialize the output layer in a data-driven approach rather than randomly initializing the output units responsible for the new classes. The second method is a specialized form of soft-targets for the network, rather than the typical hard targets used for the network where these soft-targets are designed to improve performance for the new classes while minimally perturbing others.
2. ***The stability gap is caused in part due to excessive network plasticity.*** We test this hypothesis by controlling the level of plasticity in network layers in a dynamic manner. For hidden layers, we test this hypothesis using LoRA (Hu et al., 2021), which reduces the number of trainable parameters in the hidden layers of the network. For rehearsal methods, after each rehearsal cycle these weights are folded into the original network weights. For the output layer, we test this hypothesis by freezing the output units for classes seen in earlier batches during rehearsal.

In our experiments, we find that both hypotheses are supported. This leads us to develop a combined method that greatly reduces the stability gap for both CIL and other distributions.

This paper makes the following major contributions:

1. We are the first to study overcoming the stability gap. We test the aforementioned hypotheses in CIL and discover that they both play a role.
2. We propose novel metrics to measure the stability gap and to measure the speed of learning with respect to an offline upper bound.
3. We develop a method that greatly mitigates the stability gap, e.g., for CIL on ImageNet-1K combined with Places365-LT (1365 classes total), our method requires $16.7\times$ fewer network updates than an offline model. We show the method is also effective for non-CIL distributions.
4. We demonstrate that our method reduces the stability gap and enhances performance when combined with the rehearsal methods Vanilla Rehearsal, DERpp, GDumb, and REMIND as well as the non-rehearsal method LwF.

2 RELATED WORK

A variety of methods have been proposed to learn continuously from non-stationary datasets in continual learning (see Zhou et al. (2023) for review). These methods can be broadly divided into three categories: 1) **Rehearsal-based methods** store or reconstruct subset of old data to rehearse alongside new data while learning a new batch (Chaudhry et al., 2019; Hou et al., 2019; Rebuffi et al., 2017; Wu et al., 2019), 2) **Regularization-based methods** constrain weight updates by adding additional regularization in the loss function (Aljundi et al., 2018; Chaudhry et al., 2018a; Dhar et al., 2019; Kirkpatrick et al., 2017), and 3) **Parameter-isolation based methods** allocate multiple sets of parameters or multiple copies of the model to different incremental batches (Douillard et al., 2021; Yan et al., 2021; Yoon et al., 2020). Almost all CL methods focus on the catastrophic forgetting problem with evaluations occurring on discrete batch or task transitions (Chaudhry et al., 2018a) and fail to capture the stability gap that occurs immediately after a new task is introduced.

De Lange et al. (2023) demonstrates the stability gap occurs for a variety of CL methods including rehearsal (ER (Chaudhry et al., 2019), GEM (Lopez-Paz & Ranzato, 2017)), parameter regularization (EWC (Kirkpatrick et al., 2017)), and knowledge-distillation (LwF (Li & Hoiem, 2017)). To quantify this, they propose continual evaluation metrics based on worst-case performance i.e., the largest drop in accuracy on old batches. To capture the largest drop, their evaluation metric uses a window-based approach that compares the model's performance at the start of window to that at the end of window. However, their metric does not enable comparing *different* CL models because it is based on the same model without any universal upper bound. Instead, we use an offline model as a universal upper bound to compare CL methods (see Sec. 3.2). We study overcoming the stability gap with both rehearsal and non-rehearsal methods, but we focus on rehearsal since it performs better than others (van de Ven et al., 2022).

3 CONTINUAL LEARNING PROTOCOL

In this section, we formalize our CL framework and define the metrics we use.

3.1 FORMAL SETTING

A CL system is exposed to a sequence of data batches over N learning sessions, i.e., $\{\mathcal{S}_1, ..., \mathcal{S}_N\}$. The j'th session consists of a batch of m_j labeled training samples. Each $\mathcal{S}_j$ may follow a different distribution. Total number of samples in the entire sequence is $M = \sum_{j=1}^{N} m_j$. The batch (or task) identifier j cannot be exploited during test time. Following prior works (Belouadah & Popescu, 2019; Hou et al., 2019; Hayes et al., 2020), we use a base initialization phase where prior to CL the model has already learned the first batch $\mathcal{S}_1$. When learning new batch $\mathcal{S}_j$, the learner can access $\mathcal{S}_j$ and any stored data from previous batches $\mathcal{S}_{1:j-1}$. During test time, the learner is evaluated on test data from all seen batches.

To efficiently adapt to a large-scale data stream in real-world settings, a CL system should not increase compute cost over time. For all learning sessions j after base-initialization, it is given a fixed compute budget $\mathcal{B}_j$ corresponding to the number of SGD model updates. Therefore, the total compute budget after the N learning sessions is $\mathcal{B} = \sum_{j=2}^{N} \mathcal{B}_j$, where compute constraints are imposed during CL phase ($j > 1$). $\mathcal{B}$ should be set such that total number of SGD model updates across all sessions after base-initialization is less than the number of samples of the entire sequence M, i.e., $\mathcal{B} = fM$ where $f < 1$. For CL, we set $f = 0.3$ for computational efficiency.

3.2 METRICS

Existing metrics in CL are based on the performance after each $\mathcal{S}_j$ is learned. They do not permit fine-grained analysis for a) preserving old knowledge, b) acquiring new knowledge and c) balancing both during training. De Lange et al. (2023) introduced metrics to measure the stability gap, but these metrics are model dependent and cannot be used to compare different approaches. To address this, we created new metrics to compare methods for these criteria: 1) the stability gap, $\mathcal{S}_\Delta$ (criterion a) 2) the plasticity gap, $\mathcal{P}_\Delta$ (criterion b), and 3) the continual knowledge gap, $\mathcal{CK}_\Delta$ (criterion c).

Each metric asks: ***How much performance does the learner lack on previously observed, recently observed, or all observed data compared to an oracle (upper bound) when learning new data?***

For the j'th learning session, we denote evaluation sets on old, new, and all seen data by $\mathrm{E}_{1:j-1}$, E_j, and $\mathrm{E}_{1:j}$, respectively. $\mathcal{A}_i$ is the accuracy of the current model θ_i on batch evaluation set E at training iteration i, and L is the total number of iterations. $\mathcal{A}_{oracle}$ is the final accuracy of an offline model (θ_{oracle}) trained jointly on all data. We define the **stability gap** as

$$\mathcal{S}_\Delta = 1 - \frac{1}{N-1}\sum_{j=2}^{N}\Omega_j^{old}; \text{ where } \Omega_j^{old} = \frac{1}{L}\sum_{i=1}^{L}\frac{\mathcal{A}_i(\mathrm{E}_{1:j-1}, \theta_i)}{\mathcal{A}_{oracle}(\mathrm{E}_{1:j-1}, \theta_{oracle})}. \tag{1}$$

Similarly, the **plasticity gap** is

$$\mathcal{P}_\Delta = 1 - \frac{1}{N-1}\sum_{j=2}^{N}\Omega_j^{new}; \text{ where } \Omega_j^{new} = \frac{1}{L}\sum_{i=1}^{L}\frac{\mathcal{A}_i(\mathrm{E}_{j}, \theta_i)}{\mathcal{A}_{best}(\mathrm{E}_{j}, \theta_{best})}, \tag{2}$$

where $\mathcal{A}_{best}$ stands for the best accuracy achieved by best CL model (θ_{best}) any time during training. And finally we define the **continual knowledge gap** as

$$\mathcal{CK}_\Delta = 1 - \frac{1}{N-1}\sum_{j=2}^{N}\Omega_j^{all}; \text{ where } \Omega_j^{all} = \frac{1}{L}\sum_{i=1}^{L}\frac{\mathcal{A}_i(\mathrm{E}_{1:j}, \theta_i)}{\mathcal{A}_{oracle}(\mathrm{E}_{1:j}, \theta_{oracle})}. \tag{3}$$

Ω_j records CL performance compared to $\mathcal{A}_{oracle}$ or $\mathcal{A}_{best}$. After learning all N batches, Ω_j scores are averaged to indicate average performance gain. The first batch is excluded since that is used for base initialization. For all metrics, smaller $\mathcal{S}_\Delta$, $\mathcal{P}_\Delta$, and $\mathcal{CK}_\Delta$ indicate better performance. When $\mathcal{A}_i = \mathcal{A}_{oracle}$ and $\mathcal{A}_i = \mathcal{A}_{best}$ for all L iterations, $\mathcal{S}_\Delta$, $\mathcal{P}_\Delta$, and $\mathcal{CK}_\Delta$ become zero which is desirable. Negative value means knowledge transfer between new and old batches which is also desirable. These metrics are applicable for both incremental batch learning and online CL with any data distributions including CIL and IID.

4 METHODOLOGY

Here, we describe the methods we use to test our hypotheses for why the stability gap occurs.

Weight Initialization. In CIL, typically the output units for new classes are randomly initialized causing those units to produce a high loss during backpropagation. We hypothesize that using data-driven initialization for new class units will reduce the loss and therefore reduce the stability gap. To test this, we initialize them to the mean of unit length embeddings for that class, i.e.,

$$\mathbf{w}_k = \frac{1}{V}\sum_{j=1}^{V}\frac{\mathbf{h}_j}{\|\mathbf{h}_j\|_2}, \tag{4}$$

where $\mathbf{w}_k \in \mathbb{R}^d$ is the output layer weight vector for class k, $\mathbf{h}_j \in \mathbb{R}^d$ is the j'th embedding from the penultimate layer, and V is the number of samples from class k in the batch.

Hard vs. Dynamic Soft Targets. For classification, models are often trained with hard targets, i.e., at training iteration i a one-hot vector $\mathbf{t}_i$ with a '1' in position k corresponding to the correct class. We hypothesize training networks with hard targets is partially responsible for the stability gap due to the high loss caused by the new classes. To test this, we use soft targets constructed such that the model's predictions on previously learned classes are largely preserved.

At learning iteration i, let $P(\mathbf{x}_i; \theta_i)$ be the model's output softmax probabilities for sample $\mathbf{x}_i$ from class k given the model's current parameters θ_i and the predicted class be $y_i' = \arg\max_k P(k|\mathbf{x}_i; \theta_i)$. We maintain a running average vector $\mathbf{u}_k \in \mathbb{R}^K$ of the softmax probabilities for each class that is updated when an example from class k is observed, i.e.,

$$\mathbf{u}_k \leftarrow \frac{c_k\mathbf{u}_k + P(\mathbf{x}_i; \theta_i)}{c_k + 1}, \tag{5}$$

where c_k is a counter for class k that is subsequently increased by 1, and $\mathbf{u}_k$ is initialized to a uniform distribution prior to the running updates. Subsequently, soft targets $\mathbf{t}_i$ for iteration i are constructed

by setting $\mathbf{t}_i \leftarrow \mathbf{u}_k$ and then setting the element for the correct class to 1, i.e., $\mathbf{t}_i[k] \leftarrow 1$. If $y_i' \neq k$, then we also set $\mathbf{t}_i[y_i'] \leftarrow 1/K$. Subsequently, $\mathbf{t}_i$ is normalized to sum to 1 and used to update the network. This strategy results in targets that minimally perturb the network and smaller loss values.

Limiting Hidden Layer Plasticity Using LoRA. To accumulate knowledge over time, most CL approaches update the entire network. Given that each batch of data in CL is relatively small, we hypothesize that this leads to excessively perturbing hidden representations leading to a larger stability gap. To test this hypothesis, we constrain the number of trainable parameters in hidden representations by using a network adaptor. Specifically, we inject low rank adaptation (LoRA) (Hu et al., 2021) weights into the linear layers of the network, and only these parameters and the output layer are updated greatly reducing the number of trainable parameters.

For batch j, let $\mathbf{W}^{j-1} \in \mathbb{R}^{d \times g}$ be a previously learned linear layer. At the start of a each learning session, we reparameterize this layer by replacing $\mathbf{W}^{j-1}$ with

$$\mathbf{\Theta}^j = \mathbf{W}^{j-1} + \mathbf{B}\mathbf{A}, \tag{6}$$

where $\mathbf{B} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times g}$ are the LoRA adapter parameters with rank $r \ll \min(d, g)$. Only $\mathbf{B}$ and $\mathbf{A}$ are plastic, with $\mathbf{A}$ initialized with random Gaussian values and $\mathbf{B}$ initialized to a zero matrix, so $\mathbf{B}\mathbf{A} = \mathbf{0}$ at the beginning of the learning session. At the end of the session the LoRA parameters are folded into the network, i.e., $\mathbf{W}^j \leftarrow \mathbf{\Theta}^j$. In our LoRA experiments, only the output layer and the adapter parameters are plastic.

Limiting Output Layer Plasticity via Targeted Freezing. In CIL, large changes in the network's representations for old classes increase the stability gap. While LoRA restricts plasticity in hidden representations, we hypothesize that restricting plasticity in the output layer could also be helpful for CIL. We therefore explore freezing output layer weights for classes that have been previously learned in earlier batches. For rehearsal methods, samples from classes seen in earlier batches have the hidden layers updated as usual. We refer to this technique as old output class freezing (OOCF).

Combining Mitigation Methods & SGM. We independently evaluate each of the stability gap mitigation methods. Additionally, we evaluate them in combination. We refer to the method that combines dynamic soft targets, weight initialization, OOCF, and LoRA as SGM (**S**tability **G**ap **M**itigation). Soft targets and weight initialization prevent higher loss at the output layer to enhance stability. OOCF and LoRA restrict plasticity in the network to targeted locations so that existing representations are minimally perturbed.

5 Datasets & Network Architectures

Datasets. We use seven datasets in our experiments. Our main experiments in Sec. 6.1 and Sec. 6.1.1 use a combination of **ImageNet-1K** and the long-tailed **Places365-LT** datasets, where the model has been pre-trained on ImageNet-1K for base initialization and then must incrementally learn Places365-LT, while preserving performance on old batches. We study two CL orderings for Places365-LT: 1) CIL where each batch has classes exclusively seen in that batch, and 2) IID ordering where each batch contains samples from randomly sampled classes. These are two opposite extreme situations in CL. Although the stability gap is not known to occur in the IID CL setting, this setting is critical to demonstrating the algorithm's generality. We include IID CL experiments in Appendix D. In Sec. 6.2 and Appendix E.6, we use a combination of ImageNet-1K and the balanced **Places365-Standard** datasets. For online CL experiments in Appendix G, we use a combination of ImageNet-1K and **CUB-200** datasets. We also study **CIFAR-10** for learning from scratch experiments in Appendix F. We also use **CIFAR-100** and **ImageNet-R** datasets for prompt-based CL in Appendix K. Additional dataset details are given in Appendix B.

Network Architectures. In our main results, we study CL using the **ConvNeXtV2-Femto** (Woo et al., 2023) CNN that has been pre-trained on ImageNet-1K using a fully convolutional masked autoencoder framework followed by supervised fine-tuning on ImageNet-1K. While ResNet18 is widely used in CL, it under-performs other lightweight CNNs in CL (Hayes & Kanan, 2022). ConvNeXtV2-Femto has 5.2M parameters, which is $2\times$ less than ResNet18's 11.6M parameters, and it has better top-1 accuracy on ImageNet-1K (78.23%) than ResNet18 (69.76%). Each block of ConvNext consists of one 2D convolutional layer and two 1×1 convolutional layers. For experiments using LoRA, the 1×1 convolutional layers in ConvNeXt blocks are modified to incorporate LoRA's weights. The number of trainable LoRA weights is 0.92M, which is much less than the total

number of hidden layer parameters (5.08M). We also study the stability gap using **ConvNeXtV1-Tiny** (Liu et al., 2022) (see Appendix E.4) and **MobileViT-Small** (Mehta & Rastegari) (see Appendix E.5) which are pre-trained using supervised learning. We also use **ViT** for prompt-based CL in Appendix K.

6 EXPERIMENTS

We describe our findings on how the proposed method mitigates the stability gap (Sec. 6.1) and enhances learning efficiency (Sec. 6.1.1) in a memory unconstrained setting with rehearsal. We then study memory constrained rehearsal in Sec. 6.2. The aforementioned sections use ConvNextV2-Femto pre-trained on ImageNet for base initialization. We then summarize supporting experiments with other architectures, datasets, and CL settings, including online CL, in Sec. 6.3.

6.1 MAIN RESULTS

Continual Learning Procedure. In our main results, we assume the model is pre-trained on ImageNet-1K and then must incrementally learn Places365-LT over a sequence of CL batches, while preserving its performance on ImageNet-1K. Because our goal is to understand and mitigate the stability gap, our main results use rehearsal and assume the learner has access to all previously observed data, with no constraints on memory. This is aligned with finding a better alternative to periodically retraining from scratch as more data is acquired, which is commonly done in industry where the computational budget depends on compute to a far greater extent than data storage (Prabhu et al., 2023).

As shown in Fig. 1 and Fig. 4, conventional vanilla rehearsal with ImageNet-1K pre-trained backbone exhibits stability gap. Based on prior work, early layers in the network are universal feature extractors and are little altered during CL (Ramasesh et al., 2021a), so we freeze the first 4 blocks of the CNN in all experiments, leaving the remaining 8 blocks plastic (97.7% of the parameters). During CL, the model sequentially learns 5 incremental batches of data from Places365-LT. In the CIL ordering, each CL batch contains 73 categories. During rehearsal, the model is updated over 600 minibatches, where the minibatch consists of 128 samples where 50% are selected randomly from the current CL batch and 50% from data seen in earlier CL batches and ImageNet-1K. We use unconstrained memory setting where the memory buffer stores 1.34M samples from both datasets. To measure the stability gap, we assess performance during rehearsal every 50 minibatches, where the test set consists of the ImageNet-1K validation set and all of the classes from Places365-LT from the current and prior CL batches. See Appendix C for additional implementation details.

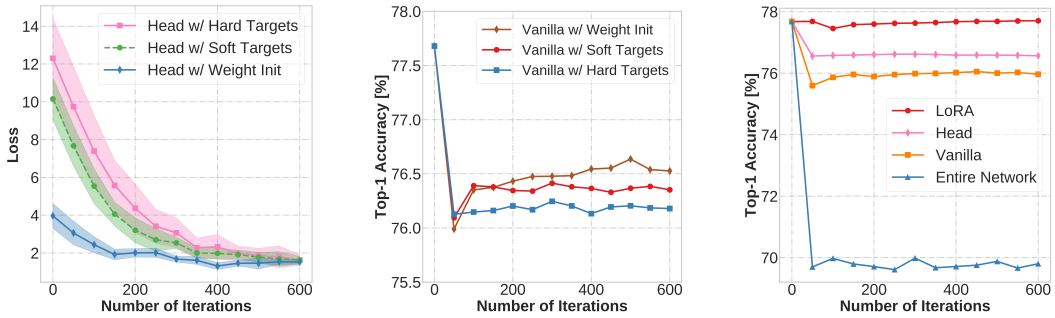
Baselines. In our experiments, we evaluate each of the methods in Sec. 4 individually and the combined SGM method. As baselines, we compare SGM with rehearsal against vanilla rehearsal, which is memory unconstrained rehearsal without any additional components, as well as offline models. The offline models are jointly trained on ImageNet and the CL batches seen up to the current batch. We also include naive finetune (vanilla variant) and head (only output layer is trainable).

Results. Our main CIL results are given in Table 1. SGM with rehearsal shows the greatest reduction in the stability gap ($\mathcal{S}_\Delta$), plasticity gap ($\mathcal{P}_\Delta$), and continual knowledge gap ($\mathcal{CK}_\Delta$). It also performs best in terms of final accuracy (α) and average accuracy over batches (μ). Of its components, LoRA reduces the stability gap the most; however, the stability gap for LoRA is $3\times$ higher than SGM. The results in Table 1 are for a single ordering of the CL batches. While it was not computationally feasible to use all CL batch orderings for every method, we repeated this experiment for 6 orderings for SGM and vanilla rehearsal. The averaged results across runs are given in Table 5, and we find that SGM consistently mitigates the stability gap achieving an $\mathcal{S}_\Delta$ of 0.001 compared to 0.020 for vanilla rehearsal. We next turn to examining the support for our two hypotheses.

Hypothesis 1. Our first hypothesis was that the stability gap in CIL is caused by having a large loss at the output layer due to the new classes, which we tested by using weight initialization and dynamic soft targets. Both methods are effective at achieving the goal of reducing the initial loss, especially weight initialization (see Fig. 2a). As shown in Table 1, both methods reduce the stability gap. We observe that weight initialization also greatly reduces the plasticity gap. Fig. 2b shows the

Table 1: **Class Incremental Learning Results.** Results after learning ImageNet-1K followed by Places365-LT over 5 batches using rehearsal. μ denotes average accuracy over batches and α is final accuracy on all 1365 classes. σ stands for final accuracy on ImageNet-1K only. $\#P$ is trainable parameters in Millions. Best and 2nd best values are indicated by bold and underline respectively.

Method	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\sigma(\uparrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	77.58	—	70.69
Naive Finetune	5.08	0.743	0.474	0.739	16.77	14.03	15.60
Output Layer (Head)	0.53	0.026	0.494	0.032	76.10	71.23	67.89
Vanilla Rehearsal	5.08	0.028	0.393	0.033	76.06	71.52	67.67
Dynamic Soft Targets	5.08	0.022	0.397	0.029	76.26	71.78	68.24
Weight Initialization	5.08	0.024	<u>0.097</u>	0.020	76.69	72.43	<u>69.22</u>
OOCF	5.08	0.026	0.376	0.032	75.95	71.57	67.94
LoRA	1.45	<u>0.018</u>	0.316	<u>0.019</u>	<u>76.92</u>	<u>72.74</u>	69.19
SGM	1.45	**0.006**	**0.082**	**0.002**	**77.64**	**73.70**	**70.30**



(a) Loss at output layer (b) Hard vs soft targets (c) Plastic layers

Figure 2: Mitigation methods averaged over 5 rehearsal cycles during CIL. (a) The loss on new classes when only training the output layer, which reveals soft targets and data-driven weight initialization greatly reduce the initial loss. (b) Accuracy on ImageNet-1K for hard vs. soft targets, which shows that soft targets reduce the stability gap. (c) Network plasticity increases the stability gap.

average performance on ImageNet-1K during the 5 rehearsal cycles for hard vs. soft targets, which reveals that hard targets increase the stability gap to a greater extent than the dynamic soft targets.

Hypothesis 2. Our second hypothesis was that the stability gap in CIL is caused by excessive plasticity in the network, which we tested by using LoRA and OOCF. Fig. 2c shows the average results on ImageNet-1K across the 5 rehearsal cycles for LoRA vs. when only the head, top 8 blocks (vanilla), or entire network are trainable. This reveals that plasticity plays a major role in the stability gap; however, this does not translate directly into the number of trainable parameters since LoRA includes the output head but exhibits a smaller decrease in performance than training only the output head of the network. Unlike others, LoRA fully recovers the performance on ImageNet-1K. As seen in Table 1, both OOCF and especially LoRA reduce the stability gap.

Interim Conclusions. Our experiments support both hypotheses. LoRA mitigates the stability and continual knowledge gaps the most. Weight initialization greatly improves the plasticity gap with LoRA helping to a lesser extent. We find that SGM, a method that combines the methods used to test these hypotheses, greatly reduces the stability gap. Compared to vanilla rehearsal, it reduces $\mathcal{S}_\Delta$, $\mathcal{P}_\Delta$, and $\mathcal{CK}_\Delta$ by a factor of $20\times$, $4.4\times$, and $15.5\times$, respectively (Table 5).

6.1.1 Learning Efficiency

One of our goals in studying CL and the stability gap is enabling more computationally efficient training. To study if SGM with rehearsal achieves this goal, we evaluated performance during CIL, where we measured performance on old and new classes every 10 iterations. For new class, we measured the number of updates and FLOPs needed to achieve 99% of the best accuracy.

Table 2: **Speed of Recovery.** SGM compared to vanilla rehearsal for the number of iterations needed to recover 97%, 98%, 99%, or 100% of the performance on ImageNet-1K for an offline model as each of the 5 batches (denoted by B) from Places365-LT are learned during CIL. A hyphen indicates the model did not recover performance whereas zero means there was no stability gap.

Recovery	With SGM					Without SGM				
	B_1	B_2	B_3	B_4	B_5	B_1	B_2	B_3	B_4	B_5
100%	260	0	70	50	80	—	—	—	—	—
99%	110	0	70	0	70	—	—	—	—	—
98%	90	0	70	0	70	60	450	110	—	—
97%	90	0	70	0	70	30	450	10	360	10

As shown in Fig. 3, SGM learns new classes much faster than vanilla rehearsal, and on average it is 61.8% more efficient. Moreover, because the curve shows a decreasing trend as learning progresses this means that SGM becomes a more efficient learner over time, with it generally requiring fewer updates for the current batch than previous batches. For old class performance, we measured the number of iterations during rehearsal needed to recover the performance of an offline model on ImageNet-1K for vanilla rehearsal and SGM. As shown in Table 2, SGM requires far fewer iterations to recover performance. Compared to an offline model trained on the combined dataset, SGM provides a 16.7× speed-up.

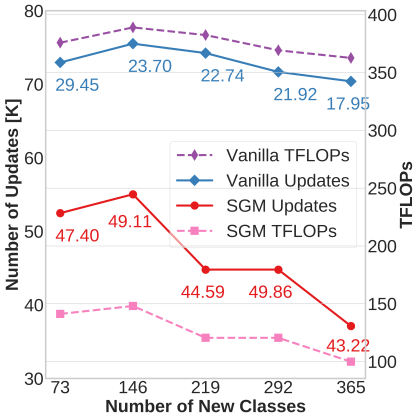


Figure 3: SGM requires fewer network updates and TFLOPs than vanilla rehearsal to reach 99% of best accuracy on new classes which is highlighted.

6.2 Memory Constrained CL Experiments

To study SGM's efficacy under memory constraints, we combined it with two popular rehearsal methods, DERpp (Buzzega et al., 2020) and GDumb (Prabhu et al., 2020), under varied memory constraints.

Training Procedure. After being pre-trained on ImageNet-1K, a model sequentially learns 5 incremental batches of data (73 classes per batch) from Places365-Standard. During learning a new batch, the model rehearses old data from memory buffer which is bounded by a maximum number of samples e.g., 192K and 24K corresponding to 6.4% and 0.8% of entire dataset (ImageNet and Places combined) respectively. Additional implementation details are given in Appendix C.

Results. As shown in Table 3, SGM improves each method's performance in all criteria. When the memory buffer is bounded by 24K samples, SGM improves final accuracy of DERpp and GDumb by 11.24% and 14.08% (absolute), respectively, and provides 2.4× and 3.1× more stability for DERpp and GDumb, respectively. Additional experiments are given in Appendices F, G, and I. SGM consistently mitigates the stability gap and enhances performance in all criteria. These results validate SGM's robustness in memory constrained CL.

Table 3: **Memory Constrained CL.** Results when SGM is combined with DERpp and GDumb for CIL on a combination of ImageNet-1K and Places365-Standard datasets. Here SGM† and SGM‡ denote variants of DERpp and GDumb respectively when SGM is integrated with them. μ denotes average accuracy over batches and α is final accuracy on all 1365 classes.

Method	Buffer (192K Samples)					Buffer (24K Samples)				
	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Joint	—	—	—	—	65.37	—	—	—	—	65.37
DERpp	0.142	0.109	0.126	62.53	53.38	0.209	0.109	0.187	57.25	44.74
SGM†	**0.071**	**0.091**	**0.061**	**67.41**	**57.28**	**0.086**	**0.095**	**0.074**	**66.29**	**55.98**
GDumb	0.133	0.110	0.120	62.85	54.72	0.224	0.114	0.202	55.54	43.10
SGM‡	**0.053**	**0.095**	**0.046**	**68.50**	**59.24**	**0.073**	**0.098**	**0.065**	**66.88**	**57.18**

6.3 SUPPORTING EXPERIMENTS

We include a number of supporting experiments in the Appendix and summarize the findings here.

IID CL. In IID CL, SGM fully overcomes the stability gap and outperforms both vanilla rehearsal and offline models (see Appendix D), showing that SGM's benefits are not specific to CIL.

Memory Constrained Learning from Scratch. In main results, the CL model is pre-trained on ImageNet-1K for base initialization. In Appendix F, we also evaluate SGM's efficacy when the CL model learns from scratch without base initialization. Using a memory buffer bounded by 5K samples, SGM with rehearsal outperforms vanilla rehearsal by 3.45% (absolute) in final accuracy.

Memory Constrained Online CL. In Appendix G, we examine SGM's efficacy in the memory constrained online CL setting using a state-of-the-art online CL method, REMIND (Hayes et al., 2020). SGM improves REMIND's final accuracy by 8.64% (absolute).

Non-Rehearsal Methods. While non-rehearsal methods are less performant for CL, it is interesting to study SGM when combined with non-rehearsal methods to determine if our findings are consistent. We find that, SGM also benefits a non-rehearsal method, LwF (Li & Hoiem, 2017) with an absolute gain of 35.24% in final accuracy and a $2.6\times$ reduction in the stability gap (see Appendix H).

Class-Balanced Rehearsal. Since our main experiments are based on conventional rehearsal without class balance, we also conduct stability gap experiments where class-balanced rehearsal is combined with SGM and improves performance over prior results (see Appendix E.3).

Supervised Pre-Training. In main results, SGM reduces the stability gap using the self-supervised ConvNeXtV2. To examine how SGM performs without a self-supervised backbone, we also experiment with ConvNeXtV1 which has been pre-trained using supervised learning. We find that SGM with rehearsal mitigates the stability gap without the self-supervised backbone (see Appendix E.4).

Vision Transformers. We also study the behavior of SGM with rehearsal compared to vanilla rehearsal with vision transformers (ViTs), which now rival CNNs. We find that SGM with ViT also reduces the stability gap by a factor of $2.4\times$ (see Appendix E.5).

Balanced Dataset. We examined how SGM with rehearsal compares with vanilla rehearsal on a balanced dataset, Places365-Standard in Appendix E.6. Our findings also hold for this dataset.

Additional Memory Constrained Experiments. We perform memory constrained CL experiments using Places365-LT in Appendix I. We find that SGM with rehearsal remains as effective as memory unconstrained CL whereas vanilla rehearsal's performance degrades.

7 DISCUSSION & CONCLUSION

With the growing energy usage of deep learning models (Luccioni et al., 2022; Patterson et al., 2021; Wu et al., 2022), we believe CL can play an important role in reducing carbon emissions. Despite making progress in mitigating catastrophic forgetting, almost all CL methods provide little computational benefit (Harun et al., 2023a). For efficiency gains by CL to be realized, we argued that the stability gap in CL must be addressed. We identified two major factors for the stability gap in CL: 1) high loss at the output layer caused by distribution shifts due to learning new classes, and 2) excessive network plasticity. To examine this, we introduced novel metrics and conducted hypothesis-driven analysis. Both of our hypotheses are supported by our experiments in large-scale image classification tasks. We demonstrated that SGM largely mitigates the stability gap in CIL and overcomes it for IID CL. SGM mitigates the stability gap across different CL algorithms, network architectures, and datasets, demonstrating its broad applicability. SGM significantly increases learning and computational efficiency. Our study was limited to image classification tasks. It would be interesting to study stability gap in other tasks such as object detection (Acharya et al., 2020), language understanding (Jang et al.) and so on. Due to computational limitations, we could not study CL for more than 1365 classes. In future work, it would be interesting to assess how well SGM scales as both the size of the dataset and number of classes increase.

REPRODUCIBILITY STATEMENT

All datasets and pre-trained models used in this work are widely used and publicly available. A GitHub repository and website will be created for replicating the experiments in this paper. It will include the exact ordering of the sequence of examples used for each CL experiment. The Appendix has a detailed description of the hyperparameter and optimizer settings used for each experiment.

REFERENCES

Manoj Acharya, Tyler L Hayes, and Christopher Kanan. Rodeo: Replay for online object detection. In *BMVC*, 2020.

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

Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 583–592, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018a.

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

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486, 2019*, 2019.

Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. In *ICLR*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5138–5146, 2019.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer vision-ECCV 2020-16th European conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, volume 12365, pp. 86–102. Springer, 2020.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. *arXiv preprint arXiv:2111.11326*, 2021.

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

Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new paradigm. In *CVPR*, 2023.

Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, and Christopher Kanan. How efficient are today's continual learning algorithms? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2430–2435, June 2023a.

Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, Ronald Kemker, and Christopher Kanan. SIESTA: Efficient online continual learning with sleep. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL https://openreview.net/forum?id=MqDVlBWRRV.

Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. In *CoLLAs*, 2022.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pp. 466–483. Springer, 2020.

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

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Chip Huyen. *Designing machine learning systems*. " O'Reilly Media, Inc.", 2022a.

Chip Huyen. Introduction to streaming for data scientists. https://huyenchip.com/2022/08/03/stream-processing-for-data-scientists.html, 2022b.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, KIM Gyeonghun, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*.

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.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *ICLR*, 2018.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34: 28648–28662, 2021.

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

Kuan-Ying Lee, Yuanyi Zhong, and Yu-Xiong Wang. Do pre-trained models benefit equally in continual learning? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6485–6493, 2023.

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

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2537–2546, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022.

Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. Matchmaker: Data drift mitigation in machine learning for large-scale systems. *Proceedings of Machine Learning and Systems*, 4: 77–94, 2022.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24 (214):1–50, 2023.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodriguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents*, pp. 60–91. PMLR, 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.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pp. 524–540, 2020.

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

Yifan Sun Teng Xi Gang Zhang Bernard Ghanem Jian Zhang Qiankun Gao, Chen Zhao. A unified continual learning framework with general parameter-efficient tuning. *International Conference on Computer Vision (ICCV)*, 2023.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *ICLR*, 2021a.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2021b.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

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.

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

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11909–11919, 2023.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. arxiv. *arXiv preprint arXiv:1708.07120*, 2017.

Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pp. 1–13, 2022.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. *arXiv preprint arXiv:2301.00808*, 2023.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023, 2021.

Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *Eighth International Conference on Learning Representations, ICLR 2020*. ICLR, 2020.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*, 2023.

Yao Zhu, YueFeng Chen, Chuanlong Xie, Xiaodan Li, Rong Zhang, Hui Xue, Xiang Tian, Yaowu Chen, et al. Boosting out-of-distribution detection with typical features. *Advances in Neural Information Processing Systems*, 35:20758–20769, 2022.

Appendix

A Implementation Details and Additional Results

We organize additional supporting experimental findings as follows:

- Appendix B provides details on the datasets used in this paper.
- Appendix C provides additional implementation and training details for all of the methods.
- Appendix D includes experimental results on IID CL and demonstrates that SGM's efficacy is not specific to CIL.
- Appendix E provides additional CIL experiments and results with rehearsal, including an analysis of learning curves, studying alternative sampling strategies for rehearsal, using a non-self-supervised backbone CNN, using a vision transformer, and using a balanced dataset. We find that SGM works well across these experiments and analysis compared to vanilla rehearsal.
- Appendix F includes findings on memory constrained learning from scratch experiments. In this setting, SGM outperforms vanilla rehearsal in all criteria.
- Appendix G studies SGM in memory constrained online CL setting using a state-of-the-art online learning method, REMIND. We observe that SGM combined with REMIND enhances performance in all metrics under various memory constraints.
- Appendix H studies the behavior of our stability gap mitigation method when used with Learning without Forgetting (LwF), a popular regularization method used in CL. We find that our method greatly improves results, illustrating that the mitigation strategy is not specific to rehearsal.
- Appendix I studies both CIL and IID CL on a long-tailed dataset when the memory buffer is constrained to only 100K samples. We find that SGM's performance is almost entirely unaffected with this memory constraint, whereas vanilla rehearsal's performance decreases across all metrics.
- Appendix J includes additional experimental results for learning efficiency comparison.
- Appendix K compares SGM with prompt-based CL methods.
- Appendix L provides justifications for our experimental setup.

B Dataset Details

This paper uses seven benchmark datasets e.g., ImageNet-1K, Places365-LT, Places365-Standard, CUB-200, CIFAR-10, CIFAR-100, and ImageNet-R. ImageNet-1K (Russakovsky et al., 2015) has 1.2 million images from 1000 categories, each with $732-1300$ training images and 50 validation images. Places365-LT (Liu et al., 2019) is a long-tailed dataset with an imbalanced class distribution. It is a long-tailed variant of the Places-2 dataset (Zhou et al., 2017). Places365-LT has 365 classes and 62500 training images with 5 to 4980 images per class. For its test set, we use the Places365-LT validation set from (Liu et al., 2019) which consists of a total of 7300 images with a balanced distribution of 20 images per class. Places365-Standard (Zhou et al., 2017) has over 1.8 million training images from 365 classes with $3068-5000$ images per class. We use the validation set consisting of 100 images per class to test the models. CUB-200 (Wah et al., 2011) has RGB images of 200 bird species with 5994 training images and 5794 test images. CIFAR-10 (Krizhevsky et al., 2009) consists of 10 classes with 50000 training images and 10000 test images. CIFAR-100 (Krizhevsky et al., 2009) consists of 100 classes and 500 training images and 100 testing images per class. ImageNet-R (Ke et al., 2020) consists of 200 classes and 24000 training images and 6000 test images.

C Additional Implementation Details

In this section we provide additional implementation details for the models.

Main Experiments. For both CIL and IID experiments, we train SGM with rehearsal, vanilla rehearsal, and head using cross-entropy loss for 600 iterations per rehearsal cycle. During each

iteration model is updated on 128 samples. All methods use the same ConvNextV2 backbone [1], use AdamW optimizer with weight decay of 0.05 and initial learning rates of 10^{-3} (SGM and vanilla) and 10^{-2} (head). The learning rate is reduced in earlier layers by a layer-wise decay factor of 0.9. The learning rate scheduler is not applied for vanilla and head due to poor performance. On the other hand, SGM uses OneCycle learning rate scheduler (Smith & Topin, 2017). The offline model is trained for 12500 iterations on all data i.e., ImageNet-1K and Places365-LT combined using an initial learning rate of 10^{-4} without a scheduler. For all experiments, we set the rank of the LoRA weight matrices to 48. In all cases, all metrics are based on Top-1 accuracy ($\%$). In general, most CL experiments including those in Sec. 6.1 adhere to the aforementioned settings unless otherwise noted.

Memory Constrained CL with DERpp and GDumb. We describe settings used in Sec. 6.2 where we combine SGM with DERpp and GDumb while using identical settings e.g., the same ImageNet-1K pre-trained ConvNeXt V2 Femto network and same optimizer settings. Each model pre-trained on ImageNet-1K learns Places365-Standard in 5 batches subsequently (73 categories per batch). Each rehearsal cycle contains total 1200 iterations with 256 samples per iteration. We set $f = 0.5$ for compute budget. DERpp employs distillation and regularization along with rehearsal to prevent catastrophic forgetting. It regularizes loss on old samples and uses an additional distillation loss on logits of old samples for promoting consistency. We set coefficients $\alpha = 0.1$ and $\beta = 0.9$ for distillation and regularization respectively. GDumb randomly removes a sample from the largest class when buffer reaches its maximum capacity and maintains a class-balanced memory buffer. For all methods, memory buffer is bounded by maximum number of samples (80% ImageNet-1K + 20% Places365-Standard). DERpp, GDumb, and SGM use an initial learning rate of 1×10^{-3}, 1×10^{-3}, and 1.5×10^{-3}, respectively, for batch size 256. The offline model uses an initial learning rate of 10^{-2} and 12K iterations with 256 samples per iteration. We assess performance during rehearsal every 100 minibatches to compute the metrics.

Class-balanced Rehearsal. For class balanced rehearsal experiments in Appendix E.3, SGM with rehearsal and vanilla rehearsal use an initial learning rate of 10^{-3} and 10^{-4}, respectively.

Non-SSL Backbone CNN. For non-SSL backbone experiments with ConvNeXt V1-Tiny (Liu et al., 2022) in Appendix E.4, initial learning rates for SGM with rehearsal, vanilla rehearsal, and offline model are 4×10^{-3}, 3×10^{-3}, and 10^{-4} respectively. ConvNeXt V1-Tiny has been pre-trained on ImageNet-1K using supervised learning [2].

ViT Backbone. For ViT backbone experiments in Appendix E.5, we select MobileViT-Small (Mehta & Rastegari) (5.6M) pretrained on ImageNet-1K using supervised learning [3]. For universal feature extraction, we freeze the first 8 blocks including stem, 6 MobielNetV2 blocks, and 1 MobileViT block. We keep remaining blocks (1 MobileNetV2 block and 2 MobileViT blocks) and layers (1 CNN layer and 1 linear layer) plastic which correspond to 96.4% of the total parameters. We apply LoRA (rank=48) to query, key and value projection matrices in the self-attention module of MobileViT transformer blocks. All methods use the AdamW optimizer with weight decay of 0.01. Vanilla rehearsal and SGM with rehearsal use an initial learning rate of 3×10^{-3} and 4×10^{-3}, respectively. The initial learning rate for the offline model is 10^{-2}. Places365-LT data is learned over 5 rehearsal cycles (73 classes per cycle) where each cycle includes 1200 iterations with 32 samples per iteration. Offline model is trained for 25K iterations with 64 samples per iteration. All other settings are identical to those in the main experiments.

Balanced (Non-LT) Dataset. For experiments with balanced dataset in Appendix E.6, SGM with rehearsal and vanilla rehearsal use an initial learning rate of 1.5×10^{-3} and 10^{-3}, respectively. Each model is pre-trained on ImageNet-1K and then learns Places365-Standard in 5 batches subsequently (73 categories per batch). Each rehearsal cycle has total 1200 iterations with 256 samples per iteration. We set $f = 0.5$ for compute budget. Hence, at the end of CL, total number of SGD model updates is 50% of total number of samples in the entire dataset (ImageNet and Places-Standard combined). We assess performance during rehearsal every 100 minibatches to compute the metrics. The offline model use an initial learning rate of 10^{-2} and 12K iterations with 256 samples per iteration.

[1] Pre-trained weights are available here: `https://github.com/facebookresearch/ConvNeXt-V2`

[2] The pre-trained weights are available here: `https://github.com/facebookresearch/ConvNeXt`

[3] The pre-trained weights are available here: `https://github.com/apple/ml-cvnets`

Memory Constrained Learning from Scratch. In Appendix F, both SGM with rehearsal and vanilla rehearsal use the AdamW optimizer with initial LR of 0.005 and weight decay of 0.05 for batch size 64. We reduce the LR for old class units in output layer by a factor of 0.9. We create Femto version of ConvNeXtV1 following ConvNextV2 Femto configuration and modify stem layer with 3×3 kernels and stride 1 to account for 32×32 image resolution of CIFAR-10 dataset. Following ConvNeXtV1 (Liu et al., 2022), we use a cosine scheduler for learning rate decay and weight decay. For learning from scratch, we do not use any pre-trained weights. Each model learns CIFAR-10 in 5 incremental batches with 2 classes per batch. We use 50 epochs for each batch and 10 linear warmup epochs for the first batch only. We assess performance during rehearsal every 5 epochs to compute the metrics. The offline model is trained on entire CIFAR-10 dataset for 100 epochs with 20 linear warmup epochs. It uses an initial LR of 0.004 and other details adhere to aforementioned settings.

Memory Constrained Online CL. In Appendix G, we use identical settings and hyperparameters for both REMIND and REMIND + SGM methods. We use ImageNet-1K pre-trained ConvNeXt V2 Femto with similar network configurations and LoRA configurations as used in main experiments. We use the AdamW optimizer and REMIND's default per-class learning rate scheduler. We set the initial learning rate to 1×10^{-3}, the final learning rate to 1×10^{-5}, and weight decay to 0.05. Following REMIND, we perform rehearsal with a mini-batch of 51 samples including 50 old samples and 1 new sample. Each method learns CUB-200 dataset in *sample-by-sample* manner. For all methods, the memory buffer is bounded by maximum number of samples ($75\% - 93\%$ ImageNet).

Regularization Methods. In Appendix H, LwF has similar configurations as vanilla rehearsal except for the initial learning rate (6×10^{-5}). LwF + SGM has a similar configurations as SGM with rehearsal except the initial learning rate (2×10^{-4}). During each iteration model is updated on 64 new samples without any rehearsal of old samples. Our mitigation approaches e.g., dynamic soft targets, data-driven weight initialization, OOCF, and LoRA are applied for LwF similarly as they are applied for rehearsal methods in main experiments.

Weight Initialization. For SGM, we use our data-driven weight initialization (Sec. 4) to initialize the weights in the final output layer. For other compared methods, we use He initialization (He et al., 2015) to initialize the weights in the final output layer.

All other settings adhere to the above mentioned general settings for the main experiments unless otherwise mentioned. Hyperparameters are tuned to maximize performance for each method. We run all experiments on the same hardware with a single GPU (NVIDIA RTX A5000).

D IID Continual Learning

To understand if SGM with rehearsal would be useful for other CL data distributions, we examine its behavior in an IID ordering where each of the 5 CL batch contains randomly sampled classes from Places365-LT. During IID CL, the model sequentially learns 5 incremental batches of data from Places365-LT where each incremental batch contains 12500 examples. Our results are summarized in Table 4. In terms of final accuracy, SGM achieves a final accuracy of 71.23%, outperforming vanilla rehearsal's 68.77% accuracy, and surprisingly even the offline model's 70.69% accuracy. SGM achieves a negative stability gap, which indicates knowledge transfer from new classes to old classes. In contrast, we found there was a small stability gap in CIL ordering, likely due to the dissimilarity among subsequent batches.

E Additional CIL Analysis & Experiments

In this section we conduct additional analysis of the CIL experiments in the main results as well as present additional experiments.

E.1 Learning Curves

In our main text, our figures are averaged across rehearsal cycles. In Fig. 4, we instead present all of the learning curves in sequence, where we denote when the next batch containing new classes is received.

Table 4: **IID Continual Learning**. Results after learning ImageNet-1K followed by Places365-LT over 5 batches with 12500 samples per batch. Here μ and α denote average accuracy and final accuracy respectively. $\#P$ denotes trainable parameters in Millions. Reported value is the average of 5 runs with standard deviation (SD) placed in parentheses as ($\pm$SD).

Method	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	—	70.69
Vanilla Rehearsal	5.08	0.014 ($\pm$0.0004)	0.173 ($\pm$0.0056)	0.034 ($\pm$0.0005)	68.45 ($\pm$0.0470)	68.77 ($\pm$0.0517)
SGM	**1.45**	**−0.004** ($\pm$0.0005)	**0.131** ($\pm$0.0017)	**0.003** ($\pm$0.0004)	**70.81** ($\pm$0.0151)	**71.23** ($\pm$0.0664)

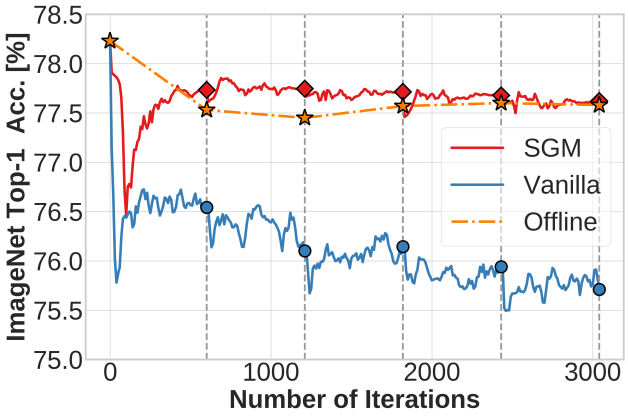


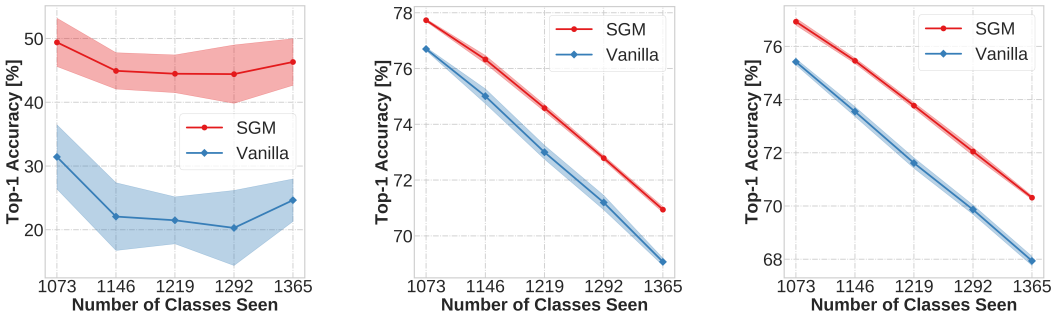
Figure 4: **Stability gap in all rehearsal cycles.** After pretraining on ImageNet-1K, the model learns 365 new classes from Places365-LT over five rehearsal cycles (73 new classes and 600 iterations per rehearsal cycle). SGM quickly recovers old performance in the beginning of CL whereas vanilla fails to obtain full recovery. After each rehearsal cycle (vertical dotted gray line), final accuracy is highlighted by diamond (SGM), star (offline), and circle (vanilla).

When rehearsal begins, accuracy on ImageNet-1k for vanilla rehearsal drops dramatically and gradually decreases throughout the rehearsal cycles. At the end, vanilla fails to recover the original performance using total 3K iterations. In contrast, SGM shows better performance throughout rehearsal cycles with reduced stability gap and full recovery compared to the offline model. Models are evaluated every 10 iterations. After each rehearsal cycle, SGM outperforms vanilla and matches or exceeds offline accuracy.

In Fig. 5, we also illustrate model's accuracy on new, old, and all classes in all rehearsal cycles where SGM achieves higher accuracy than vanilla. This indicates that SGM consistently improves model's plasticity (Fig. 5a), stability (Fig. 5b), and knowledge accumulation (Fig. 5c).

E.2 Repeated CIL Experiments

The results in Table 1 are for a single ordering of the CL batches. While it was not computationally feasible to use all CL batch orderings for every method, we repeated this experiment for 6 orderings for SGM and vanilla rehearsal. We also included head for comparison where we froze all layers except the final layer and trained final layer during rehearsal. The averaged results across runs are given in Table 5, and we find that SGM consistently mitigates the stability gap achieving an $\mathcal{S}_\Delta$ of 0.001 compared to 0.020 for vanilla rehearsal. Besides that SGM achieves outperforming scores in every other criteria compared to vanilla. SGM also outperforms head in all criteria. This indicates that updating representations in earlier layers besides head using SGM is critical for learning new knowledge and retaining old knowledge.



(a) Accuracy on new classes (b) Accuracy on old classes (c) Accuracy on all classes

Figure 5: **Stability-plasticity.** After pre-training on ImageNet-1K, the model learns 365 new classes from Places365-LT over five batches (73 new classes per batch) in CIL setting. The accuracy is averaged over 6 runs and shaded region indicates standard deviation.

Table 5: **Results averaged over 6 runs (CIL)**. Experimental results are based on ImageNet-1K and Places365-LT datasets. A continual learner pre-trained on ImageNet learns Places in 5 batches subsequently (73 categories per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total number of trainable parameters in Millions. Reported value is the average of 6 runs with standard deviation (SD) placed in parentheses as ($\pm$SD).

Method	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	—	70.69
Vanilla Rehearsal	5.08	0.020 ($\pm$0.0017)	0.385 ($\pm$0.0091)	0.031 ($\pm$0.0015)	71.68 ($\pm$0.1236)	67.94 ($\pm$0.1721)
Head	**0.53**	0.021 ($\pm$0.0011)	0.473 ($\pm$0.0250)	0.032 ($\pm$0.0009)	71.28 ($\pm$0.1410)	67.68 ($\pm$0.2710)
SGM	1.45	**0.001** ($\pm$0.0012)	**0.087** ($\pm$0.0082)	**0.002** ($\pm$0.0007)	**73.71** ($\pm$0.0763)	**70.31** ($\pm$0.0682)

E.3 Class Balanced Uniform Sampling for Rehearsal

In our main results, we sampled randomly during rehearsal without balancing for each class. However, prior work has shown that class balanced random sampling works significantly better than unbalanced uniform sampling for long-tailed datasets (Harun et al., 2023b). We conducted a CIL experiment to examine this in our memory unconstrained rehearsal setup where we learn ImageNet-1K followed by Places365-LT.

Table 6 shows that using class balanced rehearsal, SGM improves performance in most criteria compared to previous results without class balance (Table 1). When both vanilla and SGM use class balanced rehearsal, SGM outperforms vanilla by 7.3$\times$ in stability gap, 3.6$\times$ in plasticity gap and provides continual knowledge transfer ($\mathcal{CK}_\Delta < 0$).

E.4 Analysis with a Non-Self-Supervised Backbone CNN

Much of deep learning has moved toward self-supervised pre-training prior to supervised fine-tuning, especially in foundation models (Devlin et al., 2018; Brown et al., 2020; Ramesh et al., 2021), since this has been shown to reduce overfitting on the pretext dataset used for self-supervised learning and to generalize better to downstream tasks. In the main text, we used the self-supervised ConvNextV2 architecture. This may have enabled our system to achieve higher results on Places365-LT than if the CNN was initialized from ImageNet-1K with supervised learning. To determine if our general trends for the methods hold, we conducted another experiment with ConvNeXtV1-Tiny (29M), which is pre-trained on ImageNet-1K without self-supervision.

Table 6: **Class Balanced Rehearsal**. Experimental results are based on ImageNet-1K and Places365-LT datasets. A continual learner pre-trained on ImageNet learns Places in 5 batches subsequently (73 categories per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total number of trainable parameters in Millions.

Method	$\#P(\downarrow)$	$\mathcal{S}_{\Delta}(\downarrow)$	$\mathcal{P}_{\Delta}(\downarrow)$	$\mathcal{CK}_{\Delta}(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	—	70.69
Vanilla Rehearsal	5.08	0.022	0.316	0.021	72.24	69.03
SGM	**1.45**	**0.003**	**0.089**	**−0.003**	**74.00**	**70.61**

Table 7: **CIL without Self-Supervised Pre-Training**. This table shows results from ConvNeXt V1-Tiny pre-trained on ImageNet-1K using supervised learning, which then learns Places365-LT in 5 batches subsequently (73 categories per batch) in class-incremental setting. Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total trainable parameters in Millions.

Method	$\#P(\downarrow)$	$\mathcal{S}_{\Delta}(\downarrow)$	$\mathcal{P}_{\Delta}(\downarrow)$	$\mathcal{CK}_{\Delta}(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	27.00	—	—	—	—	74.16
Vanilla Rehearsal	27.00	0.030	0.396	0.035	74.73	70.67
SGM	**3.53**	**0.005**	**0.102**	**0.001**	**77.48**	**73.92**

Experimental results in Table 7 demonstrate that SGM with supervised backbone mitigates the stability gap and enhances performance in all criteria. Therefore efficacy of SGM does not depend upon self-supervised pre-training.

E.5 Analysis with using a Vision Transformer Backbone

In this section we study the behavior of the system for a ViT model pre-trained with supervised learning. For this, we select a light-weight transformer, MobileViT-Small (Mehta & Rastegari). MobileViT learns local and global representations using convolutions and transformers, respectively. It has total 5.6 million parameters and top-1 accuracy of 78.4% on ImageNet-1K.

Table 8 shows the comparison between vanilla and SGM when they have same MobileViT backbone. SGM shows better performance in all criteria using $3.8\times$ fewer parameters than vanilla rehearsal.

Table 8: **Vision Transformer Backbone**. Experimental results are based on ImageNet-1K and Places365-LT datasets. A continual learner pre-trained on ImageNet learns Places in 5 batches subsequently (73 categories per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total number of trainable parameters in Millions.

Method	$\#P(\downarrow)$	$\mathcal{S}_{\Delta}(\downarrow)$	$\mathcal{P}_{\Delta}(\downarrow)$	$\mathcal{CK}_{\Delta}(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	4.97	—	—	—	—	69.10
Vanilla Rehearsal	4.97	0.039	0.434	0.046	70.18	66.35
SGM	**1.30**	**0.016**	**0.140**	**0.016**	**72.09**	**67.96**

E.6 Balanced (Non-LT) Dataset

In real-world setting, data distribution is commonly long-tailed and imbalanced, hence we used Places365-LT dataset in the main results. However, our analysis holds for balanced and non-LT dataset as well. We study this using Places365-Standard. Results in Table 9 show that SGM outperforms vanilla rehearsal in all criteria.

F Memory Constrained Learning From Scratch

In the main text, we define our problem setting with a base initialization phase where a model acquires base knowledge using a pre-train dataset. Here we also test another problem setting without the base initialization phase where a model learns from scratch. We study stability gap and efficacy of SGM when a model is trained from scratch on CIFAR-10 dataset in 5 rehearsal cycles (2

Table 9: **Non-LT Dataset**. Experimental results are based on ImageNet-1K and Places365-Standard datasets. A continual learner pre-trained on ImageNet learns Places in 5 batches subsequently (73 categories per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total number of trainable parameters in Millions.

Method	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	—	65.37
Vanilla Rehearsal	5.08	0.078	0.201	0.082	66.37	56.63
SGM	**1.45**	**0.054**	**0.091**	**0.047**	**68.47**	**59.21**

classes per rehearsal cycle). Since the model is trained on small number of training data, it learns less transferable representations. Therefore, instead of using LoRA and freezing old class units in output layer, we train all layers and update old class units with lower learning rate. Whereas we use dynamic soft targets and data-driven weight initialization as used in our main experiments. Following DERpp (Buzzega et al., 2020), we bound buffer size by 5K samples. For comparison, we also include unbounded buffer that contains all 50K samples of CIFAR-10 dataset. We summarize our findings in Table 10 where SGM achieves higher scores than vanilla rehearsal in all metrics. SGM outperforms vanilla rehearsal by 3.43% (50K samples in buffer) and 3.45% (5K samples in buffer) in final accuracy.

Table 10: **Learning from scratch**. A model learns CIFAR-10 from scratch in 5 incremental batches (2 classes per batch). Memory buffer is bounded by max number of samples. Here μ, α, and $\#P$ denote average accuracy over batches, final accuracy, and parameters (Millions) respectively.

Method	Buffer	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Vanilla Rehearsal	50K	4.83	0.217	0.047	0.166	77.60	66.91
SGM	50K	4.83	**0.192**	**0.046**	**0.147**	**78.64**	**70.34**
Vanilla Rehearsal	5K	4.83	0.445	0.061	0.331	70.33	53.63
SGM	5K	4.83	**0.422**	**0.058**	**0.318**	**71.06**	**57.08**

G Memory Constrained Online Continual Learning

In the main text, we study stability gap in incremental batch learning setting. Here we study stability gap in online continual learning setting using a state-of-the-art online learning method, REMIND (Hayes et al., 2020). We conduct memory constrained CL experiments with CIL data ordering, where we combine SGM with REMIND while using identical configurations. We summarize the results in Table 11. We observe that SGM combined with REMIND (REMIND + SGM) outperforms REMIND (without SGM) by large margins in all metrics and shows effectiveness in online learning setting. We also observe that SGM maintains similar effectiveness across various memory constraints.

Table 11: **Online Continual Learning**. A model pre-trained on ImageNet-1K learns CUB-200 sample-by-sample with a replay mini-batch of 51 samples (50 old + 1 new). Here μ, α, and $\#P$ denote average accuracy over batches, final accuracy, and parameters (Millions) respectively.

Method	Buffer	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	—	5.08	—	—	—	—	75.99
REMIND	80994	5.08	0.146	0.837	0.162	64.15	62.35
REMIND + SGM	80994	**1.45**	**0.034**	**0.675**	**0.049**	**72.81**	**69.67**
REMIND	44394	5.08	0.156	0.834	0.172	63.37	60.94
REMIND + SGM	44394	**1.45**	**0.034**	**0.661**	**0.049**	**72.80**	**69.63**
REMIND	20000	5.08	0.166	0.844	0.183	62.58	60.37
REMIND + SGM	20000	**1.45**	**0.042**	**0.695**	**0.057**	**72.20**	**69.01**

Table 12: **Comparison with regularizaton method**. A continual learner pre-trained on ImageNet-1K learns Places365-LT in 5 batches subsequently (73 categories per batch) in CIL setting. Results are averaged over 6 runs. Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total trainable parameters in Millions. For regularization baseline, we select LwF that regularizes model based on knowledge distillation.

Method	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	5.08	—	—	—	—	70.69
Vanilla Rehearsal	5.08	0.020	0.385	0.031	71.68	67.94
SGM	**1.45**	**0.001**	**0.087**	**0.002**	**73.71**	**70.31**
LwF	5.08	0.605	0.450	0.607	24.04	4.76
LwF + SGM	**1.45**	**0.236**	**0.072**	**0.235**	**54.87**	**40.00**

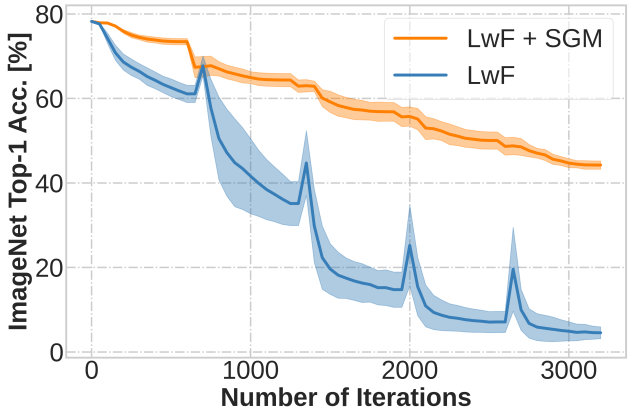


Figure 6: **Comparison with regularization method.** The Y-axis shows average accuracy of 6 runs with standard deviation (shaded region). The network is trained on ImageNet-1K and then learns 365 new classes from Places-LT over five batches (73 new classes and 600 iterations per batch). When new batch arrives, accuracy on ImageNet-1k for LwF plummets. LwF fails to recover performance and ends up with large stability gap. In contrast, LwF with SGM does not plummet like LwF and shows better performance throughout CL phase with significantly reduced stability gap.

H Using our Stability Gap Mitigation Method with Regularization Methods

We hypothesized that SGM would be helpful for non-rehearsal methods as well. We therefore study SGM using Learning without Forgetting (LwF) (Li & Hoiem, 2017), which pioneered using knowledge distillation in CL (Zhou et al., 2023). Instead of rehearsal, LwF stores a copy of the model before learning the new CL batch to update the model with distillation. LwF has been shown to reduce catastrophic forgetting in a range of CL scenarios, although it and other regularization-based methods have not been shown to be effective in the CIL setting (Zhou et al., 2023).

We conducted an experiment to compare vanilla LwF with a version of LwF that uses SGM without rehearsal during CIL of ImageNet and Places365-LT. Overall results are given in Table 12 and a learning curve is given in Fig. 6. As expected based on prior results, rehearsal methods vastly outperform LwF; however, we find that SGM provides an enormous benefit to LwF in terms of reducing the stability gap, resulting in increased accuracy.

I Memory Constrained Experiments with LT Dataset

Since, the memory constraint was relaxed in the main results for LT dataset (Places365-LT), here we study the stability gap under memory constraints when learning ImageNet-1K followed by CL of Places365-LT. The setup is otherwise identical to that used in Sec. 6.1. In memory restricted CL for both CIL and IID settings, the learner can store and access only 7.5% of entire dataset (ImageNet

and Places combined). Now learner has access to 100K samples (old and current data combined) compared to unconstrained setup where learner had access to 1.34M samples.

Table 13: **Memory constrained CL (CIL)**. A continual learner pre-trained on ImageNet-1K learns Places365-LT in 5 batches subsequently (73 categories per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total trainable parameters in Millions. First two rows are memory unconstrained methods for comparison. Memory is constrained in terms of maximum number of instances (2nd column) a model can store in the buffer.

Method	Max instances	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	1343667	5.08	—	—	—	—	70.69
Vanilla	1343667	5.08	0.028	0.393	0.033	71.52	67.67
SGM	1343667	**1.45**	**0.006**	0.082	**0.002**	**73.70**	**70.30**
Vanilla	100900	5.08	0.040	0.388	0.044	70.62	65.99
SGM	100900	**1.45**	**0.006**	**0.081**	**0.002**	73.67	70.23

Table 14: **Memory constrained CL (IID)**. A continual learner pre-trained on ImageNet-1K learns Places365-LT in 5 batches subsequently (12500 samples per batch). Here μ denotes average accuracy over batches and α is final accuracy. $\#P$ denotes total trainable parameters in Millions. First two rows are memory unconstrained methods for comparison. Memory is constrained in terms of maximum number of instances (2nd column) a model can store in the buffer.

Method	Max instances	$\#P(\downarrow)$	$\mathcal{S}_\Delta(\downarrow)$	$\mathcal{P}_\Delta(\downarrow)$	$\mathcal{CK}_\Delta(\downarrow)$	$\mu(\uparrow)$	$\alpha(\uparrow)$
Upper Bound	1343667	5.08	—	—	—	—	70.69
Vanilla	1343667	5.08	0.014	0.177	0.033	68.45	68.68
SGM	1343667	**1.45**	**−0.004**	0.129	**0.003**	70.80	**71.14**
Vanilla	100900	5.08	0.027	0.173	0.045	67.50	66.90
SGM	100900	**1.45**	**−0.004**	**0.128**	**0.003**	**70.81**	71.07

Following the common practice of storing 120K instances for ImageNet-1K with rehearsal (Rebuffi et al., 2017), we set the memory upper bound to 100K instances where 38K instances are randomly sampled from the ImageNet-1K dataset and stored in the memory buffer and remaining 62K are incrementally added to the buffer as Places365-LT is learned continually.

Our results for memory constrained rehearsal for CIL are summarized in Table 13. Results for memory constrained rehearsal in the IID setting are summarized in Table 14. Our observations and conclusions about SGM and vanilla rehearsal made in unconstrained CL still hold for memory constrained CL. In the constrained setup, overall accuracy drops and the stability gap worsens for vanilla rehearsal, whereas SGM is largely unaffected.

J Additional Learning Efficiency Comparisons

In the main text, we considered number of network updates for learning efficiency. In this section, we also measure FLOPs (floating-point operations) using DeepSpeed [4]. As shown in Fig. 7, SGM requires $31.9\times$ fewer TFLOPs than an offline fine-tuning model to achieve similar accuracy. In terms of training time, SGM requires 40 minutes whereas offline fine-tuning model requires 12.1 hours for ImageNet-1K and Places365-LT combined dataset. Fig. 8 shows training time required for learning new classes. Learning efficiency comparison between GDumb and GDumb with SGM is given in Fig. 9. All results confirm that SGM significantly enhances computational efficiency for CL.

K Prompt-based CL

Previously, we compared SGM with a variety of CL methods including rehearsal methods (vanilla, GDumb), rehearsal and knowledge distillation methods (DERpp), non-rehearsal and knowledge distillation methods (LwF), and online CL methods (REMIND) to demonstrate that SGM's efficacy is

[4] https://github.com/microsoft/DeepSpeed

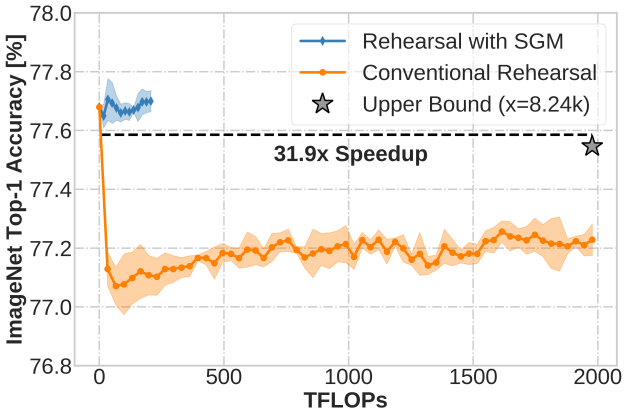


Figure 7: **Speed of recovery.** Our method, SGM, provides $31.9\times$ speedup in terms of TFLOPs compared to an offline fine-tuning with the combined 1365 class dataset (ImageNet-1K + Places365-LT).

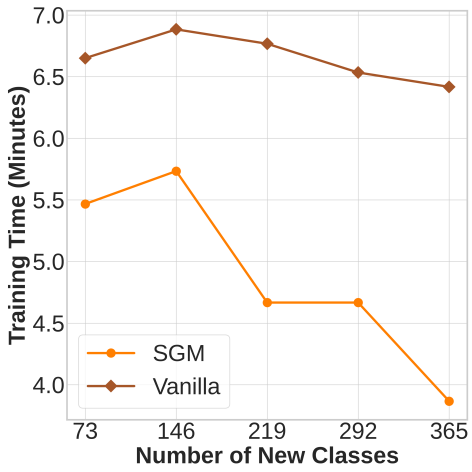


Figure 8: **Timing experiment.** SGM requires less training time than vanilla rehearsal to reach 99% of best accuracy on new classes (Places365-LT).

broadly applicable. In this section, we compare SGM with prompt-based CL methods e.g., DualPrompt (Wang et al., 2022a). For rehearsal-free CL, DualPrompt learns a set of prompt parameters to effectively instruct a pretrained model. Following DualPrompt, we use CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-R (200 classes) (Ke et al., 2020) datasets. We use ImageNet-1K pre-trained ViT-Ti/16 (7M parameters) architecture [5] that achieved 69.15% top-1 accuracy on ImageNet-1K. We compare all methods in a rehearsal-free setting (buffer=0) where each model learns 10 tasks in CIL. Each task contains 10 classes for CIFAR-100 and 20 classes for ImageNet-R. For all datasets, we use Adam optimizer and learning rate of 0.0025 for batch size 128. Other implementation details follow DualPrompt. For SGM, we apply LoRA (rank=192) to query, key and value projection matrices in the self-attention module of ViT blocks. The results are summarized in Table 15. We find that SGM enhances DualPrompt's performance on both datasets.

[5]The pre-trained weights are available here: `https://github.com/google-research/vision_transformer`

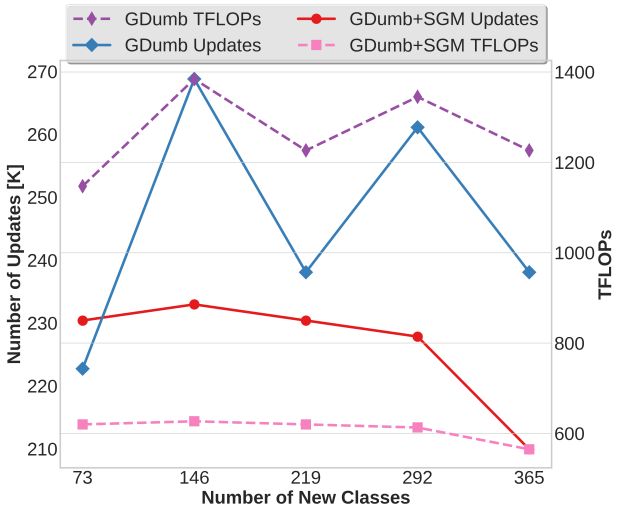


Figure 9: While adapting to new classes (Places365-Standard), GDumb+SGM requires fewer network updates and TFLOPs than GDumb to reach 99% of best accuracy on new classes.

Table 15: **Prompt-based CL.** Results when SGM is combined with DualPrompt. DualPrompt updates query, key and value projection matrices in the self-attention module of ViT blocks besides classifier head and prompts similar to SGM. Reported metrics are based on top-1 accuracy (%).

Method	CIFAR-100		ImageNet-R	
	Avg. Acc (↑)	Forgetting (↓)	Avg. Acc (↑)	Forgetting (↓)
DualPrompt	13.59	80.91	6.93	49.03
DualPrompt + SGM	**70.66**	**11.14**	**37.53**	**35.90**

L JUSTIFICATION FOR OUR EXPERIMENTAL SETUP

Memory Constraints. For CL on ImageNet-1K+Places365-Standard in Sec. 6.2, we bound the buffer size by 24K samples, which is only 0.8% of the total number of images in the two datasets (3M from the 1.2M in ImageNet-1K and 1.8M in Places). Prior work on CIL with ImageNet-1K uses 20K images (1.7% of the 1.2M in ImageNet-1K) (Hou et al., 2019; Castro et al., 2018; Rebuffi et al., 2017). Therefore as a fraction of the total dataset, our constrained experiments are more restrictive. For online CL on ImageNet-1K+CUB-200 in Sec. G, we bound the buffer size by 20K samples, which is only 1.55% of the total number of images in the two datasets (1.29M from the 1.28M in ImageNet-1K and 5.9K in CUB). For training from scratch CL on CIFAR-10 in Sec. F, following prior work (Buzzega et al., 2020), we bound the buffer size by 5K samples.

We vary buffer sizes to demonstrate that unlike compared methods, SGM maintains good performance across various memory constraints. We also conduct experiments without memory constraints to remove that variable from analysis to understand the effectiveness of our methods, and because it aligns with industry's interest in CL, where compute costs significantly exceed storage costs (Prabhu et al., 2023).

Compute Constraints. Inspired by prior works (Prabhu et al., 2023; Harun et al., 2023b;a), we bound compute by number of training iterations or SGD steps. We control compute by a hyperparameter f defined in Sec. 3. Lower f yields higher computational efficiency. In a real-world setting, a continual learner has to adapt to a large-scale data stream (ideally never-ending), thereby requiring more computation than size of the data stream may not be feasible. For example, for many applications such as on-device learning, embedded devices, and AR/VR, a continual learner has to learn new information quickly without increasing computational overhead. Our principal focus is to align CL with resource-constrained applications. That's why we choose $f = 0.3$.

Dataset Choice. The second datasets to be learned are chosen to be challenging. Given the *significant overlap* between ImageNet and CIFAR datasets (Kornblith et al., 2021), CIFAR exhibits little concept shift and is a poor test for the stability gap. Conversely, Places is challenging (Liu et al., 2019) and is widely used for out-of-distribution (OOD) detection with ImageNet pre-trained models (Zhu et al., 2022). CUB-200 is also more challenging than CIFAR-100 (Lee et al., 2023). We also consider CIFAR-10 and CIFAR-100 datasets for training from scratch and prompt-based CL experiments.

Architecture Choice. While ResNet-18 has been historically used, several CL works showed that ResNet-18 underperforms similar sized CNNs (Hayes & Kanan, 2022; Harun et al., 2023b). Moreover, ConvNeXtV2-Femto performs better than ResNet-18 (78.25% vs 69.76% on ImageNet-1K) using $2\times$ fewer parameters (5.2M vs. 11.6M parameters) (Liu et al., 2022). Many recent works use vision transformers and more advanced architectures (Wang et al., 2022b;a; Smith et al., 2023; Qiankun Gao, 2023), and we believe that CL must use the latest advances for the community to care about it rather than focusing on the ResNet architecture which is now almost a decade old. It is also worth noting that LoRA is not appropriate for ResNet architectures, but it can be used with ViT models, ConvNeXt, ConvNeXtV2, etc. Besides CNNs, we also present experiments with ViTs in Sec. E.5 and Sec.K.

Pretrained Models. Use of pretrained models aligns with many real-world applications, where a model is trained from scratch on a large volume of data, deployed, and then continually updated (Huyen, 2022a;b). Moreover, there are a growing number of CL works, that use large pretrained ViT models that have been pretrained on ImageNet-1K or ImageNet-21K (Wang et al., 2022b;a; Smith et al., 2023; Qiankun Gao, 2023). Many prior works use pretraining for CL (Belouadah & Popescu, 2019; Hou et al., 2019; Castro et al., 2018; Rebuffi et al., 2017; Hayes et al., 2020; Douillard et al., 2020; Harun et al., 2023b) and advocate the importance of pretraining for CL (Lee et al., 2023; Mehta et al., 2023; Ostapenko et al., 2022; Ramasesh et al., 2021b). However, as demonstrated in several works (Wang et al., 2022a; Mirzadeh et al., 2022), using pretrained models does not naively enhance CL performance and effectively leveraging pretrained models for CL remains an open question.

Reasons for New Metrics. The metrics in (De Lange et al., 2023) are not normalized and focus on worst-case performance during continual evaluation of a model over a sequence of tasks. This means it cannot be used to analyze whether the model is meeting the needs of industry to catch-up to the offline upper bound. By normalizing our scores with a universal offline model trained jointly on all of the data from scratch (see Sec. 3.2), we can then compare across approaches. Specifically, their worst-case evaluation attempts to find the largest drop relative to the same model's best performance, which may be quite far from an offline model. Instead, our metrics measure performance compared to a much stronger upper bound.