Appendix of Dealing with Changing Training Bias in Replay for Online Continual Learning

Anonymous CVPR submission

Paper ID 10165

1. Appendix 1. Sub-figures (D), (E) and (F) for Figure 1 of the Main Paper

The sub-figures are shown in Figure 1 here.

2. Appendix 2: Figure of the average cross-entropy loss of the new data batch and the replay data batch

We plot the average loss in Figure 2 and observe that the average cross-entropy loss of the new data batch is consistently higher than the loss of the replay data batch.

3. Appendix 3. Sub-figures (D), (E), and (F) Using the SGD Optimizer for Figure 2 of the Main Paper

The sub-figures are shown in Figure 3 here.

4. Appendix 4: Proof for the upper bound

Assume that the system has seen $n-1$ previous classes $(c_1, ..., c_{n-1})$ and there are $m-n$ new classes $(c_n, ..., c_m)$ in the current new data batch. (1) It's easy to see that $\mathcal{L}_{decom}(x_{c_n}) = \mathcal{L}_{ce}(x_{c_n})$ if $m-n=1$ because both losses degenerate to just establishing the boundary between the new class and old classes and their forms are the same. (2) It is also easy to see that $\mathcal{L}_{decom}(x_{c_n}) = \mathcal{L}_{ce}(x_{c_n})$ when $n-1=0$ because both losses degenerate to just establishing the decision boundaries between the new classes within the new data batch and their formulas become the same. (3) So let us consider the case that $m-n>1$ and $n-1>0$. For a sample x_{c_n} of class c_n from X^{new}, its cross-entropy loss is:

$$\mathcal{L}_{ce}(x_{c_n}) = -\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right) \tag{1}$$

where o_{c_s} is the logit value of x_{c_n} for class c_s. Then we have:

$$\begin{aligned}\mathcal{L}_{ce}(x_{c_n}) &= -\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{m} e^{o_{c_s}}} * \frac{\sum_{s=1}^{n} e^{o_{c_s}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right)\\ &= -\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right) - \log\left(\frac{\sum_{s=1}^{n} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right)\end{aligned} \tag{2}$$

Also, we have:

$$\begin{aligned}\mathcal{L}_{ce}(x_{c_n}) &= -\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{m} e^{o_{c_s}}} * \frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right)\\ &= -\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right) - \log\left(\frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right)\end{aligned} \tag{3}$$

Combining Eq. 2 and Eq. 3, we get:

$$\begin{aligned}\mathcal{L}_{ce}(x_{c_n}) = &-\frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right) - \frac{1}{2}\log\left(\frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right)\\ &-\frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right) - \frac{1}{2}\log\left(\frac{\sum_{s=1}^{n} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right)\end{aligned} \tag{4}$$

For the loss $\mathcal{L}_{decom}(x_{c_n})$, we have:

$$\begin{aligned}\mathcal{L}_{decom}(x_{c_n}) = &-\frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right) - \frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right)\\ &-\frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right) - \frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right)\end{aligned} \tag{5}$$

Considering the second term in Eq. 4 and the last term in Eq. 5, we have:

$$\begin{aligned}\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}} &= 1 - \frac{\sum_{s=1}^{n-1} e^{o_{c_s}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\\ \frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}} &= 1 - \frac{\sum_{s=1}^{n-1} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\end{aligned} \tag{6}$$

As $\sum_{s=n+1}^{m} e^{o_{c_s}} \geq 0$, so we have:

$$\begin{aligned}\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}} &\leq \frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\\ \log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right) &\leq \log\left(\frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right)\end{aligned} \tag{7}$$

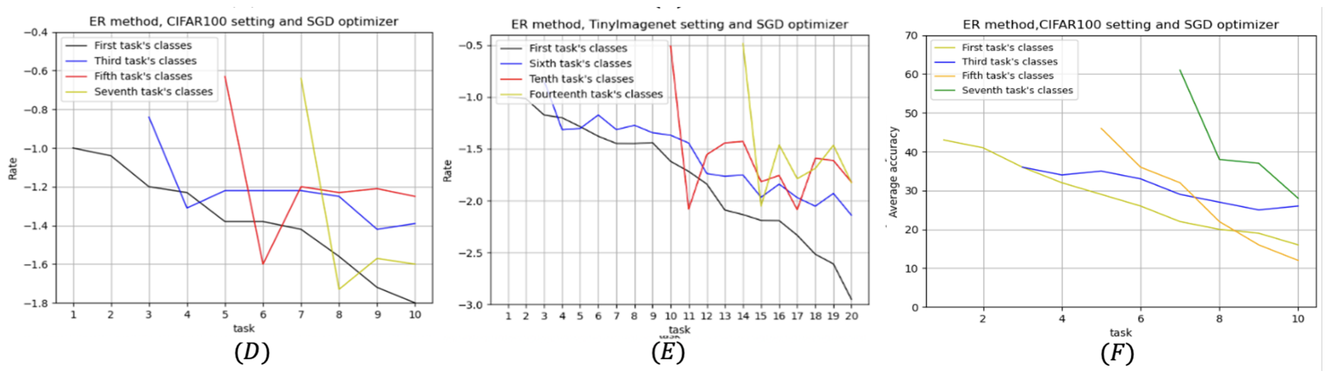


Figure 1. In (D) and (E), we report *PN* rate (*rate* in the figures) of the CIFAR100 or TinyImageNet experiments with the SGD optimizer. The memory buffer size is 1000. We choose four different tasks and plot their *PN* rates as subsequent tasks are learned. In (F), we plot the test accuracy in the CIFAR100 experiment using the SGD optimizer.

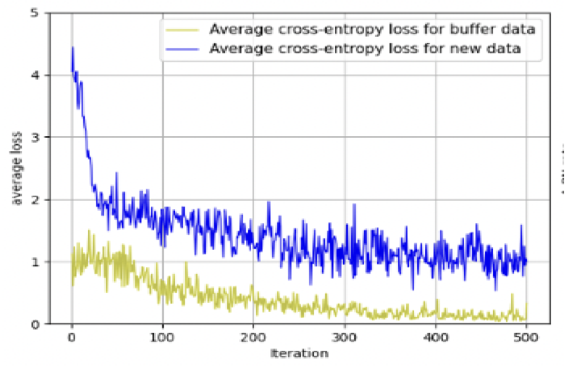


Figure 2. The average cross entropy loss of the new data batch and the replay data batch when the model is learning the second task on the CIFAR100 dataset.

Similarly, for the last term in Eq. 4 and the second term in Eq. 5, we have:

$$\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right) \leq \log\left(\frac{\sum_{s=1}^{n} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right) \tag{8}$$

Considering Eq. 10 and Eq. 11, we obtain:

$$\mathcal{L}_{decom}(x_{c_n}) - \mathcal{L}_{ce}(x_{c_n}) = \frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=n}^{m} e^{o_{c_s}}}\right) \\ -\frac{1}{2}\log\left(\frac{\sum_{s=1}^{n} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right) + \frac{1}{2}\log\left(\frac{e^{o_{c_n}}}{\sum_{s=1}^{n} e^{o_{c_s}}}\right) - \frac{1}{2}\log\left(\frac{\sum_{s=n}^{m} e^{o_{c_s}}}{\sum_{s=1}^{m} e^{o_{c_s}}}\right) \tag{9}$$

Then we have $\mathcal{L}_{decom}(x_{c_n}) - \mathcal{L}_{ce}(x_{c_n}) \geq 0$

Following (1), (2) and (3), we have proved that $\mathcal{L}_{decom}(x_{c_n})$ is a upper bound of $\mathcal{L}_{ce}(x_{c_n})$. The proof for a sampled data point x_{c_j} of class c_j $(j < n)$ from the buffer is similar.

5. Appendix 5: Justification for the GSA-CE Loss

Assuming x_i is a data sample of class c_i, the gradient on each logit of $o(x_i;\theta,\phi)$ is given by:

$$\frac{\partial \mathcal{L}_{ce}(o(x_i;\theta,\phi))}{\partial o_{c_i}} = \frac{1}{e^{A\text{-}PN(t',t,y_k)+1}} \cdot H(o_{c_i}) \tag{10}$$

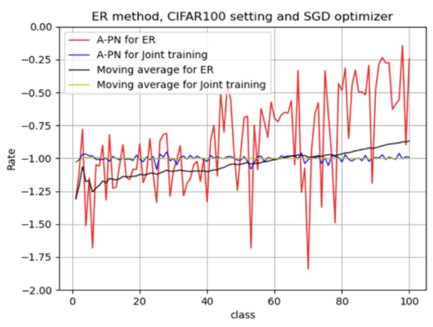
where $H(o_{c_i}) = \frac{\frac{e^{o_{c_i}}}{-PN(t,c_i)}}{\sum_{s=1\bigcap s\neq i}^{|C_{seen}|} e^{o_{c_s}} + \frac{e^{o_{c_i}}}{-PN(t,c_i)}} - 1$ and for $j \neq i$

$$\frac{\partial \mathcal{L}_{ce}(o(x_i;\theta,\phi))}{\partial o_{c_j}} = \frac{1}{e^{A\text{-}PN(t',t,y_k)}+1} \cdot H(o_{c_j}) \tag{11}$$

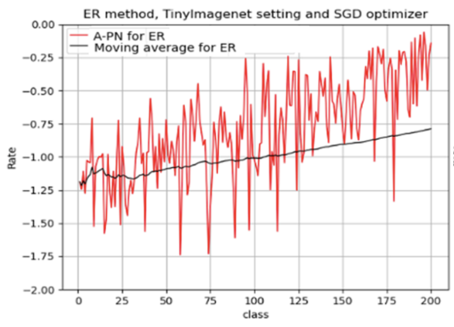
where $H(o_{c_j}) = \frac{e^{o_{c_j}}}{\sum_{s=1\bigcap s\neq i}^{|C_{seen}|} e^{o_{c_s}} + \frac{e^{o_{c_i}}}{-PN(t,c_i)}}$. For a class c_i, the smaller its accumulated gradient rate (A-$PN(t',t,c_i)$) is, the bigger its sample weight w_i is. So the model can automatically adjust the loss weight w_i to emphasize the class that has a smaller accumulated gradient rate because the smaller accumulated gradient means that the model underrates this class. However, since the accumulated gradient rate *A-PN* is calculated from all seen training data, it is not sensitive to short-term tendency. To capture short-term variations within a task, we introduce $\frac{1}{-PN(t,c_i)}$ to the logit of c_i. From Eq. 10 and Eq. 11 above, we know that if the positive gradient of c_i within a task surpasses the negative gradient of c_i within the task ($\frac{1}{-PN(t,c_i)} < 1$), the negative gradient will be adjusted by increasing $\frac{1}{-PN(t,c_i)}$ until $\frac{1}{-PN(t,c_i)} = 1$ and the positive gradient for the logits of other classes (not c_i) will be increased to help the model distinguish samples of other classes, and vice versa.

6. Appendix 6: Official Code of Baselines, Hyperparameters

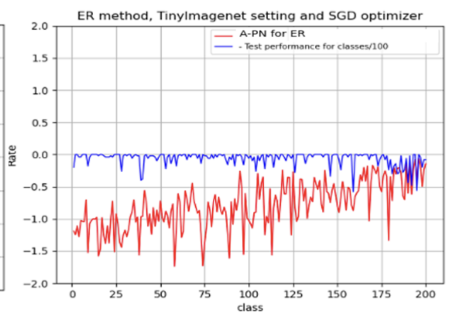
Following [12], we set each data batch (X^{new}) size N^{new} to 10 for all systems. We use X^{new} as the input of group



(D)



(E)



(F)

Figure 3. In (D) and (E), the accumulated A-PN rate (*rate* in the figures) for each class (x-axis) after the last task is trained. The buffer size is 1000. The datasets used are CIFAR100 and TinyImageNet. In (F), we report the test accuracy of all seen classes in the TinyImageNet setting with the SGD optimizer.

2. For the input of group 1 (X^{mix}), we set it to 64 in GSA. Specifically, we first sample $max(int(64 \cdot \frac{|L^{(t)}|}{\sum_{r=1}^{t}|L^{(r)}|}), 1)$ data points from X^{new} and sample $N^{buf} = 64 - N^{new}$ data points from the memory buffer, and then we combine them to produce X^{mix}. For fair comparison, for baselines, we also set their memory buffer batch (X^{buf}) size to 64 (it does not change with tasks). We use the Apex (A PyTorch Extension) https://nvidia.github.io/apex/ to accelerate training for all methods. "opt_level" is "O1". We run all experiments on a Tesla V100 32G. The code links for baselines are

The code for ER and MIR: https://github.com/optimass/Maximally_Interfered_Retrieval.
The code for ASER and SCR: https://github.com/RaptorMai/online-continual-learning.
The code for GDumb: https://github.com/drimpossible/GDumb.
The code for LUCIR: https://github.com/hshustc/CVPR19_Incremental_Learning.
The code for AGEM: https://github.com/facebookresearch/agem.
The code for GSS: https://github.com/rahafaljundi/Gradient-based-Sample-Selection.
The code for ER-AML: https://github.com/pclucas14/AML.
The code for CCIL:https://github.com/sud0301/essentials_for_CIL.
The code for BiC:https://github.com/sairin1202/BIC.
The code for SSIL:https://github.com/hongjoon0805/SS-IL-Official.

7. Appendix 7: Forgetting Rate and Training time

The average forgetting rate is computed as follows [7]: after training the model from task 1 to task j, we denote $acc_{j,i}$ as the accuracy of the trained model evaluated on the held-out test set of task $i \le j$. The average forgetting rate FR_t at task t is:

$$FR_t = \frac{\sum_{i=1}^{t-1} f_i^t}{t-1}, \text{where} f_i^t = \max_{l \in \{1,2,\ldots,t-1\}} (acc_{l,i} - acc_{t,i}) \tag{12}$$

Training times of our system GSA and the baselines are given in Figure 4. Our method is slower than some baseline systems due to the extra computation needed to ensure gradient balance. However, with the ever-increase in computing power of new hardware, this extra computation should not be a major issue. Some code optimization should also improve our algorithm greatly.

Table 1. Accuracy of the long-tailed CIFAR100 (20 tasks) experiment with different memory buffer sizes *M*. All values are averages of 15 runs.

Method	Long-tailed CIFAR100		
M	M=0.1k	*M*=1*k*	M=5k
AGEM [5]	5±5.2	5.2±8.8	6.6±0.2
ER [6]	8.7±0.2	10.7±0.2	18.9±0.1
MIR [2]	7.0±0.1	10.1±0.1	19.0±0.1
GSS [3]	7.4±0.5	10.7±5.8	15.5±0.2
ASER [12]	8.6±0.3	9.9±0.1	16.5±0.4
ER-AML [4]	6.5±0.3	8.5±0.2	13.0±0.7
GDumb [11]	9.2±0.3	12.7±0.2	21.9±0.3
SCR [9]	10.2±0.1	15.8±0.3	23.2±0.7
LUCIR [8]	5.2±0.3	8.2±0.1	13.1±0.5
CCIL [10]	12.4±0.4	15.8±0.2	19.0±0.1
OCS [14]	11.1±0.2	17.1±0.2	25.1±0.1
BiC [13]	13.0±0.3	21.0±0.2	28.1±0.8
SSIL [1]	12.1±0.3	18.2±0.2	26.1±0.6
GSA	**14.1**±0.2	**23.2**±0.6	**30.1**±0.6

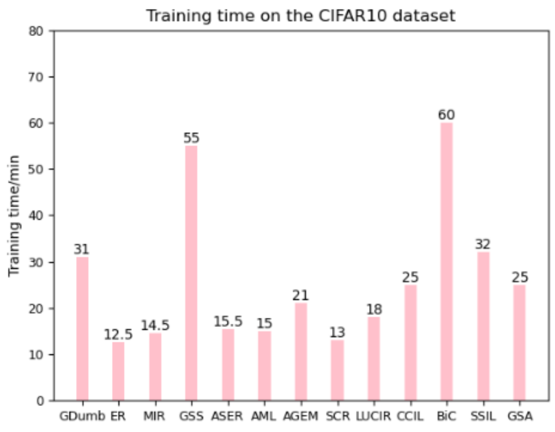


Figure 4. Training times of our method GSA and baselines on CIFAR10 in minutes.

8. Appendix 8: Long-tail Online Continual Learning Experiments Setup

This setting has 20 tasks created from the CIFAR100 dataset and each task consists of 5 randomly chosen classes from the 100 classes of CIFAR100. The number of training samples n_c in each class c is

$$n_c = n_{c,orig} \times u^{index(c)} \tag{13}$$

where $n_{c,orig}$ is the original number of training samples for class c in the CIFAR100 dataset and $index(c)$ is the class index of class c (0-indexed) and $u \in (0, 1)$. We set u as 0.98. The backbone and the optimizer are the same as those in the main experiments. We list the final average test performance in Table 1. From the table, we observe that our method outperforms the baselines over three different memory sizes.

References

[1] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 844–853, 2021. 3

[2] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019. 3

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019. 3

[4] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. Reducing representation drift in online continual learning. *arXiv preprint arXiv:2104.05025*, 2021. 3

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 3

[6] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. In *ICML-2019*, 2019. 3

[7] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. 3

[8] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 3

[9] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3589–3599, 2021. 3

[10] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3513–3522, 2021. 3

[11] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *EECV*, pages 524–540, 2020. 3

[12] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. *arXiv preprint arXiv:2009.00093*, 2020. 2, 3

[13] 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 (CVPR)*, 2019. 3

[14] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021. 3