Continual prune-and-select: class-incremental learning with specialized subnetworks

Dekhovich, Aleksandr ; Tax, David M.J.; Sluiter, Marel H.F.; Bessa, Miguel A.

DOI
10.1007/s10489-022-04441-z

Publication date
2023

Document Version
Final published version

Published in
Applied Intelligence

Applied Intelligence
https://doi.org/10.1007/s10489-022-04441-z

Continual prune-and-select: class-incremental learning with specialized subnetworks

Aleksandr Dekhovich[1] · **David M.J. Tax**[2] · **Marcel H.F Sluiter**[1] · **Miguel A. Bessa**[3]

Accepted: 28 December 2022

Abstract
The human brain is capable of learning tasks sequentially mostly without forgetting. However, deep neural networks (DNNs) suffer from catastrophic forgetting when learning one task after another. We address this challenge considering a class-incremental learning scenario where the DNN sees test data without knowing the task from which this data originates. During training, Continual Prune-and-Select (CP&S) finds a subnetwork within the DNN that is responsible for solving a given task. Then, during inference, CP&S selects the correct subnetwork to make predictions for that task. A new task is learned by training available neuronal connections of the DNN (previously untrained) to create a new subnetwork by pruning, which can include previously trained connections belonging to other subnetwork(s) because it does not update shared connections. This enables to eliminate catastrophic forgetting by creating specialized regions in the DNN that do not conflict with each other while still allowing knowledge transfer across them. The CP&S strategy is implemented with different subnetwork selection strategies, revealing superior performance to state-of-the-art continual learning methods tested on various datasets (CIFAR-100, CUB-200-2011, ImageNet-100 and ImageNet-1000). In particular, CP&S is capable of sequentially learning 10 tasks from ImageNet-1000 keeping an accuracy around 94% with negligible forgetting, a first-of-its-kind result in class-incremental learning. To the best of the authors' knowledge, this represents an improvement in accuracy above 10% when compared to the best alternative method.

Keywords Continual learning · Class-incremental learning · Sparse network representation · Catastrophic forgetting

1 Introduction

Despite significant progress, deep learning methods tend to forget old tasks while learning new ones. This is known as *catastrophic forgetting* in neural networks [2, 3]. In the conventional setting, a machine learning model has access to the entire training data at any point in time. Instead, in continual learning or lifelong learning [4] data for a given task comes in sequentially at a specific learning moment, and then new data associated with another task comes in at a different moment. Continual learning aims at creating deep learning models that do not forget previously learned tasks while being able to learn new ones, i.e. addressing catastrophic forgetting. Classification continual learning problems can be separated into two scenarios [5]: task-incremental learning (task-IL), where the task being solved is known both during training and inference; and class-incremental learning (class-IL) [6], where the task-ID is known only during training but unknown during testing. The class-IL problem is notably more challenging than task-IL. There are also incremental learning approaches for object detection [7, 8] and semantic segmentation [9, 10]. However, to the best of our knowledge, there are no examples in the literature of combining different types of learning tasks. Addressing current challenges in incremental learning brings the community one step closer to mimicking the abilities of the human brain. Recently, a brain-inspired replay method [11] was proposed to generate an internal feature representation in order to reduce forgetting. The idea behind this technique is to be able to generate features of old data, imitating memory mechanisms in the brain.

✉ Miguel A. Bessa
miguel_bessa@brown.edu

Extended author information available on the last page of the article.

Published online: 13 January 2023

There is evidence from neuroscience [12–14] that humans have special regions in the brain that are responsible for the recognition of specific patterns. Moreover, several studies show that the human brain encodes information in a sparse representation with an optimal fraction of active neurons of 1%-4% at the same time [15, 16]. Motivated by this observation, we propose a class-IL algorithm for image classification based on two steps: creating a subnetwork for a given task during training and selecting a previously obtained subnetwork during inference to make predictions. The first stage is achieved via iterative pruning that propagates input patterns through the network and eliminates the least useful connections. During inference, we first predict the current task when selecting the appropriate subnetwork from a small batch of test samples, and only then make a prediction with the selected subnetwork. We allow overlaps between subnetworks in order to induce knowledge transfer during training of new tasks. However, previously trained weights are not changed. Parameter update only occurs when training available neuron connections, which become part of a new subnetwork associated with the new task.

Our contribution To the best of our knowledge, we propose for the first time a general strategy to create overlapping subnetworks of neuronal connections that share knowledge with each other for the class-IL scenario. In doing so, we achieve a strategy with negligible forgetting, unlike other works to date. Importantly, the choice of methods for subnetwork creation and for subnetwork selection *can be* different from the ones considered herein. Our goal is to propose a simple working paradigm for class-IL problems, where firstly some connections are assigned to a specific task during training that can then be selected during inference *without* knowing the task-ID.

This paper starts discussing state-of-the-art methods for continual learning, their differences and fundamental assumptions (Section 2). Then, the proposed CP&S strategy is presented (Section 3), and evaluated (Section 4) on class-incremental learning scenarios constructed from various datasets, including CIFAR-100 [17], ImageNet-100, ImageNet-1000 [18], and CUB-200-2011 [19]. In Section 5, additional experiments are provided showing limitations of CP&S approach and opening avenues for the conclusions and future directions (Section 6).

2 Related work

Class-incremental learning As previously mentioned, continual learning problems are usually classified according to whether or not the task-ID is available during inference. We focus on the class-IL scenario where the task ID is absent during inference since it is the most realistic and challenging scenario of continual learning. All class-IL methods are usually divided into three categories [20]: *regularization* [21–26], *rehearsal* [6, 27–29] and *architectural* [30–32].

Purely *regularization-based* methods introduce an additional term in the loss function to prevent forgetting. Some approaches [21, 25, 26, 33] estimate the importance of connections for a given task and penalize the model for gradient updates during training for the next tasks. Learning without forgetting (LwF) [22] adds a term in the loss function that penalizes changes in old output heads for new data while training new output heads on new data. Regularization-based methods have the advantage of not storing past data in memory nor needing network expansion, but they perform worse compared to other approaches [5].

Rehearsal methods replay small amounts of old classes [6] or generate synthetic examples [34] to be able to predict previously seen classes. iCaRL uses the nearest class mean [35] (NCM) classifier together with fixed memory of old data to mitigate forgetting. Bias-correction methods [27, 36, 37] aim to tackle the tendency of class-IL algorithms to be biased towards classes of the last tasks, which arises due to class imbalance at the latest stages [5]. PODNet [29] has multiple terms in the loss function using old data from the memory of a fixed size, penalizing signal deviations not only in the output layer but also in intermediate ones. AFC [38] uses knowledge distillation by estimating the importance of each feature map. The estimation is based on the increase of loss function from changing channels' parameters in the feature maps. The obvious limitation of rehearsal methods is the need to keep past data, which is often not desirable in practical applications due to privacy issues as mentioned by [8].

Architectural methods follow a different strategy where the network architecture is modified to avoid forgetting. For example, the Dynamically Expandable Network (DEN) [30] expands the network architecture in an online manner, increasing network capabilities, and introducing a regularization term to prevent forgetting. However, due to the expansion of the network, the final number of parameters is greater than for the original architecture, which increases the memory costs. Supermasks in Superposition (SupSup) [31] and SpaceNet [32] find subnetworks for every task. SpaceNet assigns parameters to one task only, without sharing knowledge between tasks which limits its allocation capabilities for long sequences of tasks. In addition, SpaceNet requires to pre-define the sparsity level for each task. SupSup uses a randomly weighted backbone [39] instead of pruning to obtain a task-related subnetwork. During inference, SupSup predicts the correct subnetwork for the given test data, using all data points in the batch. In the provided experiments, the batch size is equal to 128 images which may not be applicable to real-life problems. DER [40]

dynamically expands the feature extractor by introducing new channels and freezes old feature representation while learning a new task. Later, DER uses a small portion of old data and current data to finetune the network for all tasks. To stop the growth of the number of parameters, DER uses a pruning strategy, however, the final number of parameters is unpredictable. Similarly, FOSTER [41] introduces a new module with a feature map to learn new classes. However, it uses a knowledge distillation strategy inspired by gradient boosting instead of pruning to compress the model. As a result, the outcome of FOSTER is a single fixed-sized backbone network.

A Meta-Learning approach for class-IL is proposed by iTAML [42]. The algorithm for updating parameters for all old tasks also needs fixed-sized memory, but iTAML uses a momentum-based strategy for meta-updates to overcome catastrophic forgetting. At the test stage, iTAML starts by predicting the task associated with that sample using a given test batch, and then adapts its parameters to the predicted task using data from fixed memory. Finally, with the adapted model and predicted task-ID, iTAML makes a prediction. Overall, iTAML uses samples from previous tasks to prevent forgetting, and the batch of test data to predict task-ID, making it the most demanding algorithm out of consideration. Also, the model adaptation to the predicted task makes it computationally more expensive than other state-of-the-art methods.

Iterative pruning for continual learning Typically, neural network pruning is used for model compression such that it reduces memory and computational costs. The pruning pipeline consists of three steps: network pretraining, deleting the least important connections or neurons based on some criterion, and network retraining. Iterative pruning is characterized by repeating the second and third steps several times. There are numerous approaches to pruning, namely magnitude pruning [43–45], data-driven pruning [46–49] and sensitivity-based pruning [50–53]. Iterative pruning has been recently applied in the context of task-IL but not in the more challenging class-IL scenario, where the task-ID is not known *a priori*. Unsurprisingly, pruning has been shown to lead to simplified neural networks with a small fraction of the original parameters. This can facilitate the accumulation of knowledge for new tasks, as demonstrated by task-IL methods based on iterative pruning, namely PackNet [54] that uses magnitude connections pruning [43], and CLNP [55] that uses data-driven neurons pruning [47]. Piggyback [56] learns the mask for every task, as well as CPG [57] which also expands a network in the ProgressiveNet manner [58]. The performance of these algorithms is strong for task-IL, but they have the significant limitation of requiring to know the task-ID.

We are interested in developing a class-IL method that contains the benefits of iterative pruning connections to obtain sparse network representations while being capable of selecting tasks without knowing the task-ID. To this effect, we developed a pruning strategy called NNrelief [49] that aims at leaving as many connections as possible available for future tasks, leading to sparser networks when compared to other pruning methods. The algorithm's idea is to propagate signal through the network, compute a metric called importance score for each connection which estimates its contribution to the signal of the following neuron, and then prune the least contributing connections incoming to the neuron.

Task selection Currently, there are few strategies for task selection in class-incremental learning. For example, iTAML [42] and SupSup [31] use similar ideas for task identification class-IL applied to image classification problems, and neither method uses pruning as a means to create space for new knowledge. The underlying assumption is that if a classifier network is well-trained, the highest output signal in the neuron of the output layer corresponds to the class belonging to the correct task. So, iTAML sums the largest output values of every task-related output in that layer over every test image in the batch, and then finds the layer with the highest total sum. SupSup relies on the entropy of the signal in each of the heads, in the hope that the model is confident in its prediction when it is in the correct head, meaning that the entropy of signal within the head should be smaller than in other heads. Note that both methods use batches of test samples to select the correct task: iTAML varies the batch size from 20 to 150 depending on the dataset, while SupSup uses 128 images in their experiments. A different strategy is pursued by Kim et al. [59], where an autoencoder is associated with a task during training. In the test stage, the reconstruction loss is computed for *every* autoencoder with the given test image, and the one with minimum reconstruction loss is chosen to make predictions. Subsequently, a classification model makes a prediction with the given predicted task-ID. It was shown that in the case of LwF [22] and LwM [23] this task-selection procedure improves classification accuracy. However, this task-selection approach requires training an autoencoder for every task which is impractical.

Limitations of class-IL approaches State-of-the-art class-IL methods have simplified training by replaying old data [6, 27, 29, 39], doing inference with a batch of images to determine the current task [31, 42] and by performing adaptation before inference [42]. Table 1 summarizes these assumptions for each method. In rehearsal methods, examples of previous classes are stored (with fixed or

Table 1 Assumptions used by different types of class-IL methods ("bs" means batch size)

Methods	Replay old data	test $bs > 1$	Adaptation
SI, MAS, LwF, LwM, SpaceNet	no	no	no
iCaRL, BiC, PODNet, DER, AFC, FOSTER	yes	no	no
iTAML	yes	yes	yes
CP&S (ours)	no	yes	no

growing memory), which makes them inappropriate when images should not be kept for a long time. Similarly, the adaptation of a model for a given batch of test data before making a prediction (as in iTAML) is only possible when having examples in memory. Furthermore, the need for a significant number of images in a batch during inference arises from the difficulty of identifying the task-ID correctly with one image only. These can be strong model constraints when considering real-life applications.

3 Proposed method

Our method (CP&S) is based on training a subnetwork for each given task and then selecting the correct subnetwork when doing inference for new data with an unknown task-ID. We start by training a regular neural network for a specific task (Fig. 1(a)), iteratively pruning it to find a subnetwork with good performance (Fig. 1(b)). This creates a trained subnetwork capable of performing that particular task, leaving the remaining network free for future tasks (Fig. 1(c)). Importantly, when a new task comes in (Fig. 1(d)), the corresponding new subnetwork is found by iteratively pruning the entire original network – including all free neuronal connections *and* all existing subnetworks found for other tasks (Fig. 1(e)). This is possible by freezing the parameters of previously found subnetworks (avoiding to forget past tasks associated with the corresponding subnetworks), updating all the remaining parameters of the network, and then pruning the entire network until the corresponding subnetwork is found. This way, the new subnetwork (Fig. 1(f)) can contain connections from other subnetworks but it does not affect their performance on past tasks because it did not update the parameters of *shared connections* – it only updated the parameters of *unshared connections*. This allows to have the transfer of knowledge from one task to another without forgetting (Fig. 1(g)).

The new strategy proposed herein can be implemented with different pruning algorithms to create each subnetwork, and with different task selection algorithms to find the correct subnetwork for inference. As long as the pruning and selection strategies have reasonable performance, we expect this strategy to outperform previous continual learning methods in the class-IL scenario because (1) it avoids forgetting if the task is selected correctly (unlike iTAML); (2) it allows knowledge transfer among tasks (unlike SpaceNet). Also, CP&S (3) does not need to replay old data (unlike iCaRL, RPS-net, BiC, LUCIR, PODNet,

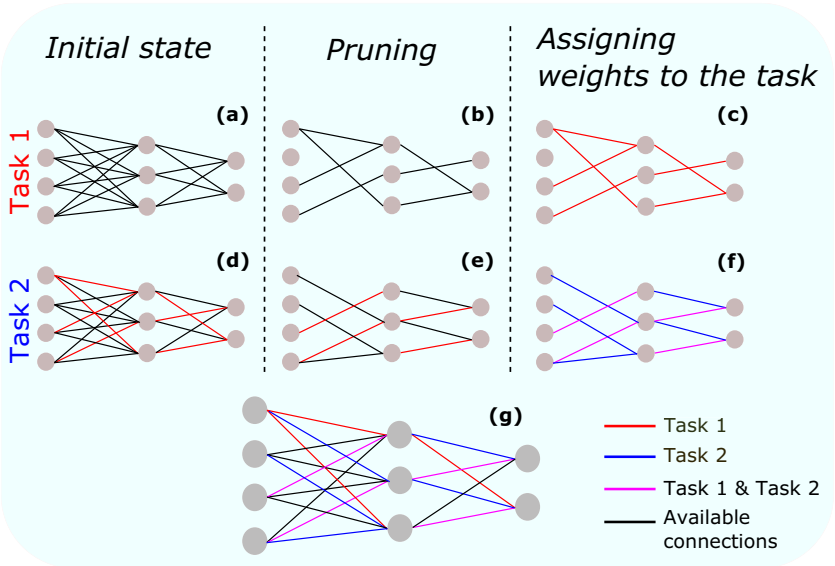

Fig. 1 The overview of Continual Prune-and-Select (CP&S) training procedure. In stages (a)–(c), the first task is learned; in (d)–(f) the network learns task 2; the final outcome is in (g), where the network is trained for both tasks

AFC); (4) The backbone architecture is fixed and never changes during training (unlike DER), or requires additional temporary modules (unlike FOSTER).

Without loss of generality, we use our recently developed NNrelief [49] pruning algorithm because it promotes sparser networks when compared to the state of the art, and it creates a renormalization effect in the network that distributes the importance of neuronal connections. Concerning task selection (in our case subnetwork selection), we considered different strategies, including the one proposed in the literature that applies to our method (see iTAML [42] and SupSup [31]).

Formally, denoting our classification network as $\mathcal{N}$ and considering T tasks, then:

$$\mathcal{N} = \cup_{t=1}^{T} \mathcal{N}^t, \tag{1}$$

where $\mathcal{N}^t$ is the subnetwork for task t, with $t = 1, 2, \ldots, T$. Each subnetwork $\mathcal{N}^t$ is found with our NNrelief pruning algorithm that determines the most important parts of the main network for solving a given task t. This algorithm estimates each connection's contribution to the total signal of a receiving neuron when compared to the other connections that are incoming to that neuron. This contribution is computed by the *importance score* (IS) of every connection:

$$s_{ij}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{\overline{|w_{ij}x_i|}}{\sum_{k=1}^{m} \overline{|w_{kj}x_k|} + |b_j|}, \tag{2}$$

for the input signal $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with N data points, $\mathbf{x}_n = (x_{n1}, \ldots, x_{nm}) \in \mathbb{R}^m$ for m neurons in that layer, where $\overline{|w_{ij}x_i|} = \frac{1}{N}\sum_{n=1}^{N} |w_{ij}x_{ni}|$ and with w_{ij} being the weight of the connection between neurons i and j, where b_j is the bias in neuron j. Then NNrelief prunes the connections entering the neuron with the lowest contribution to the importance score whose sum is less than $(1-\alpha)\sum_{i=1}^{m} s_{ij}$, where α is the hyperparameter of the algorithm, $0 \leq \alpha \leq 1$. More details are given in our original article [49].

Input: network $\mathcal{N}$, datasets $\{\mathbf{X}^t\}_{t=1}^{T}$. Initialize learning parameters p (learning rate, weight decay, number of epochs, etc.), pruning parameters (for NNrelief algorithm: α and the number of pruning iterations k)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathcal{N}^t \leftarrow \text{Pruning}(\mathcal{N}, \mathbf{X}^t, \alpha, k)$
3: $\quad$ freeze parameters $w \in \mathcal{N}^t$ and never update them
4: **end for**

Output: network $\mathcal{N}$ that learned tasks $1, 2, \ldots, t+1$.

Algorithm 1 Pseudocode for CP&S training procedure.

In the context of class-IL we receive datasets $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^T$ sequentially. The pruning algorithm then creates masks $\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^T$ for every task $t = 1, 2, \ldots, T$, where $\mathbf{M}^t = (m_{ij}^t)_{i,j}$,

$$m_{ij}^t = \begin{cases} 1, & \text{if there is an active connection} \\ & \text{between neurons } i \text{ and } j, \\ 0, & \text{otherwise} \end{cases}$$

and corresponding importance scores $S^1, S^2, \ldots, S^T$, $S^t = (s_{ij}^t)_{i,j}$.

Once the subnetworks are created during training, selecting the correct subnetwork given a batch of test data becomes essential to do inference. In this article, we define a test batch of size s as $\mathbf{X}^{test} = \{\mathbf{x}_1^{test}, \mathbf{x}_2^{test}, \ldots, \mathbf{x}_s^{test}\}$, and can simplify the notation to cases where the fully connected part of the network consists of one layer since we run all our experiments on ResNet architectures. However, there are no restrictions to apply this approach to any other type of architecture. We define the convolutional part for task t as θ^t, $\theta^t : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^d$ (H, W are the height and width of an input image and d is the length of an output feature vector), and the fully connected layers as $\varphi^t : \mathbb{R}^d \rightarrow \mathbb{R}^{num\_classes}$.

Similarly to the selection of the pruning algorithm for subnetwork creation, we can also adopt different strategies to identify the correct subnetwork associated with a particular task. In order to establish a fair comparison with the literature, we focus on the *maximum output response* (maxoutput) strategy that is used by other methods (e.g. [31, 42]), but we also show that other task selection methods can lead to good results (see Appendix for a strategy based on Importance Scores).

The maxoutput strategy for task prediction is simply formulated as:

$$t^* = \arg\max_{t=1,2,\ldots,T} \sum_{i=1}^{s} \max \varphi^t(\theta^t(\mathbf{x}_i^{test})). \tag{3}$$

This does not require data storage. There are no memory costs associated with this prediction.

Input: network $\mathcal{N}$, test batch $\mathbf{X}^{test}$.

1: predict task t^* for the test batch $\mathbf{X}^{test}$ using (3)
2: make a prediction $\hat{\mathbf{y}} = \mathcal{N}^{t^*}(\mathbf{X}^{test})$

Output: predicted classes $\hat{\mathbf{y}}$ for test data $\mathbf{X}^{test}$.

Algorithm 2 Pseudocode for CP&S inference procedure.

Complexity analysis We estimate this separately for the training and inference stages. In the training stage for one task, we perform initial training and then the prune-retrain steps over k iterations. For retraining, we use a smaller number of epochs. Therefore, if we initially train

the network with N epochs, then we define the number of retraining epochs as N_1 with $N_1 < N$. Overall, we have $N + kN_1 < (1 + k)N$ epochs. In practice, we use $k \leq 3$, so the total number of training epochs for one task can be estimated as $\mathcal{O}(N)$. During inference, the maxoutput approach requires propagating input signal through every created subnetwork. That means if T tasks are learned, we need T inference operations.

4 Experiments

We compare CP&S with different methods available in the literature. Aiming to establish a fair comparison, we use different measurements of accuracy during the learning process, namely *average multi-class accuracy* (ACC), *backward transfer metric* (BWT) [60] and *average incremental accuracy* (AIA) [6]. These metrics can be written assuming that a model learned T tasks and denoting R_{t_2,t_1} as the accuracy for task t_1 after learning up to task t_2 (inclusive, i.e. $t_2 \geq t_1$):

$$\text{ACC}(T) = \frac{1}{T}\sum_{t=1}^{T} R_{T,t} \tag{4}$$

$$\text{BWT}(T) = \frac{1}{T-1}\sum_{t=1}^{T-1} R_{t,t} - R_{T,t} \tag{5}$$

$$\text{AIA}(T) = \frac{1}{T}\sum_{t=1}^{T} \text{ACC}(t) \tag{6}$$

The idea of the BWT is to measure the forgetting of the incremental-learning models, evaluating how much information about previous tasks is lost after learning a new one. We evaluate all methods using several class orderings to obtain robust results, as recommended in [5].

Datasets We evaluate CP&S on three datasets: ImageNet-1000, including its subset ImageNet-100 [18]; CUB-200-2011 [19]; and CIFAR-100 [17]. We also consider different task construction scenarios. For completeness, the datasets are briefly described as follows:

- ImageNet-1000 consists of 1,281,167 224 × 224 RGB images for training and 50,000 images for validation of 1000 classes. We split both ImageNet-100 and ImageNet-1000 in 10 incremental steps of equal size, similarly to the literature;
- CUB-200-2011 consists of 11,788 224 × 224 RGB images of 200 classes with 5,994 training and 5,794 for testing images;
- CIFAR-100 consists of 60,000 32 × 32 RGB images of 100 classes with 6k images per class. There are 50,000 training samples and 10,000 test samples;

We start our experiments with ImageNet-100 (the first 100 classes of the ImageNet-1000 dataset) and with CIFAR-100 before considering more challenging datasets such as ImageNet-1000 and CUB-200-2011. For all datasets, we use the ResNet-18 architecture, as considered by previous methods. For ImageNet-100/1000 datasets, we split them into 10 tasks of the same size (each task having 10 classes). We compare CP&S with other state-of-the-art models, namely iCaRL [6], EEIL [36], BiC [27], RPS-net [39], iTAML [42], DER [40] and FOSTER [41]. In addition, we provide a comparison with the case of Finetuning, when no anti-forgetting actions are performed, and a network sequentially learns new tasks one by one. For comparison with other works, we either reproduce the results from the official GitHub repository using the hyperparameters mentioned in the original articles or report the results from the original works when available. See the details in Appendix A.

ImageNet-100 Figure 2 shows that CP&S outperforms state-of-the-art methods for this dataset, even when considering two different optimizers – Stochastic Gradient Descent (SGD) and Adam [61]. We use a learning rate of 0.1 for SGD and 0.01 for Adam, dividing it by 10 on epochs 30 and 60, and we consider weight decay of 10^{-4} for both optimizers. Figure 2(a) shows our predictions using a smaller

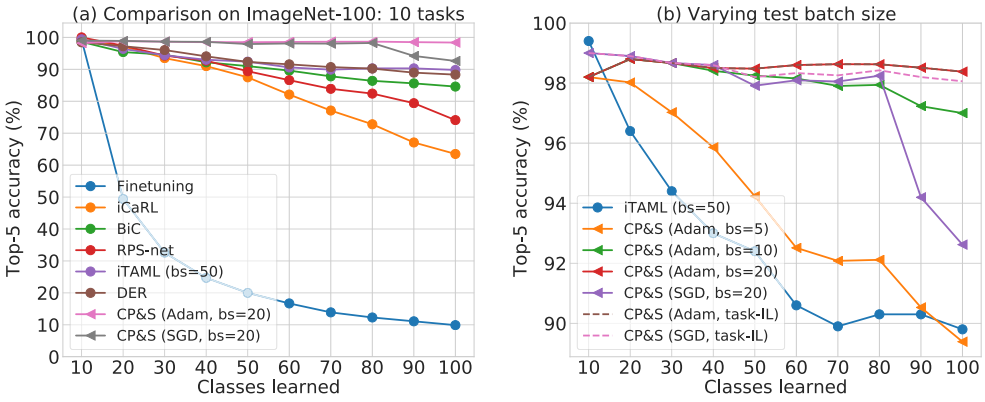

Fig. 2 Results on ImageNet-100 and comparison with other approaches. Notation: "bs" refers to the test batch size; "task-IL" refers to the task-IL scenario where the task-ID is known, providing an upper bound to the results. The pruning parameter of CP&S is $\alpha_{conv} = 0.9$ for both optimizers, SGD and Adam. The class ordering is generated by seed 1993 (iCaRL seed)

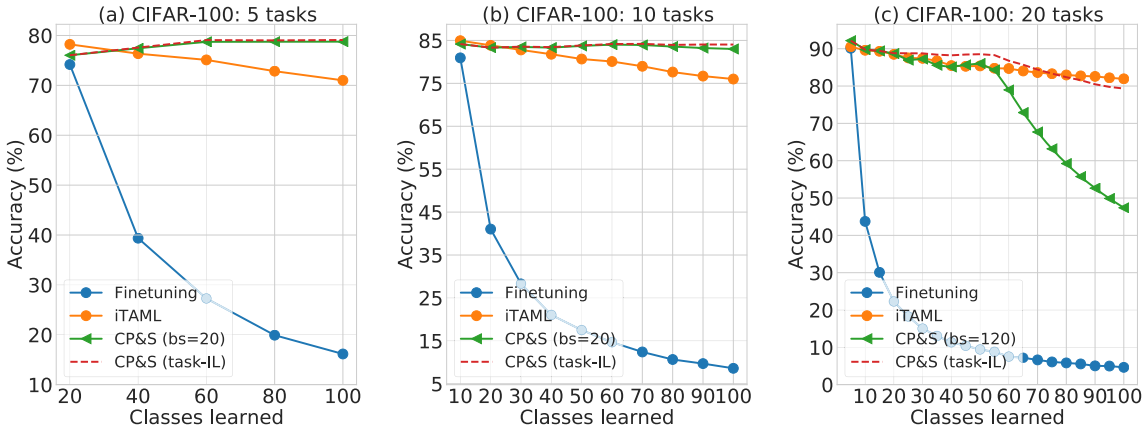

Fig. 3 Comparison with iTAML on CIFAR-100 split in 5, 10 and 20 tasks. Notation: "bs" refers to the batch size during inference, "task-IL" refers to the task-IL scenario where the task ID is known, providing an upper bound to the results. Five different class orderings are used

batch size than iTAML – 20 samples instead of 50 – and compares them with other methods. Figure 2(b) clarifies the influence of considering different test batch sizes in CP&S method, where it is demonstrated that even when using 5 or 10 samples per batch we still perform better. The same figure also shows that when using Adam we identify the correct subnetwork in 100% of the cases because we reach the upper bound provided in the task-IL scenario, i.e. where the task-ID is known and subnetwork selection is not necessary. For SGD, we observe a slight drop in accuracy after task 8 compared to the task-IL scenario, although it still outperforms iTAML even though the latter uses 50 images for task identification and requires keeping images in memory. Overall, CP&S reaches 98.38% accuracy with Adam and 92.62% with SGD, translating into improvements for this dataset beyond 8% and 2.5% when compared to next best method, and even larger when compared to other methods after all classes are learned.

We note that using Adam [61] is advantageous for CP&S due to the higher level of sparsity that is produced after pruning with NNrelief when compared with other optimizers [49]. Note that pruning makes neuron connections available for creating new subnetworks associated with future tasks. If the number of available connections is small, then new subnetworks may not be sufficiently expressive to reach high accuracy for a given task. A similar effect is expected if the number of tasks is large, as shown in the next experiments for CIFAR-100.

CIFAR-100 Before considering more challenging datasets such as ImageNet-1000 and CUB-200-2011, we focus on CIFAR-100 where we split its 100 classes by a different number of tasks: 5, 10 and 20 tasks composed of 20, 10 and 5 classes, respectively. For iTAML, we followed the original implementation with hyperparameters described in the paper including the test batch size equal to 20, and using *ResNet-18(1/3)* [60] which is a modified version of the standard ResNet-18 architecture where the number of filters is divided by three. For CP&S, we use Adam for training using 70 epochs and starting with the learning rate 0.01 which is then divided by 5 every 20 epochs. We use $\alpha_{conv} = 0.9$ and 3 pruning iterations as the pruning parameters of NNrelief.

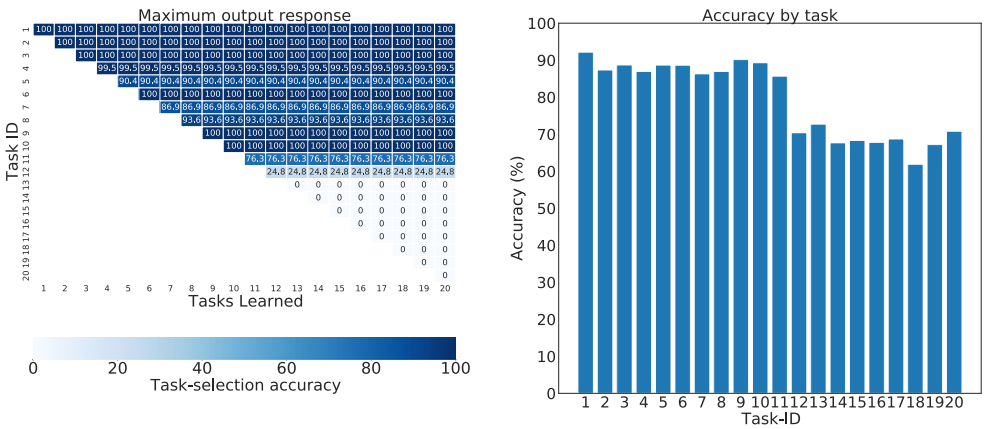

Fig. 4 CIFAR-100 divided into 20 tasks of 5 classes each. Task-selection accuracy with maxoutput (**left**) and accuracy by task (**right**)

Figure 3 shows that when considering 5 or 10 tasks CP&S significantly outperforms iTAML with the same batch size. However, for twenty tasks our performance drops sharply after the eleventh task, even in the ideal case where the task-ID is given (task-IL scenario). Despite being able to alleviate this drop by considering a larger test batch size, or by considering a different strategy for task (subnetwork) selection based on Importance Scores (see Appendix A), we observe this drop occurs approximately at the same number of tasks, independently of the class ordering used. Figure 4(left) provides an explanation of what occurs for the 20 tasks case by showing a heatmap with the task-selection accuracy by row for every task after a new task is learned. We also evaluate the prediction accuracy for each task when the task-ID is known (task-IL scenario) so that we can isolate the effect of not being able to appropriately select the subnetwork of interest for a given task and the effect of achieving low accuracy for a specific task. We observe that even in this case, the accuracy for each new task after the eleventh also drops (see Fig. 4 (right)). Therefore, our method performs well until we reach a saturation point when there are not enough neuron connections available to create a sufficiently large subnetwork to achieve high prediction accuracy for a new task. This is a logical conclusion, as one can only learn new tasks while sufficient neuronal connections remain available for training. Increasing the size of the original architecture eliminates this issue.

In addition, note that we do not keep data in memory (no replay), nor do we need to use adaptation to estimate the task before making a final prediction, unlike the methods reviewed above. We also use smaller test batch sizes than iTAML, despite using the same task selection strategy. The following experiments show that this conclusion holds for more challenging datasets.

ImageNet-1000 Focusing now on a more challenging dataset, we split ImageNet-1000 into 10 tasks of 100 classes. To evaluate CP&S, we train ResNet-18 with 90 epochs and SGD with a learning rate equal to 0.1 dividing it by 10 every 30 epochs. Figure 5(a) shows that CP&S performs better than the state-of-the-art, exhibiting more than 10% higher Top-5 accuracy than the next best method, which is DER [40] and more than 20% improvement over the second best BiC [27]. We found this result to be particularly striking, since the prediction accuracy remains around 94% with virtually no forgetting for the first time in the literature, to the best of our knowledge. Figure 5(b) also shows results for different test batch sizes for determining the task-ID and corresponding subnetwork. Once again, a batch size of 20 provides a good trade-off between accuracy and sample size. Interestingly, prediction accuracy is better for CP&S method than others even when using only 5 test samples in the batch. With 20 images in the test batch, we can almost reach the upper bound of the task-IL scenario, completely reaching it when using 50 images (i.e. identifying the task-ID correctly in 100% of the cases).

Table 2 Average incremental accuracy on ImageNet-1000

Method	Top-1 AIA
iCaRL [6]	38.4
DER [40]	66.73
FOSTER [41]	68.3
CP&S (ours)	**79.08**

Bold numbers indicate the best performance

In addition, we provide a comparison on the ImageNet-1000 dataset calculating Top-1 accuracy. In Table 2, we observe that CP&S outperforms the two most recent state-of-the-art methods, DER and FOSTER, by more than 10%.

CUB-200-2011 We split CUB-200-2011 dataset into four tasks with 50 classes in each of the tasks. For testing, we take standard ResNet-18 pretrained on ImageNet-1000 [18] and fine-tuned with SGD. For iTAML, we also use pretrained weights and use the same hyperparameters for fine-tuning that are used in the original paper for other large-scale datasets. The pruning parameter for CP&S is $\alpha_{conv} = 0.95$ and only one pruning iteration is used.

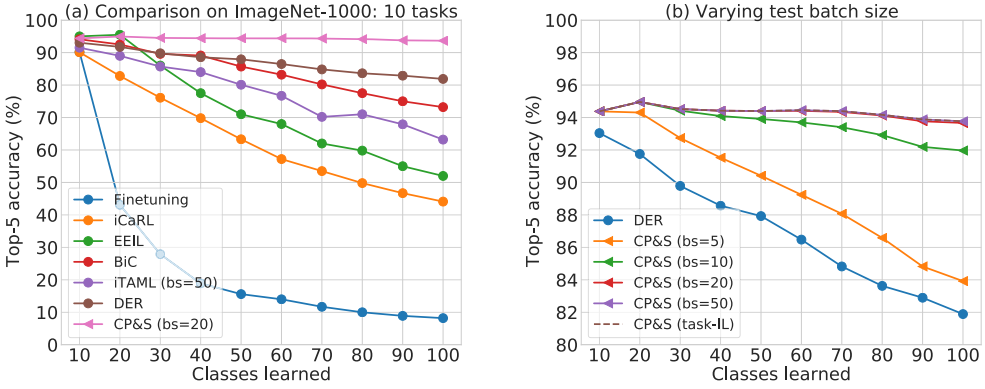

Fig. 5 Results obtained for ImageNet-1000 dataset and comparison with other approaches **(a)** and different batch sizes **(b)**. Notation: "bs" refers to the test batch size; "task-IL" refers to the task-IL scenario (upper bound obtained when task ID is known). The pruning parameter is $\alpha_{conv} = 0.9$ for CP&S. Class ordering is generated by seed 1993 (referring to iCaRL's seed)

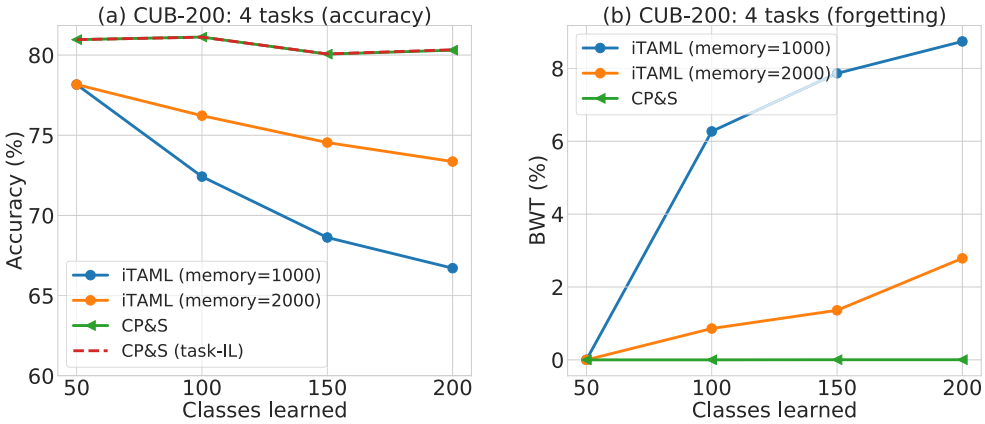

Fig. 6 Comparison with iTAML on four tasks constructed from CUB-200-2011. Notation: "memory" is the number for images from previous tasks; "task-IL" refers to task-IL scenario as an upper-bound for our approach

Figure 6 presents the accuracy and BWT history with 20 test images per batch, once again using maxoutput as the task-selection strategy. For 20 test images, it can be observed that CP&S once again exhibits almost no forgetting of information about previous tasks while learning new ones. However, iTAML even though it keeps 2000 images in memory, continuously forgets previous tasks. In addition, note that 2000 images represent 1/3 of the CUB-200-2011 dataset, and that we see a dramatic loss of performance for iTAML when using 1000 images (which is still 1/6 of all the images). In the case of 5 images per test batch, we obtain similar forgetting as iTAML with 2000 images in memory but still a better forgetting metric than iTAML with 1000 images in memory. A more detailed comparison can be found in Appendix C.

In summary, CP&S outperformed the state-of-the-art for all datasets considered with the exception of CIFAR-100 when considering a large number of tasks. We demonstrate that we can perform better for small-scale and large-scale datasets (ImageNet-1000) where the second best methods are different. We considered scenarios where each task has a small or a large number of classes, including cases where there is a small number of training examples (CUB-200-2011) without keeping them in memory.

5 Further analysis

CP&S method's performance degrades if there are too many tasks because the number of available neuron connections is not enough to create an expressive subnetwork and to select the correct task. This was shown for CIFAR-100 when considering 20 tasks (see Fig. 3). In addition, there are scenarios where task selection during inference should be performed by a different strategy instead of maxoutput. For example, when there is an imbalanced number of classes within the tasks we note that using the modification of Importance Scores (IS) to select tasks is advantageous. Focusing on fully connected layers, for the given dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s\}$ we can compute:

$$\hat{s}^t_{ij} = \overline{w_{ij} \cdot m^t_{ij} \cdot \theta^t_i(\mathbf{X})} = \begin{cases} \overline{w_{ij} \cdot \theta^t_i(\mathbf{X})}, & \text{if there is an active connection between neurons } i \text{ and } j \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $\overline{w_{ij} \cdot \theta^t_i(\mathbf{X})} = \frac{1}{s}\sum_{k=1}^{s} w_{ij} \cdot \theta^t_i(\mathbf{x}_k)$ and θ^t_i are the feature extractor layers of task t.

Suppose we have importance scores $S^1, S^2, \ldots, S^T$ obtained from the training set, we can estimate the importance scores of these connections based on $\mathbf{X}^{test}$ for every subnetwork $t = 1, 2, \ldots, T$, and denote these estimations as $\hat{S}^1, \hat{S}^2, \ldots, \hat{S}^T$.

Assuming that importance scores should be similar for train and test data for the true task-ID, we can formulate the decisive rule as:

$$t^* = \underset{t=1,2,\ldots,T}{\arg\min} \sqrt{\sum_{i,j} (s^t_{ij} - \hat{s}^t_{ij})^2}, \tag{8}$$

where s^t_{ij} and $\hat{s}^t_{ij}$ are the elements of matrices S^t and $\hat{S}^t$ respectively, $t = 1, 2, \ldots, T$.

We consider the case where the first task consists of 50 classes, and 10 classes are in each of the following tasks, providing the comparison with iCaRL [6], LUCIR [27], PODNet [29] and AFC [38] (see Table 1 to recall different assumptions for each method). However, we also show that when using IS for task selection we require a larger batch size to improve task identification (60 test samples).

The maxoutput strategy does not work well in this case because most of the parameters are assigned to the first task (with 50 classes). As a result, this strategy predicts the first task when considering the last tasks for almost every batch, as shown in Appendix A.

As a final comment, we also investigated an alternative solution when we have a first task that is significantly larger than the following ones. This can be solved by

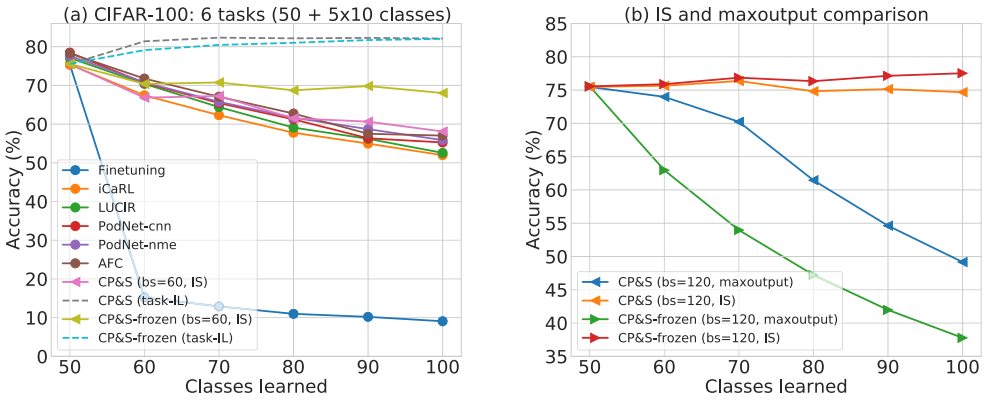

Fig. 7 Accuracy history for ResNet-32 trained with CP&S and state-of-the-art. The pruning parameter is $\alpha_{conv} = 0.9$ for CP&S strategy, and "task-IL" refers to the same upper bound mentioned in the previous figures. Three different class orderings are used

pretraining convolutional weights with the first task and training only the task-specific parts in the network. We denoted this last strategy as "CP&S-frozen" since we pretrain *all* convolutional parameters with the first 50 classes, and, for the next tasks we train task-related batch normalization parameters and the fully connected part that is task-specific by construction. So, in this last strategy each subnetwork consist of a common convolutional part (pretrained on the first task), batch normalization layers and output classification head. We present an additional task-selection accuracy comparison between IS and maxoutput in Appendix A. We again observe poor performance for maximum output response strategy. The final results can be seen in Fig. 7 and Table 3.

We believe this knowledge transfer strategy might be interesting to explore in the future, where the first heads of the network specialize in selecting tasks and the deeper layers specialize in class prediction for each task.

Knowledge transfer Let us explore how many parameters are used by every task in the case of ResNet-18 on ImageNet-1000. We consider the union and the intersection of all masks as sets. In Fig. 8(a) we show how the union and intersection are distributed across parameters after the last task is learned in the case of ResNet-18 on ImageNet-1000. From the union, we observe that the last layers are almost fully occupied in contrast to the first layers. From the intersection, it can be seen that a significant fraction of parameters is shared between each of the tasks across all layers. At the same time, about 85% are assigned to two and more tasks (Fig. 8(b)). Notably, 35% of parameters are shared across all ten tasks and about 50% of parameters are used for nine and more tasks. From these figures, we can conclude that almost all parameters are occupied at the end, having significant overlaps between subnetworks. However, looking at Fig. 5, we see that performance remains stable, without drops. This allows us to conclude that subnetworks share knowledge between tasks, which helps to assimilate new patterns.

Table 3 Comparison between algorithms by average incremental accuracy and backward transfer metric at the end of all tasks

Method	AIA (%)	BWT (%)
iCaRL [6]	61.63 ± 0.25	12.77 ± 0.30
LUCIR [27]	63.29 ± 0.36	10.09 ± 0.12
PODNet-CNN [29]	64.56 ± 0.28	11.90 ± 0.04
PODNet-NME [29]	65.07 ± 0.44	**1.18** ± 0.16
AFC [38]	65.73 ± 0.09	7.46 ± 0.38
CP&S (bs=60, IS)	64.97 ± 4.23	11.68 ± 0.20
CP&S-frozen (bs=60, IS)	**70.55** ± 5.05	4.90 ± 2.35

Mean values and standard deviation are computed using three different orderings

Bold numbers indicate the best performance

6 Conclusion

To overcome the problem of catastrophic forgetting while learning new tasks, we propose a continual learning algorithm that trains subnetworks for each task. During training of a task, weights are pruned and then fixed, such that future tasks cannot destroy the weights in this subnetwork, while still being able to use them for other subnetworks. During evaluation of new data, the correct task-ID and associated subnetwork have to be inferred from a small batch of samples. We describe existing task-prediction approaches and propose a new one based on the neural connection strength. Although the task-ID needs to be inferred, no memory is needed to store examples from previous tasks, unlike alternative approaches. The main drawback of the current implementation of the CP&S strategy is the need to have a small batch of test data due to the difficulty of determining the correct task-ID – a limitation also observed in the best-performing methods in the literature. Notwithstanding, our work demonstrates that combining subnetwork creation and subnetwork selection methods into one paradigm provides a general approach

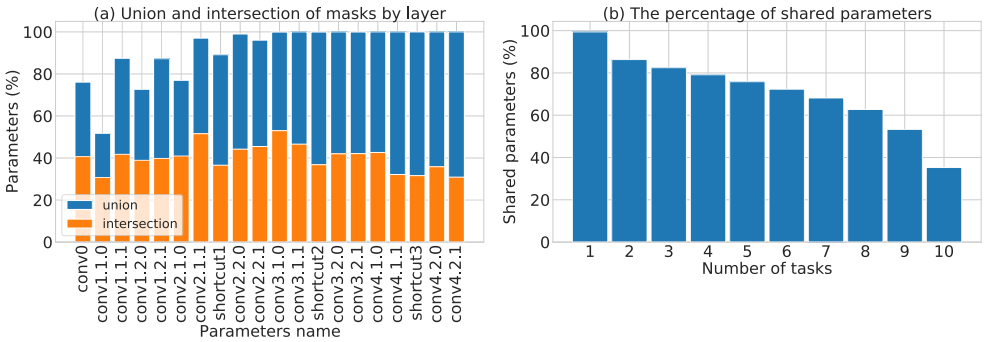

Fig. 8 Visualization of employed masks and shared parameters for ResNet-18 on ImageNet-1000

to solve class-IL problems. We believe the proposed strategy can be further improved by developing better task-prediction strategies that do not need a batch of test data. CP&S outperforms all state-of-the-art methods on a variety of datasets. For ImageNet-1000, we show an improvement of more than 10% accuracy when compared to previous algorithms. Even though we apply CP&S to image classification tasks, no additional limitations are foreseen when applying it to other machine learning problems.

Appendix A: Additional information on CIFAR-100 experiments

Task-selection We present CP&S results with different test batch sizes and task-selection strategies in Fig. 9.

Also, we provide an additional comparison between maxoutput and IS strategies in Figs. 10 and 11. In both cases, we observe the advantage of importance scores (IS) over maxoutput strategy in the case of imbalanced tasks.

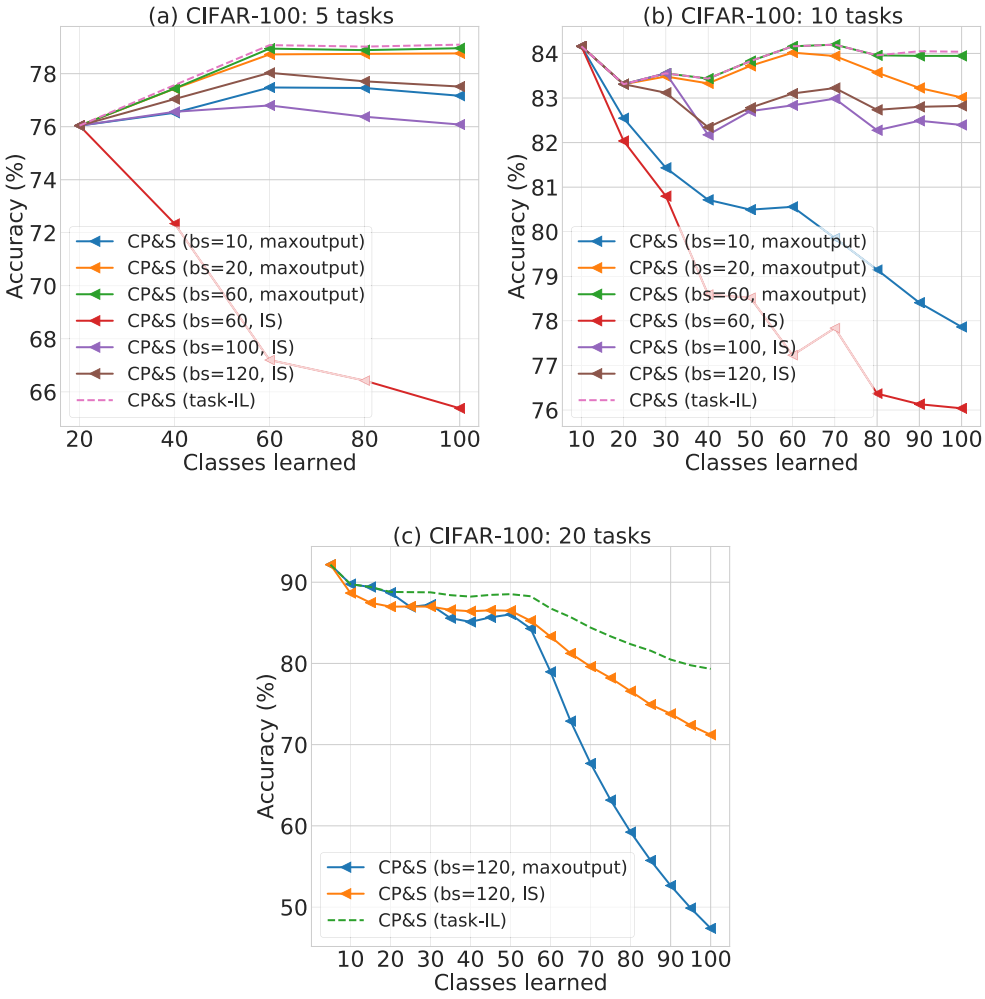

Fig. 9 The performance of CP&S with different batch sizes and task-selection strategies

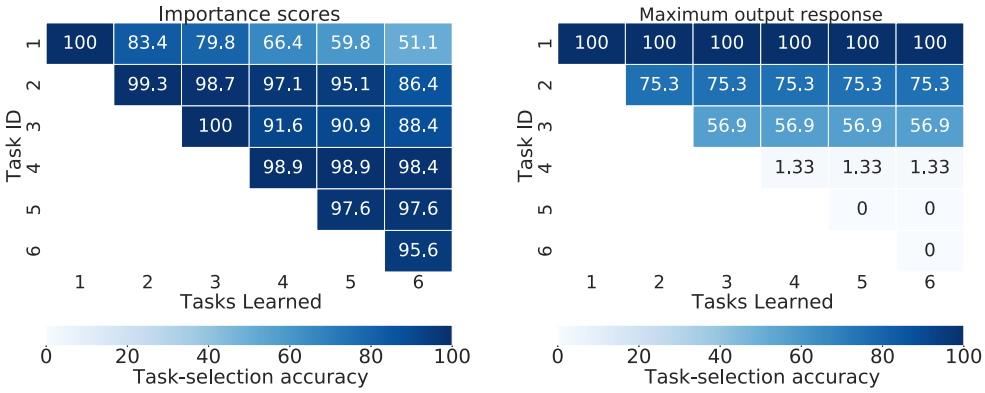

Fig. 10 Task-selection accuracy using Importance Scores (IS) (**left**) as opposed to maxoutput (**right**) on CIFAR-100 with class imbalance (50 classes in the first task and 10 classes in each of the following five tasks) for CP&S. The test batch size is 60 images in both cases

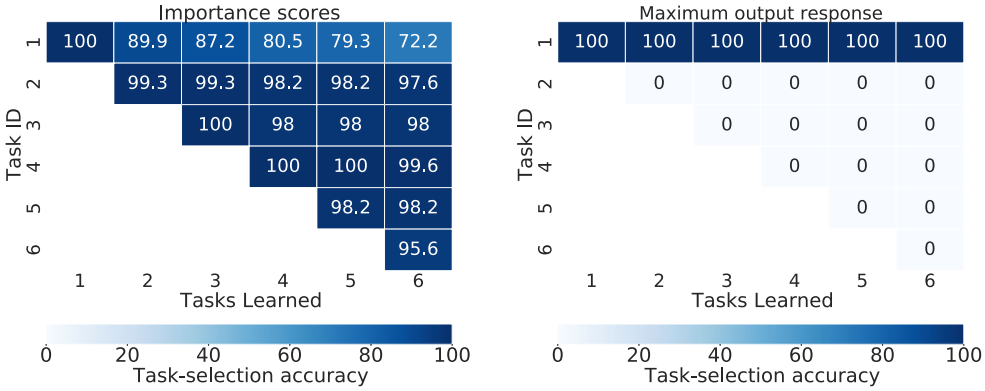

Fig. 11 Task-selection accuracy using Importance Scores (IS) (**left**) as opposed to maxoutput (**right**) on CIFAR-100 with class imbalance (50 classes in the first task and 10 classes in each of the following five tasks) for CP&S-frozen. The test batch size is 60 images in both cases

Table 4 Hyperparameters for (ResNet-18)/3 training on CIFAR-100 (5/10/20 tasks)

Method	# epochs	Optimizer	LR	LR scheduler	Weight decay
iTAML	70	RAdam [62]	0.01	on epochs 20, 40, 60 multiply LR by 0.2	0
CP&S (ours)	70	Adam	0.01	on epochs 20, 40, 60 multiply LR by 0.2	0.0005

Table 5 Hyperparameters for ResNet-32 training on CIFAR-100 (6 tasks)

Method	# epochs	Optimizer	LR	LR scheduler	Weight decay
iCaRL	70	SGD	2.0	on epochs 49 and 63 multiply LR by 0.2	0.00005
LUCIR	160	SGD	0.1	on epochs 80, 120 multiply LR by 0.1	0.0005
PODNet	160	SGD	0.1	on epochs 80, 120 multiply LR by 0.1	0.0005
AFC	160	SGD	0.1	on epochs 80, 120 multiply LR by 0.1	0.0005
CP&S (ours)	160	Adam	0.001	on epochs 80, 120 multiply LR by 0.1	0.0005

Training hyperparameters In Table 4, we show the hyperparameters that we used for experiments on CIFAR-100 in Section 4. For iTAML, all the parameters are taken from the original work and the results were reproduced using the official GitHub repository. Memory buffer contains 2000 training samples to mitigate forgetting. For CP&S, we used 3 pruning iterations, 1000 training samples per task to estimate importance scores in NNrelief and $\alpha_{conv} = 0.9$. For retraining (after pruning sep), we use 40 epochs with Learning Rate (LR) 0.01 multiplied by 0.2 on epochs 15, 25 and 40.

In Table 5, we present the training hyperparameters for experiments in Section 5. To reproduce the results, we use PODNet and AFC GitHub repositories using the hyperparameters from the original works. All the previous works use 2000 training samples in the fixed-size memory buffer to mitigate forgetting. For CP&S, we used 1 pruning iteration, 1000 training samples per task to estimate importance scores in NNrelief and $\alpha_{conv} = 0.9$. For retraining (after the pruning step), we use 50 epochs with LR 0.001 multiplied by 0.1 on epochs 20 and 40.

Appendix B: ImageNet-100/1000 results

For ImageNet-100/1000, we present exact numbers from which the plots are constructed for CP&S in Tables 6 and 7.

Table 6 ImageNet-100 results with different test batch sizes and task-IL scenario trained with SGD and Adam

Optimizer	Batch size	1	2	3	4	5	6	7	8	9	10
Adam	20	98.20	98.80	98.67	98.50	98.48	98.60	98.63	98.63	98.50	98.38
	10	98.20	98.80	98.67	98.41	98.25	98.15	97.90	97.94	97.23	97.00
	5	98.20	98.02	97.03	95.86	94.23	92.51	92.08	92.12	90.53	89.39
	task-IL	98.20	98.80	98.67	98.50	98.48	98.60	98.63	98.63	98.50	98.38
SGD	20	99.00	98.90	98.67	98.60	97.90	98.09	98.05	98.25	94.20	92.62
	task-IL	99.00	98.90	98.67	98.60	98.20	98.33	98.26	98.43	98.20	98.06

Table 7 ImageNet-1000 results with different test batch sizes and task-IL scenario trained with SGD

Optimizer	Batch size	1	2	3	4	5	6	7	8	9	10
SGD	50	94.38	94.96	94.52	94.42	94.40	94.45	94.40	94.16	93.88	93.77
	20	94.38	94.96	94.52	94.42	94.40	94.40	94.34	94.12	93.77	93.66
	10	94.38	94.96	94.42	94.09	93.91	93.70	93.40	92.92	92.19	91.97
	5	94.38	94.31	92.74	91.53	90.41	89.24	88.07	86.59	84.82	83.92
	task-IL	94.38	94.96	94.52	94.42	94.40	94.45	94.40	94.16	93.88	93.77

Appendix C: CUB-200-2011 additional comparison

In this section, we provide an additional comparison for ResNet-18 on CUB-200-2011 dataset using 5 test images per batch to predict the task-ID in Fig. 12.

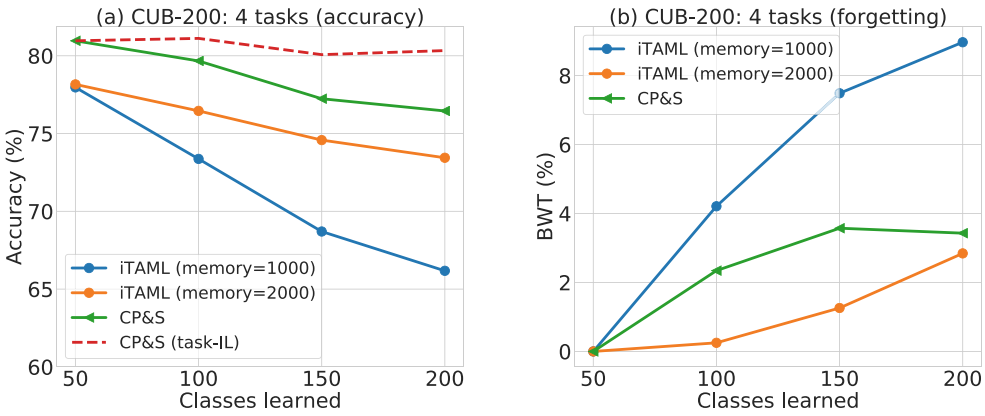

Fig. 12 Comparison with iTAML on four tasks constructed from CUB-200-2011. Notation: "memory" is the number for images from previous tasks; "task-IL" refers to task-IL scenario as an upper-bound for CP&S

Acknowledgements The authors would like to thank SURFsara for providing the access to Snellius HPC cluster. A preprint version of this work is published on arXiv under the CC BY license: Dekhovich, A., Tax, D. M., Sluiter, M. H., & Bessa, M. A. Continual Prune-and-Select: Class-incremental learning with specialized subnetworks. arXiv preprint arXiv:2208.04952 (2022).

Author Contributions Not applicable

Funding Not applicable

Data Availability Not applicable

Code Availability PyTorch [1] implementation of the code is available at: https://github.com/adekhovich/continual_prune_and_select

Declarations

Conflict of Interests The authors have no conflicts of interest to declare.

Ethics approval Not applicable

Consent to participate Not applicable

Consent for Publication Not applicable

References

1. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems, vol 32
2. French RM (1999) Catastrophic forgetting in connectionist networks. Trends Cognit Sci 3(4):128–135
3. Goodfellow IJ, Mirza M, Xiao D, Courville A, Bengio Y (2014) An empirical investigation of catastrophic forgetting in gradient-based neural networks. In: 2nd International conference on learning representations, ICLR
4. Thrun S (1998) Lifelong learning algorithms. In: Learning to learn. Springer, Boston. MA, pp 181–209
5. Masana M, Liu X, Twardowski B, Menta M, Bagdanov AD, Van De Weijer J (2020) Class-incremental learning: survey and performance evaluation on image classification. IEEE Trans Pattern Anal Mach Intell
6. Rebuffi S-A, Kolesnikov A, Sperl G, Lampert CH (2017) icarl: incremental classifier and representation learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2001–2010
7. Shmelkov K, Schmid C, Alahari K (2017) Incremental learning of object detectors without catastrophic forgetting. In: Proceedings of the IEEE international conference on computer vision, pp 3400–3409
8. Zhang J, Zhang J, Ghosh S, Li D, Tasci S, Heck L, Zhang H, Kuo C-CJ (2020) Class-incremental learning via deep model consolidation. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 1131–1140
9. Michieli U, Zanuttigh P (2019) Incremental learning techniques for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision workshops
10. Yan S, Zhou J, Xie J, Zhang S, He X (2021) An em framework for online incremental learning of semantic segmentation. In: Proceedings of the 29th ACM international conference on multimedia, pp 3052–3060
11. Van De Ven GM, Siegelmann HT, Tolias AS (2020) Brain-inspired replay for continual learning with artificial neural networks. Nature Commun 11(1):1–14
12. Lerner Y, Honey CJ, Silbert LJ, Hasson U (2011) Topographic mapping of a hierarchy of temporal receptive windows using a narrated story. J Neurosci 31(8):2906–2915
13. Zadbood A, Chen J, Leong YC, Norman KA, Hasson U (2017) How we transmit memories to other brains: constructing shared neural representations via communication. Cerebral cortex 27(10):4988–5000

14. Huttenlocher PR (1990) Morphometric study of human cerebral cortex development. Neuropsychologia 28(6):517–527
15. Lennie P (2003) The cost of cortical computation. Current Biol 13(6):493–497
16. Attwell D, Laughlin SB (2001) An energy budget for signaling in the grey matter of the brain. J Cerebral Blood Flow Metabolism 21(10):1133–1145
17. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
18. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee, pp 248–255
19. Wah C, Branson S, Welinder P, Perona P, Belongie S (2011) The caltech-ucsd birds-200-2011 dataset. Tech Report CNS-TR-2011-001, California institute of technology
20. Delange M, Aljundi R, Masana M, Parisot S, Jia X, Leonardis A, Slabaugh G, Tuytelaars T (2021) A continual learning survey: defying forgetting in classification tasks. IEEE Trans Pattern Anal Mach Intell
21. Zenke F, Poole B, Ganguli S (2017) Continual learning through synaptic intelligence. In: International conference on machine learning. PMLR, pp 3987–3995
22. Li Z, Hoiem D (2017) Learning without forgetting. IEEE Trans Pattern Anal Mach Intell 40(12):2935–2947
23. Dhar P, Singh RV, Peng K-C, Wu Z, Chellappa R (2019) Learning without memorizing. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5138–5146
24. Liu X, Masana M, Herranz L, Van De Weijer J, Lopez AM, Bagdanov AD (2018) Rotate your networks: better weight consolidation and less catastrophic forgetting. In: 2018 24th International conference on pattern recognition (ICPR). IEEE, pp 2262–2268
25. Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A et al (2017) Overcoming catastrophic forgetting in neural networks. Proc National Acad Sci 114(13):3521–3526
26. Aljundi R, Babiloni F, Elhoseiny M, Rohrbach M, Tuytelaars T (2018) Memory aware synapses: learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV), pp 139–154
27. Hou S, Pan X, Loy CC, Wang Z, Lin D (2019) Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 831–839
28. Belouadah E, Popescu A (2019) Il2m: class incremental learning with dual memory. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 583–592
29. Douillard A, Cord M, Ollion C, Robert T, Valle E (2020) Podnet: pooled outputs distillation for small-tasks incremental learning. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, 23–28 Aug 2020, proceedings, Part XX 16. Springer, pp 86–102
30. Yoon J, Yang E, Lee J, Hwang SJ (2018) Lifelong learning with dynamically expandable networks. In: 6th International conference on learning representations, ICLR
31. Wortsman M, Ramanujan V, Liu R, Kembhavi A, Rastegari M, Yosinski J, Farhadi A (2020) Supermasks in superposition. In: Advances in neural information processing systems
32. Sokar G, Mocanu DC, Pechenizkiy M (2021) Spacenet: make free space for continual learning. Neurocomputing 439:1–11
33. Chaudhry A, Dokania PK, Ajanthan T, Torr PH (2018) Riemannian walk for incremental learning: understanding forgetting and intransigence. In: Proceedings of the European conference on computer vision (ECCV), pp 532–547
34. Shin H, Lee JK, Kim J, Kim J (2017) Continual learning with deep generative replay. In: Advances in neural information processing systems
35. Mensink T, Verbeek J, Perronnin F, Csurka G (2012) Metric learning for large scale image classification: generalizing to new classes at near-zero cost. In: European conference on computer vision. Springer, pp 488–501
36. Castro FM, Marín-Jiménez MJ, Guil N, Schmid C, Alahari K (2018) End-to-end incremental learning. In: Proceedings of the European conference on computer vision (ECCV), pp 233–248
37. Wu Y, Chen Y, Wang L, Ye Y, Liu Z, Guo Y, Fu Y (2019) Large scale incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 374–382
38. Kang M, Park J, Han B (2022) Class-incremental learning by knowledge distillation with adaptive feature consolidation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 16071–16080
39. Rajasegaran J, Hayat M, Khan S, Khan FS, Shao L (2019) Random path selection for incremental learning. In: Advances in neural information processing systems
40. Yan S, Xie J, He X (2021) Der: dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3014–3023
41. Wang FL, Zhou D-W, Ye H-J, Zhan D-C (2022) Foster: feature boosting and compression for class-incremental learning. In: European conference on computer vision
42. Rajasegaran J, Khan S, Hayat M, Khan FS, Shah M (2020) itaml: an incremental task-agnostic meta-learning approach. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13588–13597
43. Han S, Pool J, Tran J, Dally WJ (2015) Learning both weights and connections for efficient neural networks. In: Advances in neural information processing systems, pp 1135–1143
44. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2017) Pruning filters for efficient convnets. In: 5th international conference on learning representations, ICLR
45. Frankle J, Carbin M (2019) The lottery ticket hypothesis: finding sparse, trainable neural networks. In: 7th International conference on learning representations, ICLR
46. Hu H, Peng R, Tai Y-W, Tang C-K (2016) Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv:1607.03250
47. Huang Z, Wang N (2018) Data-driven sparse structure selection for deep neural networks. In: Proceedings of the European conference on computer vision (ECCV), pp 304–320
48. Luo J-H, Wu J, Lin W (2017) Thinet: a filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, pp 5058–5066
49. Dekhovich A, Tax DM, Sluiter MH, Bessa MA (2021) Neural network relief: a pruning algorithm based on neural activity. arXiv:2109.10795
50. LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Advances in neural information processing systems, pp 598–605
51. Hassibi B, Stork DG, Wolff GJ (1993) Optimal brain surgeon and general network pruning. In: IEEE international conference on neural networks. IEEE, pp 293–299
52. Hassibi B, Stork DG (1993) Second order derivatives for network pruning: optimal brain surgeon. In: Advances in neural information processing systems, pp 164–171
53. Lebedev V, Lempitsky V (2016) Fast convnets using group-wise brain damage. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2554–2564
54. Mallya A, Lazebnik S (2018) Packnet: adding multiple tasks to a single network by iterative pruning. In: Proceedings of the

IEEE conference on computer vision and pattern recognition, pp 7765–7773
55. Golkar S, Kagan M, Cho K (2019) Continual learning via neural pruning. In: NeurIPS workshop on real neurons & hidden units
56. Mallya A, Davis D, Lazebnik S (2018) Piggyback: adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European conference on computer vision (ECCV), pp 67–82
57. Hung C-Y, Tu C-H, Wu C-E, Chen C-H, Chan Y-M, Chen C-S (2019) Compacting, picking and growing for unforgetting continual learning. In: Advances in neural information processing systems, vol 32
58. Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Pascanu R, Hadsell R (2016) Progressive neural networks. arXiv:1606.04671
59. Kim ES, Kim JU, Lee S, Moon S-K, Ro YM (2020) Class incremental learning with task-selection. In: 2020 IEEE international conference on image processing (ICIP). IEEE, pp 1846–1850
60. Lopez-Paz D, Ranzato M (2017) Gradient episodic memory for continual learning. In: Advances in neural information processing systems, vol 30, pp 6467–6476
61. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd International conference on learning representations, ICLR
62. Liu L, Jiang H, He P, Chen W, Liu X, Gao J, Han J (2020) On the variance of the adaptive learning rate and beyond. In: 8th International conference on learning representations, ICLR

Affiliations

Aleksandr Dekhovich[1] · David M.J. Tax[2] · Marcel H.F Sluiter[1] · Miguel A. Bessa[3]

[1] Department of Materials Science and Engineering, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands

[2] Pattern Recognition and Bioinformatics Laboratory, Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands

[3] School of Engineering, Brown University, 184 Hope St., Providence, RI 02912, USA