
On Class Orderings for Incremental Learning

Marc Masana^{*1} Bartłomiej Twardowski^{*1} Joost van de Weijer¹

Abstract

The influence of class orderings in the evaluation of incremental learning has received very little attention. In this paper, we investigate the impact of class orderings for incrementally learned classifiers. We propose a method to compute various orderings for a dataset. The orderings are derived by simulated annealing optimization from the confusion matrix and reflect different incremental learning scenarios, including maximally and minimally confusing tasks. We evaluate a wide range of state-of-the-art incremental learning methods on the proposed orderings. Results show that orderings can have a significant impact on performance and the ranking of the methods.

1. Introduction

Incremental learning (IL) has gained popularity over the last years as a way to continuously introduce new concepts to an existing model. Incrementally learning tasks relieves the issues of maintaining and retraining large datasets, and costs associated with it. However, retaining knowledge when re-training on different data in artificial neural networks is not a trivial task. Whenever a network is trained only on new data, the abrupt loss of previously acquired knowledge manifests. This phenomenon is known as *catastrophic forgetting* (McCloskey & Cohen, 1989). In recent years, many methods have been proposed to alleviate this problem which stands in the way to advanced life-long learning systems (Lesort et al., 2020; De Lange et al., 2019; Parisi et al., 2019).

The problem of continual learning is often simplified to an incremental learning of new concepts (classes) in a well-defined, equally divided, sequence of tasks. This may sound artificial, but is a common choice in recent works (Aljundi et al., 2017; Li & Hoiem, 2017; Rebuffi et al., 2017; Chaudhry et al., 2018; Belouadah & Popescu,

^{*}Equal contribution ¹LAMP team, Computer Vision Center, UAB Barcelona, Spain. Correspondence to: Marc Masana <mmasana@cvc.uab.cat>.

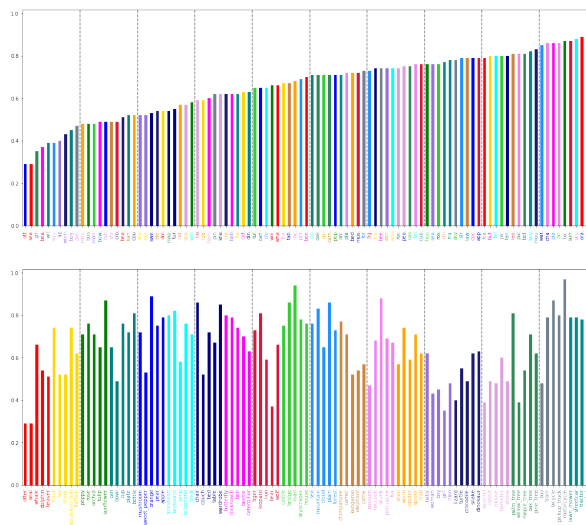


Figure 1. Accuracies of CIFAR-100 classes after a single non-incremental training on ResNet-32 incrementally ordered by class accuracy (top), and grouped with provided coarse-grained labels (bottom). Dashed lines represent task boundaries for an equally divided 10-task split. Colors denote coarse-grained group labels.

2019). Having the same experimental settings makes it easy to compare results and performance. Specifically, class-incremental learning (class-IL) defines the setting where the task-ID is unknown at inference time, making the predictions of the learned models task-agnostic. Most class-IL methods consider multi-class incremental learning (Hou et al., 2019), where only tasks that at least have two or more classes are considered. Those also conform to the prevalent practise of making all tasks have the same number of classes. This rises the question about data preparation: are we missing something important when ordering the tasks and classes? In this work we investigate how relevant is class ordering and how it can affect the results of well-known class-IL methods.

It has become common to compare different approaches by using CIFAR-100, while recently some other larger datasets have been getting more exposure. In Fig. 1, we show that class ordering can influence task accuracy within a particular split for an already trained model. Therefore, some influence can be expected to multi-class-IL. Overall performance—measured as a value of mean avg. accuracy across all tasks after reaching the last one—can depend on how hard is

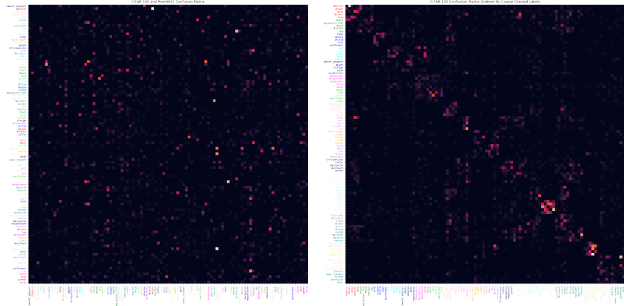


Figure 2. CM from CIFAR-100 with original class order (left) and coarse grained labels order (right) after a joint training on ResNet-32. Diagonal values are skipped for better visualization.

the ordering. In our research, we propose a method based on confusion matrix (CM) ordering (see Fig. 2) that helps explore the *difficulty* of the incrementally learned classifier even further.

The main contributions of this paper are: 1) proposing a novel method for class ordering in IL scenarios based on confusion matrix values, 2) investigating IL methods robustness to class ordering, 3) analysing of some commonly used split strategies in comparison to the random ones.

2. Related work

Class-IL: We chose some class-IL methods that are common comparison in the literature, and some that are current state-of-the-art. LwF (Li & Hoiem, 2017) is a regularization-based method which adds a constrain loss to the outputs of older classes to not change too much when learning a new task. Similarly, EWC (Kirkpatrick et al., 2017) also applies a regularization constrain on the weights to limit their shift. iCaRL (Rebuffi et al., 2017) proposed to extend LwF by keeping a small memory with data exemplars which is replayed during training. Following this idea, BiC (Wu et al., 2019) and LUCIR (Hou et al., 2019) extend the usage of distillation and exemplars to also apply a bias correction to the outputs of different tasks, allowing to compensate the imbalance introduced by new tasks. Finally, IL2M (Belouadah & Popescu, 2019) also proposes bias correction, although over Finetuning since they show that it works better than applying it over LwF.

Class ordering: Most works on class-IL report results by using a random order of classes on CIFAR-100, i.e. iCaRL, LUCIR, BiC. Furthermore, the popularity of iCaRL and the interest in comparing with it, makes quite common their specific class ordering (their code fixes the random seed to 1993). However, none of them look deeper into the choosing of that specific class order. In (Masana et al., 2020) and (De Lange et al., 2019), the authors touch the subject of class ordering, showing that some methods report

different results based on different class orderings. Only a random ordering and a semantically split ordering were investigated, without any dedicated method to order classes harder or easier, as this was not their main focus.

Curriculum Learning and Classification Complexity:

In contrast to curriculum learning, our objective is not to obtain the best model performance (Pentina et al., 2015), but to propose an evaluation for class-IL methods under different scenarios. Another similar research direction is assessing how complex is a classification task. In our case, we could use a known measure (Lorena et al., 2019), or the one proposed in (Nguyen et al., 2019) for IL.

3. Class ordering

In multi class-IL for image classification, a sequence of tasks where each consists of m_t classes is learned one at a time, extending the knowledge of the model in incremental steps. Given a set of paired data \mathbf{x}_i with their respective class labels $y_i \in C^t$, where $C^t = \{c_1^t, c_2^t, \dots, c_{m_t}^t\}$ denotes the set of m^t classes of task t . When training on task t , only data $(\mathbf{x}_i, y_i) \sim D^t$ is available. We consider disjoint classes between all tasks, $C^t \cap C^s = \emptyset$ for $t \neq s$ as in (Aljundi et al., 2017; Chaudhry et al., 2018; Dhar et al., 2019; Hou et al., 2019; Liu et al., 2018; Rebuffi et al., 2017; Yu et al., 2020). After training each task we evaluate the learned model on all classes seen so far $C = \bigcap_{i < t} C^i$.

Let $M \in \mathbb{N}^{|C| \times |C|}$ be the confusion matrix from learning all classes in a non-incremental way (usually known as *joint training*), where each element M_{ij} defines how many times samples of class i are predicted as class j . Based on the information contained in M , we can estimate how *confusing* class i is by looking at how much it gets wrongly classified as any another class j . Consequently, also how easy or difficult they are to tell apart when having all data available during the training session. We assume that any advantage that comes by having all information available can be mitigated or removed when changing to an IL setting. In joint training, specific features in the network can be learned to focus on differentiating two classes that can easily be confused. However, in an IL setting, those discriminative features become more difficult to learn or can be modified afterwards, especially when the classes belong to different tasks. Thus, the difficulty of the task can be perceived differently in each scenario. Depending on the method, it may handle this issue differently, and therefore lead to more catastrophic forgetting.

As a result, we define different class orderings. In order to illustrate the effect of class ordering, we take the CIFAR-100 dataset (Krizhevsky, 2009) as an example, although the proposed strategies are extensible to other datasets. First, we define the baseline class ordering as:

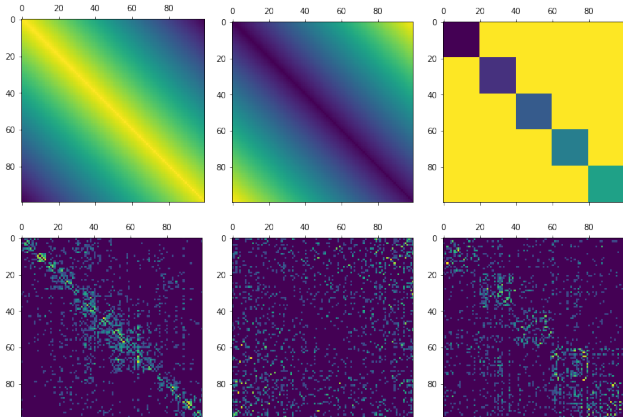


Figure 3. Class ordering objectives (top) and resulting confusion matrices (bottom).

- **random**: takes a permutation of the class order. By default taking the original class ordering which the dataset provides (usually alphabetically ordered or similar), or otherwise a permutation corresponding to a random seed. As explained in Sec. 2, in the case of CIFAR-100, some works decide to fix the seed to the same as iCaRL (Rebuffi et al., 2017).

If we train a model in a single training session (joint training) with all data for all classes, we can calculate CM, as seen on Fig. 2. Based on that, we define two more class orders as:

- **max confusion** (maxConf): highly miss-classified classes are next to each other—max confusion is happening around the CM diagonal. This creates an IL split with more difficult intra-task classification.
- **min confusion** (minConf): enforces classes which are rarely miss-classified to be in the same task—max confusion is happening at the corners of the CM. Intra-task and adjacency tasks classification becomes easier, but also pushes the most miss-classified classes towards the first and last tasks.

Finding the above orderings based on the CM values is a non-trivial task, where a brute-force naive approach of $O(n!)$ cannot be applied even to moderate size problems. To reduce the complexity, we re-formulate finding the class ordering as an optimization problem where the objective is to maximize the value of the fitness function to a desired M . Then, we use an objective weight matrix $W \in \mathbb{N}^{|C| \times |C|}$ (see Fig. 3) which is used to calculate a score value:

$$\text{score}(M, W) = \text{tr}(W^T M), \quad (1)$$

to assess adaptation in a search for a solution. In this case, a global optimum is not necessary since it is hard to establish. Instead, we use the simulated annealing optimization algorithm to find an ordering in a constrained time limited

by the number of fitting iterations. The same approach has been previously used for CM ordering for better visualization of large matrices in (Thoma, 2017a) and for HASYv2 dataset (Thoma, 2017b).

We can also define an objective CM, and try to converge to a permutation that better accommodates to incremental tasks by introducing their boundaries. If the splits of C^1, \dots, C^t can be know upfront, which is usually the case, we can incorporate task boundaries in the weighting matrix for both scenarios. Therefore, we can define three more orderings:

- **increasing task confusion** (incTaskConf): maximize confusion in all tasks around the diagonal of M with increasing confusion between them. The objective matrix is presented in Fig. 3 (right).
- **equal task confusion** (eqTaskConf): similar to max confusion, but introducing task boundaries should cause less confusion between adjacent tasks.
- **decreasing task confusion** (decTaskConf): maximize confusion in all tasks around the diagonal while decreasing it between them. Similar to increasing, but with inverted diagonal weights from Fig. 3 (right).

In the specific case of CIFAR-100, since a coarse grained hierarchy of the classes exists, we can also define an ordering based on the provided two level taxonomy. In each task we can have classes related to the same group or similar groups in order to make the classification harder.

- **coarse grained**: ordered by a provided grouping or taxonomy. For CIFAR-100 classes are divided into 20 groups, as shown with bar colors in Fig. 1 and labels in Fig. 2 (right).

4. Experimental results

We compare the class orderings on CIFAR-100 considering ten equal tasks trained on ResNet-32 from scratch. Training starts with learning rate (LR) of 0.1, momentum of 0.9, weight decay of $2e-4$, and a LR scheduler with ten epochs patience and a LR factor of $1/3$ until LR is lower than $1e-4$ or 200 epochs have passed. Method hyperparameter are chosen following the framework from (De Lange et al., 2019) for the first 3 tasks, and fixed afterwards.

We compare all orderings proposed in Sec. 3 on Finetuning (FT) and LwF in Fig. 4. Both methods seem to have very little difference in general behaviour on the different orderings. Random, iCaRL seed and minConf provide a better performance after all tasks. The results point to minConf being the most stable, and Random being generally closer to it. After the first task, incTaskConf has the highest performance since it learns the less confusing group of classes. However, after all tasks, it ends up having one of the lowest performances, together with maxConf.

Table 1. CIFAR-100 results for class-IL with growing memory of 20 exemplars per class (10 runs average and standard deviation). The best score for each task of each method is in bold. Underscore marks the lowest score.

	task	Random	iCaRL seed	coarse	maxConf	minConf	decTaskConf	eqTaskConf	incTaskConf
LwF	2	51.8±5.3	50.8±2.6	45.9±2.7	52.4±2.0	44.6±1.9	<u>41.5</u> ±3.2	53.1±3.9	62.5 ±2.7
	5	<u>31.8</u> ±3.9	33.2±3.3	36.3±2.2	33.0±2.9	33.8±2.3	33.9±2.6	35.9±3.2	37.4 ±2.9
	10	<u>24.8</u> ±2.6	27.3±2.2	26.5±2.0	32.6 ±3.1	25.4±1.4	29.9±2.4	31.6±3.3	29.4±1.9
iCaRL	2	61.4±3.8	58.7±3.6	49.9±3.1	55.1±1.6	55.5±1.3	<u>44.9</u> ±2.4	60.0±2.7	67.2 ±3.9
	5	42.0±3.6	42.5±2.7	41.7±2.2	39.6±2.3	43.7±2.4	<u>33.4</u> ±2.6	44.5±2.4	45.9 ±3.6
	10	33.8±3.7	34.2±2.6	32.8±1.7	35.4 ±2.6	33.4±2.3	<u>28.0</u> ±2.2	35.4±3.0	32.0±3.1
BiC	2	61.2±5.3	56.4±3.8	50.4±3.0	52.8±2.6	51.5±2.7	41.2±2.1	57.9±4.1	69.7 ±2.7
	5	44.8±3.2	45.2±3.0	44.4±2.6	40.7±3.7	47.7±1.3	<u>37.1</u> ±2.9	45.3±3.1	52.4 ±2.8
	10	39.3±1.9	40.1±2.8	39.3±2.3	37.5±2.7	40.1 ±1.3	<u>37.2</u> ±2.9	38.6±2.0	37.8±2.6
LUCIR	2	63.2±3.4	59.6±3.3	53.4±2.9	54.0±3.2	53.3±3.3	<u>47.2</u> ±1.9	61.2±2.0	72.0 ±2.2
	5	40.0±3.4	40.7±3.5	40.3±2.6	37.2±5.6	39.5±2.8	<u>36.5</u> ±2.9	45.8±1.4	47.3 ±1.9
	10	<u>27.2</u> ±2.7	29.6±3.2	<u>26.2</u> ±3.1	28.9±5.8	27.7±1.9	29.1±3.3	31.9 ±2.0	28.5±1.6
IL2M	2	58.2±5.1	52.2±3.7	43.2±5.9	51.2±1.1	51.1±3.0	<u>42.1</u> ±2.7	54.4±2.7	65.3 ±2.6
	5	44.2±4.1	41.0±3.7	44.0±2.4	40.6±2.6	47.5±1.7	<u>37.1</u> ±3.0	43.6±2.6	51.0 ±2.4
	10	38.2±2.0	37.9±2.0	38.6 ±2.1	38.5±2.7	<u>36.8</u> ±1.7	37.4±2.4	38.3±2.3	37.6±2.3

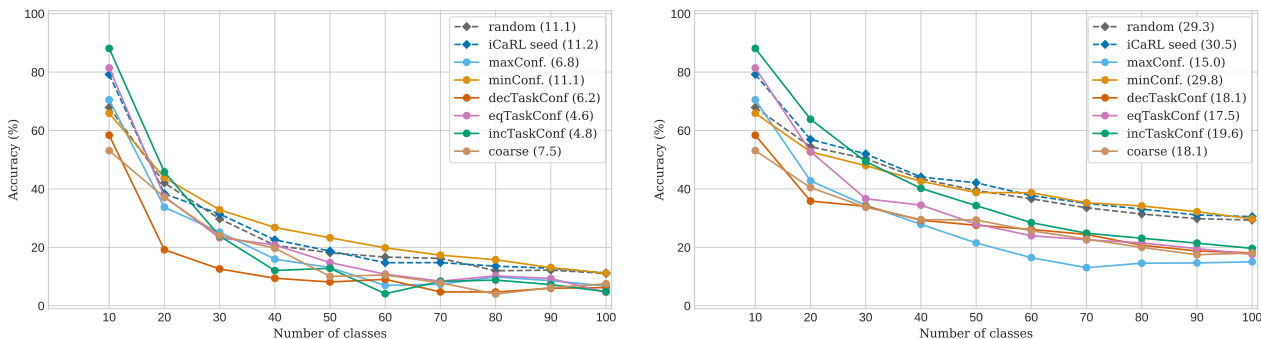


Figure 4. FT (left) and LwF (right) with different class orderings for CIFAR-100 on ResNet-32 from scratch without exemplars memory.

Results on different methods are presented in Tab. 1, using 20 exemplars per class with herding selection – LwF is adapted to use exemplars. As expected, decTaskConf results in the *most confusing* class ordering for the first tasks with the lowest performance. Analogously, incTaskConf achieves the best performance. This is due to not having learned all classes at this point but only the most or least confusing, respectively. This behaviour is different than the one seen in the setting without exemplars, where incTaskConf is only better until task 2. LwF and LUCIR have a similar task 10 overall performance (avg. 28%), iCaRL follows (avg. 33%), while BiC and IL2M have a better one (avg. 38%). In addition, the standard deviation across all orderings is low for BiC and IL2M (~2.3). Next comes iCaRL and LUCIR with a bit larger deviation (~3.4), and LwF being the least robust (~3.6). Interestingly, looking at the best performance at task 10 for each method individually, each of them does well at a different class ordering, thus changing the optic of the result and the ranking of the methods.

5. Conclusions

Class orderings for class-IL influence the overall evaluation performance. For a single method the spread between most extreme orderings can be significant. Comparing the orderings, we found that the random ordering obtains among the highest performances when used by non-exemplar methods. The proposed class orderings based on the confusion matrix can be used as a tool for checking robustness of class-IL approaches. A direct extension of this work would be to use different datasets and ordering methods. For a fairer comparison of methods, we recommend to compare methods on several class orderings.

Acknowledgements

We would like to thank Xialei Liu for his helpful discussion. Marc Masana acknowledges 2019-FI_B2-00189 grant from Generalitat de Catalunya.

References

- Aljundi, R., Chakravarty, P., and Tuytelaars, T. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- Belouadah, E. and Popescu, A. I12m: Class incremental learning with dual memory. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 583–592, 2019.
- Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018.
- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.
- Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5138–5146, 2019.
- Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 831–839, 2019.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.
- Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A. M., and Bagdanov, A. D. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *International Conference on Pattern Recognition (ICPR)*, 2018.
- Lorena, A., Garcia, L. P., Lehmann, J., de Souto, M., and Ho, T. How complex is your classification problem?: A survey on measuring classification complexity. *ACM Computing Surveys*, 52:1–34, 09 2019. doi: 10.1145/3347711.
- Masana, M., Tuytelaars, T., and van de Weijer, J. Ternary feature masks: continual learning without any forgetting. *arXiv preprint arXiv:2001.08714*, 2020.
- McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.
- Nguyen, C. V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., and Soatto, S. Toward understanding catastrophic forgetting in continual learning, 2019.
- Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.
- Pentina, A., Sharmanska, V., and Lampert, C. H. Curriculum learning of multiple tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.
- Thoma, M. Analysis and optimization of convolutional neural network architectures, 2017a.
- Thoma, M. The hasyv2 dataset, 2017b.
- Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. Large scale incremental learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 374–382, 2019.
- Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S., and Weijer, J. v. d. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6982–6991, 2020.

Appendices

A. Plots for growing memory setting

In this appendix we present plots that reflect evaluation of each method from Table 1 for different class orderings after learning each task.

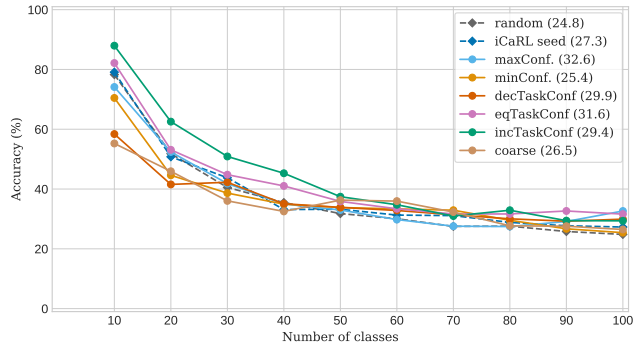


Figure 5. LwF results for different class orderings for CIFAR-100 on ResNet-32 from scratch with 20 exemplars per class growing memory.

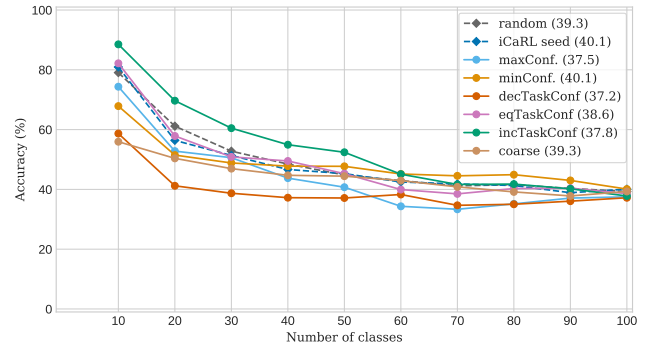


Figure 7. BiC results for different class orderings for CIFAR-100 on ResNet-32 from scratch with 20 exemplars per class growing memory.

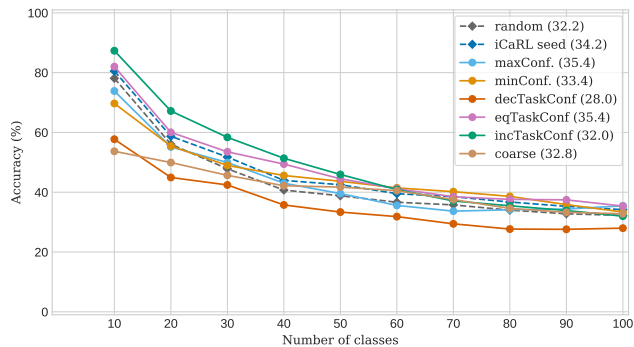


Figure 6. iCaRL results for different class orderings for CIFAR-100 on ResNet-32 from scratch with 20 exemplars per class growing memory.

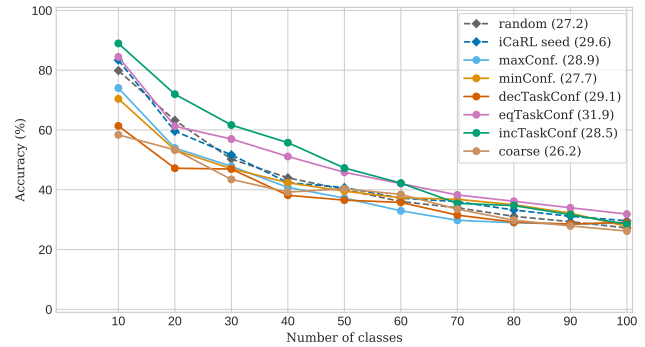


Figure 8. LUCIR results for different class orderings for CIFAR-100 on ResNet-32 from scratch with 20 exemplars per class growing memory.

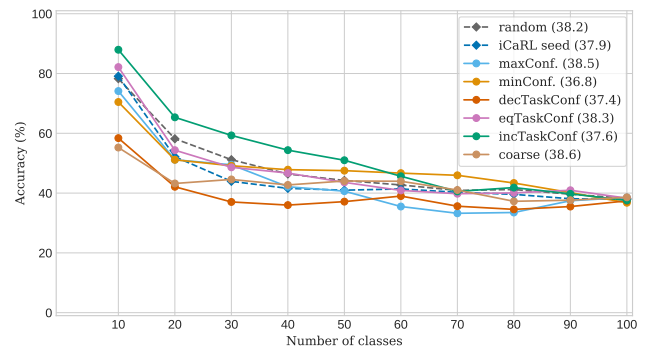


Figure 9. IL2M results for different class orderings for CIFAR-100 on ResNet-32 from scratch with 20 exemplars per class growing memory.