Appendix

Form.	Dataset	Model	#Params	k	Acc
B1	MNIST	MLP-400	479K	4400	97.8
		NIN	260K	250	97.2
		Improv	1.8x	17.6x	-0.6%
B1	SVHN	ResNet18	11.2M	4400	93.4
		NIN	968K	2000	89.3
		Improv	11.6x	2.2x	-4.1%
A1/A2	CIFAR10	ResNet18	11.2M	500	45.8
		DenseNet100	769K	500	47.5
		Improv	14.6x	-	1.7%
D	CIFAR100-Task	ResNet18-S	1108K	1105	60.3
		DenseNet100	794K	1105	62.1
		Improv	1.4x	-	1.8%
B2	CIFAR100-Class	ResNet18	11.2M	2000	24.1
		DenseNet100	800K	2000	27.5
		Improv	14.6x	-	3.4%

Table 1. Minimizing the resource consumption (storage size and parameters).

1 Additional Experiments

We perform additional experiments to study variance of performance across memory sizes, models, training timesteps on each dataset. We first study the variation across models and different values of k and tabulate these results in Table 1. For MNIST and SVHN, we could reduce the replay buffer size by over 17x and 2x respectively with small impact on performance, a good tradeoff. For CIFAR10 and CIFAR100, we substitute the bulky ResNet18 model with a DenseNetBC-100-12 [1] which gives us a major decrease of over 14x in parameters. In CIFAR10 and CIFAR100 (Task-IL and Class-IL), we additionally improve accuracy by 1.7%, 1.8% and 3.4% respectively after compressing the models. On MNIST and SVHN, we could compress the size of stored dataset k , showing existing formulations use too much memory. But as dataset complexity increases, larger memory size is required. We could also compress the parameter storage by large margins while increasing accuracy by substituting popular but efficient models, showing the scope for using more efficient models for CL formulations.

We further study the trade-off between accuracy and training time of our improved models on the improved k values by varying the passes on memory.

Passes/Form.	B1-MNIST	B1-SVHN	A1-CIFAR10	D	B2	C2
k	250	2000	500	1105	2000	9000
	NIN	NIN	DenseNet100	DenseNet100	DenseNet100	DenseNet100
8	91.7 (96.0)	72.5 (81.4)	28.4 (28.4)	49.5 (50.8)	7.3 (8.8)	32.3 (33.9)
16	95.9 (96.3)	85.1 (85.6)	32.3 (33.5)	52.6 (53.0)	10.0 (11.4)	39.4 (41.9)
32	96.9 (**97.2**)	88.8 (87.3)	37.8 (36.7)	56.2 (58.3)	15.0 (18.1)	47.2 (48.4)
64	97.4 (97.2)	**89.2** (87.5)	39.9 (40.1)	60.6 (61.2)	21.9 (22.5)	54.0 (54.3)
128	97.4 (97.2)	89.2 (88.9)	43.9 (42.9)	**62.1** (61.9)	26.5 (24.1)	**56.9** (56.3)
256	96.6 (97.5)	88.4 (89.7)	46.4 (43.9)	62.0 (62.4)	**27.6** (25.7)	54.1 (54.8)
512	96.1 (97.7)	86.9 (88.4)	**47.5** (45.9)	62.3 (61.5)	27.6 (25.8)	54.0 (54.7)

Table 2. Accuracy of tweaked (NIN/DenseNet) GDumb models with number of passes. The bolded accuracies represent the reported results in previous experiments, while accuracies in (brackets) are obtained without cutmix regularization. We can halve the training time with slight tradeoff of upto 1% accuracy.

We present the results in Table 2, with **bold** being the selected models, along with (brackets) representing accuracies obtained by an ablation without cutmix regularization. We show that we can further reduce our training time by half with a minor (approx 1%) tradeoff in accuracy. We also observe that cutmix regularization improves performance by 0.2% to 1.5% margins. We strongly recommend regularization in GDumb since it only has access to the few samples in memory.

2 Experiment formulations

We first detail each of the seven selected formulation and then present results on each of them.

Formulation A1 ([2]): We benchmark on two datasets: MNIST and CIFAR10 from this setting. They randomly divided MNIST and CIFAR10 into 5 disjoint tasks of 2 classes each. The architectures used for MNIST is a MLP with 2 hidden layers of 400 nodes and ResNet18 for CIFAR10. They use a small limit (k) of 300 and 500 stored samples for MNIST and 200, 500 and 1000 stored samples for CIFAR10. In MNIST, they select 1000 samples per task, while in CIFAR10 they utilize 9,750 samples per task in the online stream. On MNIST, we additionally compare with GSS [3] who use the same setup except they have an MLP with 100 hidden size. We compare accuracy on the hold-out test set after all tasks are done.

Formulation A2 ([4]): They split MNIST and CIFAR10 into 5 disjoint subsets by their labels as different tasks. Each task consists of 1,000 online training examples in MNIST and 10,000 training examples in CIFAR10 similar to the above setup. The goal is to classify over all 10 classes on a held-out test set when the training ends. The architectures used for MNIST is a MLP with 2 hidden layers of 100 nodes and ResNet18 for CIFAR10, identical to the above setup. However, since the accuracies obtained for ER, GEM and ER-MIR are very

different, we list them as a different formulation and compare our performance against them.

Formulation A3 ([5]): They split MNIST and CIFAR10 into 5 disjoint subsets. Each task consists of 1,000 online training examples in MNIST and 9,750 training examples in CIFAR10 similar to the A1. The goal is to classify over all 10 classes on a held-out test set when the training ends. The architectures used for MNIST is a MLP with 2 hidden layers of 100 nodes and ResNet18 for CIFAR10, identical to the above setup. However, since the accuracies obtained for ER, GEM and ER-MIR are very different (for MNIST), we list them as a different formulation as merging tables is not possible.

Formulation B1 ([6,7]): It consists of two datasets: MNIST and SVHN, randomly divided into 5 disjoint tasks of 2 classes each. The architecture used is a MLP with 2 hidden layers of 400 nodes for MNIST and ResNet18 for SVHN. The formulation controls the total static memory overhead among all proposed approaches, resulting in storage capacity of 4400 for memory-based approaches [6].

Formulation B2 ([8]): The formulation use CIFAR100, split into 20 tasks of 5 classes each. They use a ResNet-32 model and a limit of 2000 stored samples (k). We additionally report the average of accuracy after each task as described in [8] referred to as accuracy (avg in t), along with accuracy after all tasks referred to as accuracy.

Formulation B3 ([9]): This formulation tests small-task increments on CIFAR100 and Imagenet100. Given a class-order, they use the first 50 tasks for pretraining and then subsequent 50 tasks with 1 class each in a CI-CL fashion including the initial 50 classes. Hence, they have a CI-CL classification with 51-100 classes. They start off with 1000 samples in memory and add 20 samples to memory for each subsequent task added. They use ResNet32 on CIFAR100 and ResNet18 on Imagenet100 to match the formulation and provide 3 class-orders for CIFAR100 and 1 class-order for Imagenet100 which we use. We measure average and last-task accuracy on using the same set of class-orders, using the same memory and networks as specified in the formulation.

Formulation C1 ([6]): It splits MNIST into 5 disjoint tasks of 2 classes each. The architecture used is a MLP with 2 hidden layers of 400 nodes for MNIST. The formulation controls the total static memory overhead, resulting in storage capacity of 4400 for memory-based approaches. We use three different class-to-task mappings to get performance.

Formulation C2 ([10]): TinyImagenet is divided into 10 disjoint tasks of 20 classes each in this formulation. We compare with the overall best accuracies obtained (Table 10 in [10]). The table lists their best performance observed over different architectures, regularization strategies, hyperparameter searches, etc. We test for two stored sample limit (k): 4500 and 9000. We use DenseNetBC-100-12 architecture as detailed in subsequent sections. Note that although it differs from the architectures tested; it is a fairly standard efficient architecture.

Formulation D ([11]): We borrow the benchmark on CIFAR100, consisting of 20 disjoint tasks of 5 classes each. We train 17 tasks, store upto 13 samples per class and use the same reduced ResNet18 architecture.

Formulation E ([3]): It consists of two datasets (MNIST and CIFAR10) with class-imbalanced, blurry boundary setting. Each has 5 tasks, each task having 2 classes each. MNIST has 2000 online samples of current task and 200 from each other tasks for every task in the formulation. Similarly, in CIFAR10 we keep 90% of the data for each task, and introduce 10% of data from the other tasks. The same architectures are used as in GSS [3], which are a 2-layer MLP with hidden size 100 for MNIST and a ResNet18 for CIFAR10. The limit on stored samples (k) is 300 for MNIST and 500 for CIFAR10.

References

1. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. (2017)
2. Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Charlin, L., Tuytelaars, T.: Online continual learning with maximally interfered retrieval. In: NeurIPS. (2019)
3. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. In: NeurIPS. (2019)
4. Jin, X., Du, J., Ren, X.: Gradient based memory editing for task-free continual learning. (2020)
5. Ji, X., Henriques, J., Tuytelaars, T., Vedaldi, A.: Automatic recall machines: Internal replay, continual learning and the brain. arXiv preprint arXiv:2006.12323 (2020)
6. Hsu, Y.C., Liu, Y.C., Kira, Z.: Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In: NeurIPS-W. (2018)
7. Rajasegaran, J., Hayat, M., Khan, S., Khan, F.S., Shao, L.: Random path selection for incremental learning. In: NeurIPS. (2019)
8. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. (2017)
9. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Small-task incremental learning. In: ECCV. (2020)
10. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: Continual learning: A comparative study on how to defy forgetting in classification tasks. arXiv preprint arXiv:1909.08383 (2019)
11. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. In: ICLR. (2019)