PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning

Arthur Douillard[1,2], Matthieu Cord[2,3], Charles Ollion[1], Thomas Robert[1], and Eduardo Valle[4]

[1] Heuritech, Paris, France
{arthur.douillard,thomas.robert,charles.ollion}@heuritech.com
[2] Sorbonne University, Paris, France
matthieu.cord@sorbonne-universite.fr
[3] valeo.ai, Paris, France
[4] University of Campinas, Campinas, Brazil
dovalle@dca.fee.unicamp.br

Abstract. Lifelong learning has attracted much attention, but existing works still struggle to fight catastrophic forgetting and accumulate knowledge over long stretches of incremental learning. In this work, we propose PODNet, a model inspired by representation learning. By carefully balancing the compromise between remembering the old classes and learning new ones, PODNet fights catastrophic forgetting, even over very long runs of small incremental tasks – a setting so far unexplored by current works. PODNet innovates on existing art with an efficient spatial-based distillation-loss applied throughout the model and a representation comprising multiple proxy vectors for each class. We validate those innovations thoroughly, comparing PODNet with three state-of-the-art models on three datasets: CIFAR100, ImageNet100, and ImageNet1000. Our results showcase a significant advantage of PODNet over existing art, with accuracy gains of 12.10, 6.51, and 2.85 percentage points, respectively.[5]

Keywords: incremental-learning, representation-learning pooling

1 Introduction

Lifelong machine learning [34,8,37] focuses on models that accumulate and refine knowledge over large timespans. Incremental learning – the ability to aggregate different learning objectives seen over time into a coherent whole – is paramount to those models. To achieve incremental learning, models must fight *catastrophic forgetting* [34,8] of previous knowledge. Lifelong and incremental learning have attracted much attention in the past few years, but existing works still struggle to preserve acquired knowledge over many cycles of short incremental learning steps.

---

[5] Code is available at: github.com/arthurdouillard/incremental_learning.pytorch

We will focus on image classifiers, which are ordinarily trained once on a fixed set of classes. In *incremental learning*, however, the classifier must learn the classes by steps, in training cycles called *tasks*. At each task, we expose the classifier to a new set of classes. Incremental learning would reduce trivially to ordinary classification if we were allowed to store all training samples, but we are imposed a limited *memory*: a maximum number of samples for previously learned classes. This limitation is motivated by practical applications, in which privacy issues, or storage and computing limitations prevent us from simply retraining the entire model for each new task [23,24]. Furthermore, incremental learning is different from transfer learning in that we aim to have good performance in both old and new classes.

To overcome catastrophic forgetting, different approaches have been proposed: reusing a limited amount of previous training data [33,3]; learning to generate the training data [17,36]; extending the architecture for new phases of data [39,22]; using a sub-network for each phase [7,11]; or constraining the model divergence as it evolves [18,25,1,23,33,3].

In this work, we propose PODNet, approaching incremental learning as representation learning, with a distillation loss that constrains the evolution of the representation. By carefully balancing the compromise between remembering the old classes and learning new ones, we learn a representation that fights catastrophic forgetting, remaining stable over long runs of small incremental tasks. Our model innovates on existing art with (1) an *efficient spatial-based* distillation-loss applied *throughout the model*; and (2) as a refinement, a representation comprising multiple proxy vectors for each class, resulting in a more flexible representation.

In this paper, we first present the existing state of the art (Sec. 2), which we close by detailing our contributions. We then describe our model (Sec. 3), and evaluate it in an extensive set of experiments (Sec. 4) on CIFAR100, ImageNet100, and ImageNet1000, including ablation studies assessing each contribution, and extensive comparisons with existing methods.

2   Related Work

To approach the problem of incremental learning, consider a single incremental task: one has a classifier already trained over a set of old classes and must adapt it to learn a set of new classes. To perform that single task, we will consider: (1) the data/class representation model; (2) the set of constraints to prevent catastrophic forgetting; (3) the experimental context (including the constraints over the memory for previous training data) for which to design the model.

Data/class representation model. Representation learning was already implicitly present in iCaRL [33]: it introduced the Nearest Mean Exemplars (NME) strategy which averages the outputs of the deep convolutional network to create a single proxy feature vector per class that are then used by a nearest-neighbor classifier predict the final classes. Hou et al. [14] adopted this method and also introduced another, named CNN, which uses the output class probabilities to

classify incoming samples, freezing (during training) the classifier weights associated with old classes, and then fine-tuning them on an under-sampled dataset.

Hou et al. [14], in the method called here UCIR, made representation learning explicit, by noticing that the limited memory imposed a severe imbalance on the training samples available for the old and for the new classes. To overcome that difficulty, they designed a metric-learning model instead of a classification model. That strategy is often used in few-shot learning [9] because of its robustness to few data. Because classical metric architectures require special training sampling (e.g., semi-hard sampling for triplets), Hou et al. chose instead to redesign the classifier's last layer of their model to use the cosine similarity [27].

Model constraints to prevent catastrophic forgetting. Constraining the model's evolution to prevent forgetting is a fruitful idea proposed by several methods [18,25,1,23,33,3]. Preventing the model's parameters from diverging too much forces it to remember the old classes, but care must be taken to still allow it to learn the new ones. We call this balance the *rigidity-plasticity trade-off*.

Existing art on knowledge distillation/compression [13] was an important source of inspiration for constraints on models. The goal is to distill a large trained model (called teacher) into a new smaller model (called student). The distillation loss forces the features of the student to approach those of its teacher. In our case, the student is the current model and the teacher—with same capacity– is its version at the previous task. Zagoruyko and Komodakis [19] investigated attention-based distillation for image classifiers, by pooling the intermediate features of convolutional networks into attention maps, then used in their distillation losses. Li and Hoiem [23] - and several authors after them [33,3,38] - used a binary cross-entropy between the output probabilities by the models. Hou et al. [14], used instead *Less-Forget*, a cosine-similarity constraint on the flat feature embeddings after the global average pooling. Dhar et al. [5] proposed to constrain the gradient-based attentions generated by GradCam [35], a visualization method. Wu et al. [38] proposed BiC, an algorithm oriented towards large-scale datasets, which employs a small linear model learned on validation data to recalibrate the output probabilities before applying a distillation loss.

Experimental context. A critical component of incremental learning is the convention used for the memory storing samples of previous data. An usual convention is to consider a fixed amount of samples allowed in that memory, as illustrated in Fig. 1.

Still, there are two experimental protocols for such fixed-sample convention: we may either use the memory budget at will (M_{total}), or add a constraint on the number of samples per class for the old classes (M_{per}). When $M_{\text{total}} = M_{\text{per}} \times \#$ *of classes*, both settings have equivalent *final* memory size, but the latter, that we adopt, is much more challenging since early tasks cannot benefit from the full memory size. *The granularity of the increments* is another critical element: with a fixed number of classes, increasing the number of tasks decreases the number of classes per task. More tasks imply stronger forgetting of the earliest classes, and pushing that number creates a challenging protocol, so far unexplored by
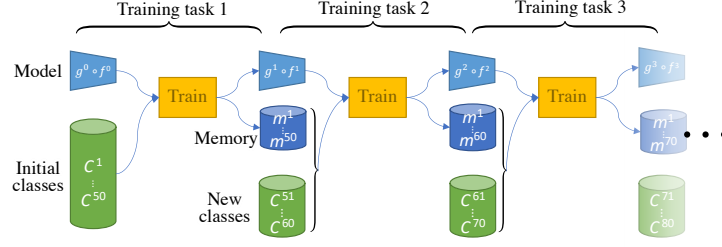
Fig. 1. Training protocol for incremental learning. At each training task we learn a new set of classes, and the model must retain knowledge about *all* classes. The model is allowed a *limited* memory of samples of old classes.

existing art. Hou et al. evaluate at most 10 tasks on CIFAR100, while we propose as much as 50 tasks.

Finally, to score the experiments, Rebuffi et al. [33] proposed a global metric that they called **average incremental accuracy**, taking into account the entire history of the run, averaging the accuracy at the end of each task (including the first).

Contributions. As seen, associating representation learning to model constraints is a particularly fruitful idea for incremental learning, but requires carefully balancing the goals of rigidity (to avoid catastrophic forgetting) and plasticity (to learn new classes).

Employing a distillation-based loss to constrain the evolution of the representation has also resulted in leading results [14,38,40,5]. Our model improves existing art by employing a *novel and efficient spatial-based* distillation loss, which we are able to apply *throughout the model*.

Implicit or explicit proxy vectors representing each class inside the models have lead to state of the art results [33,14]. Our model extends that idea allowing for *multiple proxy vectors* per class, resulting in a more flexible representation.

3   Model

Formally, we learn the model in T *tasks*, task t comprising a set of new classes C_N^t, and a set of old classes C_O^t, and aiming at classifying all seen classes $C_O^t \cup C_N^t$. Between tasks, the new set C_O^t will be set to $C_O^{t-1} \cup C_N^{t-1}$, but the amount of training samples from C_O^t (called *memory*) is constrained to exactly M_{per} samples per class, while all training samples in the dataset are allowed for the classes in C_N^t, as shown in Fig. 1. The resulting imbalance, if unmanaged, leads to *catastrophic forgetting* [34,8], i.e., learning the new classes at the cost of forgetting the old ones.

Our base model is a deep convolutional network $\hat{\mathbf{y}} = g(f(\mathbf{x}))$, where $\mathbf{x}$ is the input image, $\mathbf{y}$ is the output vector of class probabilities, $\mathbf{h} = f(\mathbf{x})$ is the "feature
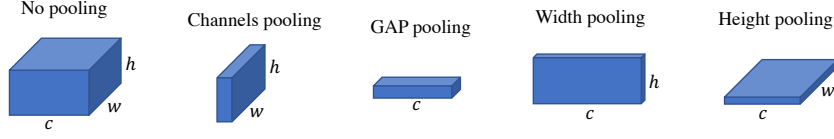
Fig. 2. Different possible poolings. The output from a convolutional layer $\mathbf{h}^t_{\ell,c,w,h}$ may be pooled (summed over) one or more axes. The resulting loss considers only the pooled activations instead of the individual components, allowing more plasticity across the pooled axes.

extraction" part of the network (all layers up to the next-to-last), $\hat{\mathbf{y}} = g(\mathbf{h})$ is the final classification layer, and $\mathbf{h}$ is the final embedding of the network before classification (Fig. 3). The superscript t denotes the model learned at task $t:f^t$, g^t, $\mathbf{h}^t$, etc.

3.1   POD: Pooled Outputs Distillation loss

Constraining the evolution of the weights is crucial to reduce forgetting. Each new task t learns a new (student) model, whose weights are not only initialized with those of the previous (teacher) model, but also constrained by a distillation loss. That loss must be carefully balanced to prevent forgetting (rigidity), while allowing the learning of new classes (plasticity).

To this goal, we propose a set of constraints we call **Pooled Outputs Distillation (POD)**, applied not only over the final embedding output by $\mathbf{h}^t = f^t(\mathbf{x})$, but also over the output of its intermediate layers $\mathbf{h}^t_\ell = f^t_\ell(\mathbf{x})$ (where by notation overloading $f^t_\ell(\mathbf{x}) \equiv f^t_\ell \circ \ldots \circ f^t_1(\mathbf{x})$, and thus $f^t(\mathbf{x}) \equiv f^t_L \ldots \circ f^t_\ell \circ \ldots f^t_1(\mathbf{x})$).

The convolutional layers of the network output tensors $\mathbf{h}^t_\ell$ with components $\mathbf{h}^t_{\ell,c,w,h}$, where c stands for channel (filter), and $w \times h$ for column and row of the spatial coordinates. The loss used by POD may pool (sum over) one or several of those indexes, more aggressive poolings (Fig. 2) providing more freedom, and thus, plasticity: the lowest possible plasticity imposes an exact similarity between the previous and current model while higher plasticity relaxes the similarity definition.

Pooling is an important operation in Computer Vision, with a strong theoretical motivation. In the past, pooling has been introduced to obtain invariant representations [26,21]. Here, the justification is similar, but the goal is different: as we will see, the pooled indexes are aggregated in the proposed loss, allowing *plasticity*. Instead of the model acquiring invariance to the input image, the desired loss acquires invariance to model evolution, and thus, representation. The proposed pooling-based formalism has two advantages: first, it organizes disparately proposed distillation losses into a neat, general formalism. Second, as we will see, it allowed us to propose novel distillation losses, with better plasticity-rigidity compromises. Those topics are explored next.

Pooling of convolutional outputs. As explained before, POD constrains the output of each intermediate convolutional layer $\mathbf{h}_{\ell,c,w,h}^t = f_{\ell}^t(\cdot)$ (in practice, each stage of a ResNet [12]). As a reminder, c is the channel and $w \times h$ are the spatial coordinates. All POD variants use the Euclidean distance of ℓ^2-normalize tensors, here noted as $\|\cdot - \cdot\|$. They differ on the type of pooling applied before that distance is computed. On one extreme, one can apply no pooling at all, resulting in the most strict loss, the most rigid constrains, and the lowest plasticity:

$$\mathcal{L}_{\text{POD-pixel}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) = \sum_{c=1}^{C} \sum_{w=1}^{W} \sum_{h=1}^{H} \left\| \mathbf{h}_{\ell,c,w,h}^{t-1} - \mathbf{h}_{\ell,c,w,h}^t \right\|^2 . \tag{1}$$

By pooling the channels, one preserves only the spatial coordinates, resulting in a more permissive loss, allowing the activations to reorganize across the channels, but penalizing global changes of those activations across the space,

$$\mathcal{L}_{\text{POD-channel}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) = \sum_{w=1}^{W} \sum_{h=1}^{H} \left\| \sum_{c=1}^{C} \mathbf{h}_{\ell,c,w,h}^{t-1} - \sum_{c=1}^{C} \mathbf{h}_{\ell,c,w,h}^t \right\|^2 ; \tag{2}$$

or, contrarily, by pooling the space (equivalent, up to a factor, to a Global Average Pooling), one preserves *only* the channels:

$$\mathcal{L}_{\text{POD-gap}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) = \sum_{c=1}^{C} \left\| \sum_{w=1}^{W} \sum_{h=1}^{H} \mathbf{h}_{\ell,c,w,h}^{t-1} - \sum_{w=1}^{W} \sum_{h=1}^{H} \mathbf{h}_{\ell,c,w,h}^t \right\|^2 . \tag{3}$$

Note that the only difference between the variants is in the position of the summation. For example, contrast equations Equation 1 and 2: in the former the differences are computed between activation pixels, and then totaled; in the latter, first the channel axis is flattened, then the differences are computed, resulting in a more permissive loss.

We can trade a little plasticity for rigidity, with less aggressive pooling by aggregating statistics across just one of the spatial dimensions:

$$\mathcal{L}_{\text{POD-width}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) = \sum_{c=1}^{C} \sum_{h=1}^{H} \left\| \sum_{w=1}^{W} \mathbf{h}_{\ell,c,w,h}^{t-1} - \sum_{w=1}^{W} \mathbf{h}_{\ell,c,w,h}^t \right\|^2 ; \tag{4}$$

or, likewise, for the vertical dimension, resulting in POD-height. Each of those variants measure the distribution of activation pixels across their respective axis. These two complementary intermediate statistics can be further combined together:

$$\mathcal{L}_{\text{POD-spatial}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) = \mathcal{L}_{\text{POD-width}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) + \mathcal{L}_{\text{POD-height}}(\mathbf{h}_{\ell}^{t-1}, \mathbf{h}_{\ell}^t) . \tag{5}$$

$\mathcal{L}_{\text{POD-spatial}}$ is minimal when the average statistics over the dataset, on both width and height axes, are similar for the previous and current model. It brings the right balance between being too rigid (Equation 1) and being too permissive (Equation 2 and 3).

Constraining the final embedding. After the convolutional layers, the network, by design, flattens the spatial coordinates, and the formalism above needs adjustment, as a summation over w and h is no longer possible. Instead, we set a flat constraint on the final embedding $\mathbf{h}^t = f^t(\mathbf{x})$:

$$\mathcal{L}_{\text{POD-flat}}(\mathbf{h}^{t-1}, \mathbf{h}^t) = \left\| \mathbf{h}^{t-1} - \mathbf{h}^t \right\|^2 . \tag{6}$$

Combining the losses, analysis. The final POD loss combines the two components:

$$\mathcal{L}_{\text{POD-final}}(\mathbf{x}) = \frac{\lambda_c}{L-1} \sum_{\ell=1}^{L-1} \mathcal{L}_{\text{POD-spatial}} \left(f_\ell^{t-1}(\mathbf{x}), f_\ell^t(\mathbf{x}) \right) + \\ \lambda_f \mathcal{L}_{\text{POD-flat}} \left(f^{t-1}(\mathbf{x}), f^t(\mathbf{x}) \right) . \tag{7}$$

The hyperparameters λ_c and λ_f are necessary to balance the two terms, due to the different nature of the intermediate outputs (spatial and flat).

As mentioned, the strategy above generalizes disparate propositions existing both in the literature of incremental learning, and elsewhere. When $\lambda_c = 0$, it reduces to the cosine constraint of *Less-Forget*, proposed by Hou et al. for incremental learning, which constrains only the final embedding [14]. When $\lambda_f = 0$ and POD-spatial is replaced by POD-pixel, it suggests the Perceptual Features loss, proposed for style transfer [16]. When $\lambda_f = 0$ and POD-spatial is replaced by POD-channel, the strategy hints at the loss proposed by Komodakis et al. [19] to allow distillation across different networks, a situation in which the channel pooling responds to the very practical need to allow the comparison of architectures with different number of channels.

As we will see in our evaluations of pooling strategies (Sec. 4.2), what proved optimal was a completely novel idea, POD-spatial, combining two poolings, each of which flattens one of the spatial coordinates. That relatively rigid strategy (channels and one of the spatial coordinates are considered in each half of the loss) makes intuitive sense in our context, which is *small-task* incremental learning, and thus where we expect a slow drift of the model across a single task.

3.2 Local Similarity Classifier

Hou et al. [14] observed that the class imbalance of incremental learning have concrete manifestations on the parameters of the final layer on classifiers, namely the weights for the over-represented (new) classes becoming much larger than those for the underrepresented (old) classes. To overcome this issue, their method (called here UCIR) ℓ^2-normalizes both the weights and the activations, which corresponds to taking the cosine similarity instead of the dot product. For each class c, their last layer becomes

$$\hat{\mathbf{y}}_c = \frac{\exp\left(\eta \langle \boldsymbol{\theta}_c, \mathbf{h} \rangle \right)}{\sum_i \exp\left(\eta \langle \boldsymbol{\theta}_i, \mathbf{h} \rangle \right)} , \tag{8}$$
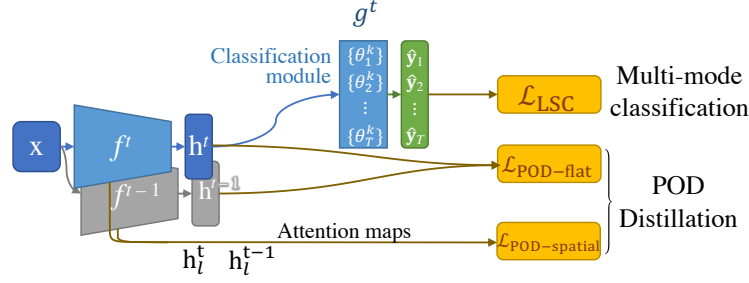
Fig. 3. Overview of PODNet: the distillation loss POD prevent excessive model drift by constraining intermediate outputs of the ConvNet f and the LSC classifier g learns a more expressive multi-modal representation.

where $\boldsymbol{\theta}_c$ are the last-layer weights for class c, η is a learned scaling parameter, and $\langle \cdot, \cdot \rangle$ is the cosine similarity.

However, this strategy optimizes a *global similarity*: its training objective increases the similarity between the extracted features and their associated weights. For each class, the normalized weight vector acts as a *single* proxy [28], towards which the learning procedure pushes all samples in the class.

We observed that such global strategy is hard to optimize in an incremental setting. To avoid forgetting, the distillation losses (Sec. 3.1) tries to keep the final embedding $\mathbf{h}$ consistent through time so that the class proxies stay relevant for the classifier. Unfortunately catastrophic forgetting, while alleviated by current methods, is not solved and thus the distribution of $\mathbf{h}$ may change. The cosine classifier is very sensitive to those changes as it models a unique majority mode through its class proxies.

Local Similarity Classifier. The problem above lead us to amend the classification layer during training, in order to consider multiple proxies/modes per class. A shift in the distribution of $\mathbf{h}$ will have less impact on the classifier as more modes are covered.

Our redesigned classification layer, which we call Local Similarity Classifier (LSC), allows for K multiple proxies/modes during training. Like before, the proxies are a way to interpret the weight vector in the cosine similarity, thus we allow for K vectors $\boldsymbol{\theta}_{c,k}$ for each class c. The similarity $s_{c,k}$ to each proxy/mode is first computed. An averaged class similarity $\hat{\mathbf{y}}_c$ is the output of the classification layer:

$$s_{c,k} = \frac{\exp \langle \boldsymbol{\theta}_{c,k}, \mathbf{h} \rangle}{\sum_i \exp \langle \boldsymbol{\theta}_{c,i}, \mathbf{h} \rangle}, \qquad \hat{\mathbf{y}}_c = \sum_k s_{c,k} \langle \boldsymbol{\theta}_{c,k}, \mathbf{h} \rangle. \tag{9}$$

The multi-proxies classifier optimizes the similarity of each sample to its ground truth class representation and minimizes all others. A simple cross-entropy loss would work, but we found empirically that the NCA loss [10,28] converged faster. We added to the original loss a hinge $[\,\cdot\,]_+$ to keep it bounded, and a small margin

δ to enforce stronger class separation, resulting in the final formulation:

$$\mathcal{L}_{\text{LSC}} = \left[-\log \frac{\exp\left(\eta(\hat{\mathbf{y}}_y - \delta)\right)}{\sum_{i \neq y} \exp \eta \hat{\mathbf{y}}_i} \right]_+ . \qquad (10)$$

Weight initialization for new classes. The incremental learning setting imposes detecting new classes at each new task t. New weights $\{\boldsymbol{\theta}_{c,k} \mid \forall c \in C_N^t, \forall k \in 1...K\}$ must be added to predict them. We could initialize them randomly, but the class-agnostic features of the ConvNet f, extracted by the model trained so far offer a better prior. Thus, we employ a generalization of Imprinted Weights [31] procedure to multiple modes: for each new class c, we extract the features of its training samples, use a k-means algorithm to split them into K clusters, and use the centroids of those clusters as initial values for $\boldsymbol{\theta}_{c,k}$. This procedure ensures mode diversity at the beginning of a new task and resulted in a one percentage point improvement on CIFAR100 [20].

3.3   Complete model formulation

Our model has the classical structure of a convolutional network $f(\cdot)$ acting as a features extractor, and a classifier $g(\cdot)$ producing a score per class. We introduced two innovations to this model: (1) our main contribution is a novel distillation loss (POD) applied all over the ConvNet, from the spatial features $\mathbf{h}_\ell$ to the final flat embedding $\mathbf{h}$; (2) as further refinement we propose that the classifier learns a multi-modal representation that explicitly keeps multiple proxy vectors per class, increasing the model expressiveness and thus making it less sensible to shift in the distribution of $\mathbf{h}$. The final loss for current model $g^t \circ f^t$, i.e., the model trained for task t, is simply their addition $\mathcal{L}_{\{f^t;g^t\}} = \mathcal{L}_{\text{LSC}} + \mathcal{L}_{\text{POD-final}}$.

4   Experiments

We compare our technique (PODNet) with three state-of-the-art models. Those models are particularly comparable to ours since they all employ a sample memory with a fixed capacity. Both iCaRL [33] and UCIR [14] use the same inference method – *Nearest-Mean-Examplars* (NME), although UCIR also proposes a second inference method based on the classifier probabilities (called here UCIR-CNN). We evaluate PODNet with both inference methods for a small scale dataset, and the later for larger scale datasets. BiC [38], while not focused on representation learning, is specially designed to be effective on large scale datasets, and thus provided an interesting baseline.

Datasets. We employ three images datasets – extensively used in the literature of incremental learning – for our experiments: CIFAR100 [20], ImageNet100 [4,14,38], and ImageNet1000 [4]. ImageNet100 is a subset of ImageNet1000 with only 100 classes, randomly sampled from the original 1000.

Table 1. Average incremental accuracy for PODNet *vs.* state of the art. We run experiments three times (random class orders) on CIFAR100 and report averages ± standard deviations. Models with an asterisk * are reported directly from Hou et al [14]

	CIFAR100			
	50 steps	25 steps	10 steps	5 steps
New classes per step	1	2	5	10
iCaRL * [33]	—	—	52.57	57.17
iCaRL	44.20 ± 0.98	50.60 ± 1.06	53.78 ± 1.16	58.08 ± 0.59
BiC [38]	47.09 ± 1.48	48.96 ± 1.03	53.21 ± 1.01	56.86 ± 0.46
UCIR (NME) * [14]	—	—	60.12	63.12
UCIR (NME) [14]	48.57 ± 0.37	56.82 ± 0.19	60.83 ± 0.70	63.63 ± 0.87
UCIR (CNN) * [14]	—	—	60.18	63.42
UCIR (CNN) [14]	49.30 ± 0.32	57.57 ± 0.23	61.22 ± 0.69	64.01 ± 0.91
PODNet (NME)	**61.40 ± 0.68**	**62.71 ± 1.26**	**64.03 ± 1.30**	**64.48 ± 1.32**
PODNet (CNN)	**57.98 ± 0.46**	**60.72 ± 1.36**	**63.19 ± 1.16**	**64.83 ± 0.98**

Protocol. We validate our model and the compared baselines using the challenging protocol introduced by Hou et al. [14]: we start by training the models on half the classes (i.e., 50 for CIFAR100 and ImageNet100, and 500 for ImageNet1000). Then the classes are added incrementally in steps. We divide the remaining classes equally among the steps, e.g., for CIFAR100 we could have 5 steps of 10 classes or 50 steps of 1 class. Note that a training of 50 steps is actually made of 51 different tasks: the initial training followed by the incremental steps. Models are evaluated after each step on *all the classes seen until then*. To facilitate comparison, the accuracies at the end of each step are averaged into a unique score called *average incremental accuracy* [33]. If not specified otherwise, the average incremental accuracy is the score reported in all our results.

Following Hou et al. [14], for all datasets, and all compared models, we limit the memory M_{per} to 20 images per old class. For results with different memory settings, refer to Sec. 4.2.

Implementation details. For fair comparison, all compared models employ the same ConvNet backbone: ResNet-32 for CIFAR100, and ResNet-18 for ImageNet. We remove the ReLU activation at the last block of each ResNet end-of-stage to provide a signed input to POD (Sec. 3.1). We implemented our method (called here PODNet) in PyTorch [30]. We compare both ours and UCIR's implementation [14] of iCaRL. Results of UCIR come from the implementation of Hou et al. [14]. We provide their reported results and also run their code ourselves. We used our implementation of BiC in order to compare with the same backbone. We sample our memory images using *herding selection* [33] and perform the inference with two different methods: the *Nearest-Mean-Examplars* (NME) proposed for iCarl, and also adopted on one of the variants of UCIR [14], and the "CNN" method introduced for UCIR (see Sec. 2). Please see the supplementary materials for the full implementation details.

Table 2. Average incremental accuracy, PODNet *vs.* state of the art. Models with an asterisk * are reported directly from Hou et al. [14]

	ImageNet100				Imagenet1000	
	50 steps	25 steps	10 steps	5 steps	10 steps	5 steps
New classes per step	1	2	5	10	50	100
iCaRL* [33]	—	—	59.53	65.04	46.72	51.36
iCaRL [33]	54.97	54.56	60.90	65.56	—	—
BiC [38]	46.49	59.65	65.14	68.97	44.31	45.72
UCIR (NME)* [14]	—	—	66.16	68.43	59.92	61.56
UCIR (NME) [14]	55.44	60.81	65.83	69.07	—	—
UCIR (CNN)* [14]	—	—	68.09	70.47	61.28	64.34
UCIR (CNN) [14]	57.25	62.94	67.82	71.04	—	—
PODNet (CNN)	**62.48**	**68.31**	**74.33**	**75.54**	**64.13**	**66.95**
	± **0.59**	± **2.45**	± **0.93**	± **0.26**		

4.1 Quantitative results

The comparisons with all the state of the art are tabulated in Table 1 for CIFAR100 and Table 2 for ImageNet100 and ImageNet1000. All tables shows the average incremental accuracy for each considered models with various number of steps on the incremental learning run. The "New classes per step" row shows the amount of new classes introduced per task.

CIFAR100. We run our comparisons on 5, 10, 25, and 50 steps with respectively 10, 5, 2, and 1 classes per step. We created three random class orders to ran each experiment thrice, reporting averages and standard deviations. For CIFAR100 only, we evaluated our model with two different kind of inference: NME and CNN. With both methods, our model surpasses all previous state of the art models on all steps. Moreover, our model relative improvement grows as the number the steps increases, surpassing existing models by 0.82, 2.81, 5.14, and 12.1 percent points (*p.p.*) for respectively 5, 10, 25, and 50 steps. Larger numbers of steps imply stronger forgetting; those results confirm that PODNet manages to reduce drastically the said forgetting. While PODNet with NME has the largest gain, PODNet with CNN also outperforms the previous state of the art by up to 8.68 *p.p.*. See Fig. 4 for a plot of the incremental accuracies on this dataset. In the extreme setting of 50 increments of 1 class (Fig. 4a), our model showcases large differences, with slow degradation ("*gradual forgetting*" [8]) due to forgetting throughout the run, while the other models show a quick performance collapse ("*catastrophic forgetting*") at the start of the run.

ImageNet100. We run our comparisons on 5, 10, 25, and 50 steps with respectively 10, 5, 2, and 1 classes per step. For both ImageNet100, and ImageNet1000 we report only PODNet with CNN, as the kNN-based NME classifier did not generalize as well to larger-scale datasets. With the more complex images of Im-

Table 3. Ablation studies performed on CIFAR100 with 50 steps. We report the average incremental accuracy

(a) Comparison of the performance of the model when disabling parts of the complete PODNet loss

(b) Comparison of distillation losses based on intermediary features. All losses evaluated with POD-flat

Classifier	POD-flat	POD-spatial	NME	CNN
Cosine			40.76	37.93
Cosine	✓		48.03	46.73
Cosine		✓	54.32	57.27
Cosine	✓	✓	56.69	55.72
LSC-CE	✓	✓	59.86	57.45
LSC			41.56	40.76
LSC	✓		53.29	52.98
LSC		✓	**61.42**	57.64
LSC	✓	✓	61.40	**57.98**

Loss	NME	CNN
None	53.29	52.98
POD-pixels	49.74	52.34
POD-channels	57.21	54.64
POD-gap	58.80	55.95
POD-width	60.92	57.51
POD-height	60.64	57.50
POD-spatial	**61.40**	**57.98**
GradCam [5]	54.13	52.48
Perceptual Style [16]	51.01	52.25

ageNet100, our model also outperforms the state of the art on all tested runs, by up to 6.51 *p.p.*

ImageNet1000. This dataset is the most challenging, with much greater image complexity than CIFAR100, and ten times the number of classes as ImageNet100. We evaluate the models in 5 and 10 steps, and results confirm the consistent improvement of PODNet against existing arts by up to 2.85 *p.p.*

4.2   Further analysis & ablation studies

Ablation Studies. Our model has two components: the distillation loss POD and the LSC classifier. An ablation study showcasing the contribution of each component is displayed in Table 3a: each additional component improves the model performance. We evaluate every ablation on CIFAR100 with 50 steps of 1 new class each. The reported metric is the average incremental accuracy. The table shows that our novel method of constraining the whole ConvNet is beneficial. Furthermore applying only POD-spatial still beats the previous state of the art by a significant margin. Using both POD-spatial and POD-flat then further increases results with a large gain. We also compare the results with the Cosine classifier [27,14] against the Local Similarity Classifier (LSC) with NCA loss. Finally, we add LSC-CE: our classifier with multi-mode but with a simple cross-entropy loss instead of our modified NCA loss. This version brings to mind SoftTriple [32] and Infinited Mixture Prototypes [2], used in the different context of few-shot learning. The latter only considers the closest mode of each class in its class assignment, while LSC considers all modes of a class, thus, taking into account the intra-class variance. That allows LSC to decrease class similarity
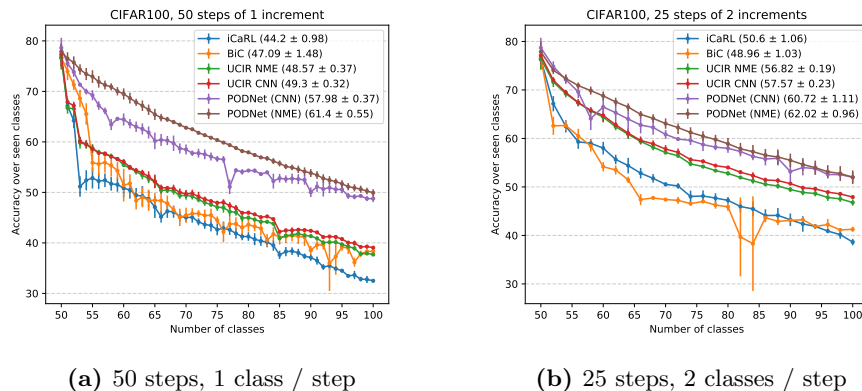
(a) 50 steps, 1 class / step   **(b)** 25 steps, 2 classes / step

Fig. 4. Incremental Accuracy on CIFAR100 over three orders for two different step sizes. The legend reports the average incremental accuracy.

when intra-class variance is high (which could signal a lack of confidence in the class).

Spatial-based distillation. We apply our distillation loss POD differently for the flat final embedding **h** (POD-flat) and the ConvNet's intermediate features maps $\mathbf{h}_\ell$ (POD-spatial). We designed and evaluated several alternative for the latter whose results are shown in Table 3b. Refer to Sec. 3.1 and Fig. 2 for their definition. All losses are evaluated with POD-flat. "*None*" is using only POD-flat. Overall, we see that not using pooling results in bad performance (POD-pixels). Our final loss, POD-spatial, surpasses all others by taking advantages of the statistics aggregated from both spatial axis. For the sake of completness we also included losses not designed by us: GradCam distillation [5] and Perceptual Style [16]. The former uses a gradient-based attention while the later – used for style transfer – computes a gram matrix for each channel.

Forgetting and plasticity balance. Forgetting can be drastically reduced by imposing a high factor on the distillation losses. Unfortunately, it will also degrade the capacity (its *plasticity*) to learn new classes. When POD-spatial is added on top of POD-flat, we manage to increase the oldest classes performance (+7 percentage points) while the newest classes performance were barely reduced (-0.2 *p.p.*). Because our loss POD-spatial constraints only statistics, it is less stringent than a loss based on exact pixels values as POD-pixel. The latter hurts the newest classes (-2 *p.p.*) for a smaller improvement of old classes (+5 *p.p.*). Furthermore our experiments confirmed that LSC reduced the sensibility of the model to distribution shift, as the performance it brings was localized on the old classes.

Robustness of our model. While previous results showed that PODNet improved significantly over the state-of-the-arts, we wish here to demonstrate here

Table 4. Effect of the memory size per class M_{per} on the models performance. Results from CIFAR100 with 50 steps, we report the average incremental accuracy

M_{per}	5	10	**20**	50	100	200
iCaRL [33]	16.44	28.57	44.20	48.29	54.10	57.82
BiC [38]	20.84	21.97	47.09	55.01	62.23	**67.47**
UCIR (NME) [14]	21.81	41.92	48.57	56.09	60.31	64.24
UCIR (CNN) [14]	22.17	42.70	49.30	57.02	61.37	65.99
PODNet (NME)	**48.37**	**57.20**	**61.40**	**62.27**	**63.14**	63.63
PODNet (CNN)	**35.59**	**48.54**	**57.98**	**63.69**	**66.48**	**67.62**

Table 5. Effect of the initial task size and the M_{total} on the models performance. We report the average incremental accuracy

(a) Evaluation of an easier memory constraint ($M_{\text{total}} = 2000$)

(b) Varying initial task size, with $M_{\text{per}} = 20$, and followed by 50 to 90 tasks made of a single class

	Nb. steps			Initial task size				
Loss	50	10	Loss	10	20	30	40	**50**
iCaRL [33]	42.34	56.52	iCaRL [33]	40.97	41.28	43.38	44.35	44.20
BiC [38]	48.44	55.03	BiC [38]	41.58	40.95	42.27	45.18	47.09
UCIR (NME) [14]	54.08	62.89	UCIR (NME) [14]	42.33	40.81	46.80	46.71	48.57
UCIR (CNN) [14]	55.20	63.62	UCIR (CNN) [14]	43.25	41.69	47.85	47.51	49.30
PODNet (NME)	**62.47**	**64.60**	PODNet (NME)	**45.09**	**49.03**	**55.30**	**57.89**	**61.40**
PODNet (CNN)	**61.87**	**64.68**	PODNet (CNN)	**44.95**	**47.68**	**52.88**	**55.42**	**57.98**

the robustness of our model to various factors. In Table 4, we compared how PODNet behaves against the baseline when the memory size per class M_{per} changes: PODNet improvements increase as the memory size decrease, up to a gain of $26.20\,p.p.$ with NME (resp. $13.42\,p.p.$ for CNN) with $M_{\text{per}} = 5$. Notice that by default, the memory size is 20 in Sec. 4.1. We also compared our model against baselines with a more flexible memory $M_{\text{total}} = 2000$ [33,38], and with various initial task size (by default it is 50 on CIFAR100). In the former case (Table 5a), models benefit from a larger memory per class in the early tasks. In the later case (Table 5b), models initialization is worse because of a smaller initial task size. In these settings very different from Sec. 4.1, PODNet still outperformed significantly the compared models, proving the robustness of our model.

5   Conclusion

We introduced in this paper a novel distillation loss (POD) constraining the whole convolutional network. This loss strikes a balance between reducing forgetting of old classes and learning new classes, essential for long incremental runs, by carefully chosen pooling. As a further refinement, we proposed a multi-mode similarity classifier, more robust to shift in the distribution inherent to incremental learning. Those innovations allow PODNet to outperform the previous state of the art in a challenging experimental context, with severe sample-per-class memory limitation, and long runs of many small-sized tasks, by a large margin. Extensive experiments over three datasets show the robustness of our model on different settings.

6   Supplementary Material

6.1   Spatial-based distillation without POD-flat

In Table 3.b of the main paper, we compared distillation loss alternatives to our final POD-spatial. In this table, the spatial-based losses were evaluated with POD-flat. In Table 6, we evaluate those same losses without POD-flat. "*None*" is using only our LSC classifier without any distillation losses. Notice that POD-spatial —and its sub-components POD-width and POD-height– are the only losses barely affected by POD-flat's absence. Note that all alternative losses were tuned on the validation set to get the best performance, including those from external papers. Still, our proposed loss, POD-spatial, outperforms all, both with and without POD-flat.

6.2   Implementation details

For all datasets, images are augmented with random crops and flips. For CIFAR100, we additionally change image intensity by a random value in the range [-63, 63]. We train our model for 160 epochs for CIFAR100, and 90 epochs for both ImageNet100 and ImageNet100, with a SGD optimizer with momentum of 0.9. For all datasets, we start with a learning rate of 0.1, a batch size of 128, and cosine annealing scheduling. The weight decay is $5 \cdot 10^{-4}$ for CIFAR100, and $1 \cdot 10^{-4}$ for ImageNet100 and ImageNet1000. For CIFAR100 we set model hyperparameters $\lambda_c = 3$ and $\lambda_f = 1$, while for ImageNet100 and 1000 we set $\lambda_c = 8$ and $\lambda_f = 10$. Our model uses POD-spatial and POD-flat except when explicitly stated otherwise. Following Hou et al. [14], we multiply both losses by the adaptive scaling factor: $\lambda = \sqrt{N/T}$ with N being the number of seen classes and T the number of classes in the current task.

Table 6. Comparison of distillation losses based on intermediary features. All losses evaluated without POD-flat.

Loss	NME	CNN
None	41.56	40.76
POD-pixels	42.21	40.81
POD-channels	55.91	50.34
POD-gap	57.25	53.87
POD-width	61.25	57.51
POD-height	61.24	57.50
POD-spatial	**61.42**	**57.64**
GradCam [5]	41.89	42.07
Perceptual Style [16]	41.74	40.80

For POD-spatial, before sum-pooling we take the features to the power of 2 element-wise. The vector resulting from the pooling is then L2 normalized.

6.3   Number of proxies per class

While our model's expressiveness increases with more proxies in $\mathcal{L}_{\text{LSC}}$, it remains fairly stable for values between 5 and 15, thus, for simplicity, we kept it fixed to 10 in all experiments.

In initial experiments, we had the following pairs for the number of clusters (k) and average incremental accuracy (acc): k=1, acc=56.80%; k=2, 57.14%; k=4, acc=57.40%; k=6, acc=57.46%; k=8, acc=57.95%, and k=10, acc=57.98% — i.e., a 1.18 p.p. improvement moving from k=1 to k=10. On ImageNet100, with 10 steps/tasks (increments of give classes per task), moving from k=1 to k=10 improved 1.51 p.p. on acc.

6.4   Reproducibility

Code Dependencies The Python version is 3.7.6. We used the PyTorch [30] (version 1.2.0) deep learning framework and the libraries Torchvision (version 0.4.0), NumPy [29] (version 1.17.2), Pillow (version 6.2.1), and Matplotlib [15] (version 3.1.0). The CUDA version is 10.2. Initial experiments were done with the data loaders library Continuum [6]. PODNet's full code is released at: github.com/arthurdouillard/incremental_learning.pytorch.
We provide all configuration files necessary to reproduce results, including seeds and class ordering.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the IEEE European Conference on Computer Vision (ECCV) (2018) 2, 3

2. Allen, K., Shelhamer, E., Shin, H., Tenenbaum, J.: Infinite mixture prototypes for few-shot learning. In: International Conference on Machine Learning (ICML) (2019) 12

3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the IEEE European Conference on Computer Vision (ECCV) (2018) 2, 3

4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009) 9

5. Dhar, P., Singh, R.V., Peng, K.C., Wu, Z., Chellappa, R.: Learning without memorizing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 3, 4, 12, 13, 16

6. Douillard, A., Lesort, T.: Continuum, data loaders for continual learning. https://github.com/Continvvm/continuum (2020). https://doi.org/10.5281/zenodo.3759673 16

7. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint library (2017) 2

8. French, R.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences (1999) 1, 4, 11

9. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 3

10. Goldberger, J., Hinton, G.E., Roweis, S.T., Salakhutdinov, R.R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems (NeurIPS) (2005) 8

11. Golkar, S., Kagan, M., Cho, K.: Continual learning via neural pruning. Advances in Neural Information Processing Systems (NeurIPS), Neuro AI Workshop (2019) 2

12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 6

13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Advances in Neural Information Processing Systems (NeurIPS), Deep Learning and Representation Learning Workshop (2015) 3

14. Hou, S., Pan, X., Change Loy, C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2, 3, 4, 7, 9, 10, 11, 12, 14, 15

15. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering (2007) 16

16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Proceedings of the IEEE European Conference on Computer Vision (ECCV) (2016) 7, 12, 13, 16

17. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: Proceedings of the International Conference on Learning Representations (ICLR) (2018) 2

18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences (2017) 2, 3

19. Komodakis, N., Zagoruyko, S.: Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In: Proceedings of the International Conference on Learning Representations (ICLR) (2017) 3, 7
20. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical Report (2009) 9
21. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. Object Categorization: Computer and Human Vision Perspectives, Cambridge University Press (2006) 5
22. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. Proceedings of the International Conference on Learning Representations (ICLR) (2019) 2
23. Li, Z., Hoiem, D.: Learning without forgetting. Proceedings of the IEEE European Conference on Computer Vision (ECCV) (2016) 2, 3
24. Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition. In: Annual Conference on Robot Learning (2017) 2
25. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems (NeurIPS) (2017) 2, 3
26. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (1999) 5
27. Luo, C., Zhan, J., Xue, X., Wang, L., Ren, R., Yang, Q.: Cosine normalization: Using cosine similarity instead of dot product in neural networks. In: International Conference on Artificial Neural Networks (2018) 3, 12
28. Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017) 8
29. Oliphant, T.E.: A guide to NumPy. Trelgol Publishing USA (2006) 16
30. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: Advances in Neural Information Processing Systems (NeurIPS), Autodiff Workshop (2017) 10, 16
31. Qi, H., Brown, M., Lowe, D.G.: Low-shot learning with imprinted weights. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 9
32. Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., Jin, R.: Softtriple loss: Deep metric learning without triplet sampling. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2019) 12
33. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3, 4, 9, 10, 11, 14
34. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science (1995) 1, 4
35. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Gradcam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017) 3
36. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Advances in Neural Information Processing Systems (NeurIPS) (2017) 2

37. Thrun, S.: Lifelong learning algorithms. Springer Learning to Learn (1998) 1
38. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 3, 4, 9, 10, 11, 14
39. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. In: Proceedings of the International Conference on Learning Representations (ICLR) (2018) 2
40. Zhou, P., Mai, L., Zhang, J., Xu, N., Wu, Z., Davis, L.S.: M2kd: Multi-model and multi-level knowledge distillation for incremental learning. arXiv preprint library (2019) 4