This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

Selective Freezing for Efficient Continual Learning

Amelia Sorrenti, Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi,
Concetto Spampinato, Simone Palazzo
PeRCeiVe Lab, University of Catania
Catania, Italy
www.perceivelab.com

Abstract

This paper aims to tackle the challenges of continual learning, where sequential learning from a stream of tasks can lead to catastrophic forgetting. Simultaneously, it addresses the need to reduce the computational demands of large-scale deep learning models to mitigate their environmental impact. To achieve this twofold objective, we propose a method that combines selective layer freezing with fast adaptation in a continual learning context. We begin by conducting an extensive analysis of layer freezing in continual learning, revealing that certain configurations allow for freezing a substantial portion of the model without significant accuracy degradation. Leveraging this insight, we introduce a novel approach that optimizes plasticity on new tasks while preserving stability on previous tasks by dynamically identifying a subset of layers to freeze during training. Experimental results demonstrate the effectiveness of our approach in achieving competitive performance with manually-tuned freezing strategies. Moreover, we quantitatively estimate the reduction in computation and energy requirements achieved through our freezing strategy by considering the number of parameters and updates required for model training.

1. Introduction

In recent years, deep learning has made tremendous strides in various application domains, achieving state-of-the-art performance in fields such as computer vision [14, 13, 11, 15] and natural language processing [10, 3, 32]. However, the ability to continuously learn from a stream of tasks, referred to as continual learning, has not experienced a comparable level of advancement. Continual learning is still a major challenge for deep learning models, with catastrophic forgetting, a phenomenon where the model forgets the knowledge of previous tasks while learning new ones, being a primary roadblock [8, 28].

Simultaneously, there has been an escalating concern within the research community around the environmental implications of training large-scale deep learning models. These models, especially those at the forefront of performance in various fields, often require substantial computational resources for training. As these computational demands increase, so does the associated energy consumption, leading to a significant environmental footprint [29]. Therefore, looking for ways to reduce the computational demands of these models without a substantial loss in performance has become an imperative need.

In this paper, we begin by presenting an exhaustive analysis of the effect of layer freezing in a continual learning context. Our findings reveal that there exist configurations that allow us to freeze a significant portion of the model without suffering a notable decrease in accuracy. This shows that, contrary to common assumptions, not all parts of a neural network need to be actively trained at all times.

Then, we propose a method designed to address the twin challenges of continual learning and computational efficiency. Our approach leverages the intuition that suitable freezing of layers in deep learning models during training can support both objectives, by providing a mechanism to preserve important features from previous tasks, while reducing the amount of computation required, which directly affects the energy impact of model training. In detail, at the beginning of each continual learning task at training time, we perform a fast adaptation stage of the model on the new task, by testing different freezing strategies to dynamically identify a subset of layers that optimizes plasticity on the new task and stability on the previous ones. Experimental results show that our approach achieves performance that is competitive with manually-tuned optimal freezing strategies. This eliminates the need for time-consuming and expertise-intensive hyperparameter tuning, further augmenting the computational efficiency of our approach. Additionally, we show that our freezing strategy effectively reduces the amount of computation and energy requirements, which we quantitatively estimate in terms of number of parameters and number of updates required to train a model.

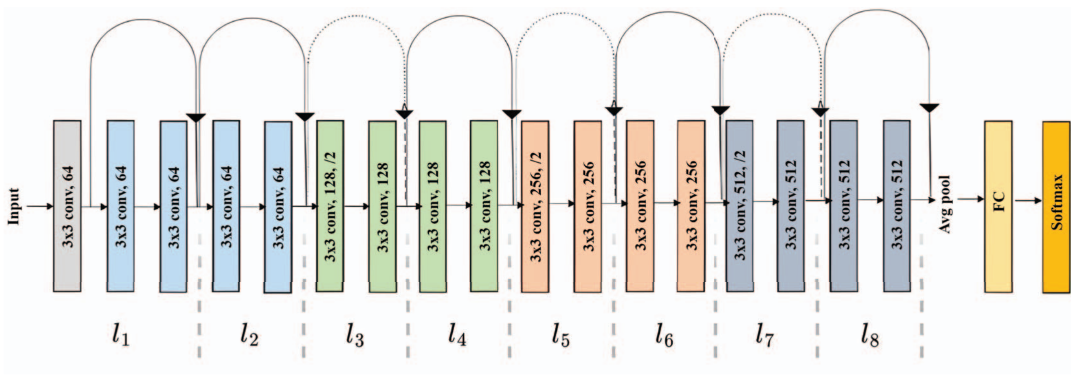

Figure 1: **ResNet-18 architecture and selective freezing strategy**. After an initial convolutional layer (apart from the last fully-connected layer), each colored portion represents a network's *main block*, consisting of two residual *basic blocks*, each applying two convolutions. For our selective freezing strategy, we treat each basic block as the smallest unit of freezing, named *layer*.

The results presented herein may serve as a stepping stone towards more sustainable and efficient deep learning, without compromising the power of continual learning. This aligns with the broader goal of achieving both environmentally-friendly and effective machine learning solutions, bringing us closer to fully realizing the transformative potential of deep learning.

2. Related work

Early work by McCloskey and Cohen revealed catastrophic forgetting in Artificial Neural Networks (ANNs) [28], a problem that persists despite recent deep learning advancements [39, 41]. To address this, Continual Learning (CL) methods were developed, aiming to uphold model accuracy on prior data while learning from evolving inputs [40, 30, 23]. Rehearsal-based strategies maintain a memory of past examples, interleaving them with new data [33, 7, 4, 2, 1, 5]. Pseudo-rehearsal methods use a generative model for this [35, 38]. Some studies adopt learning objectives to retain performance on prior tasks [34, 16, 24, 43], while others employ semi-supervised techniques to derive general features [6, 31, 20].

Other approaches in the literature have investigated strategies based on the freezing of network parameters. Once a task has been learned, parameter freezing allows the acquired knowledge to remain fixed, thus reducing forgetting of previous tasks. Most works have focused on defining *mask-based* methods, where a learned mask for individual weights or groups of weights is used to selectively freeze or constrain certain parameters. Piggyback [25] uses a binary mask on the weights of a pre-trained model to create different sub-networks, introducing an overhead of 1 bit per network parameter for each task. Similarly, Kang et al. [19] introduced Winning SubNetworks (WSN), which sequentially learns and selects an optimal subnetwork for each task by means of an *accumulate binary mask*. WSN updates only weights that have not been selected in previous tasks, resulting in a task-specific subnetwork. HAT [36] learns near-binary attention vectors by using gated task embeddings for each task. These vectors are then used to define hard attention masks for each task, which are used to constrain the network's weight updates for the following task. Masana et al. [26] proposed a ternary mask-based approach for the task-incremental learning scenario. Differently from the above-mentioned approaches, these masks are applied to the features of each layer rather than weights, reducing the number of mask parameters for each new task. Sparse-MAML [42] proposes a meta-learning approach, where a subset of weights is frozen during the inner-loop learning process. Similarly to the previous methods, a binary mask is learned and then multiplied element-wise with gradient updates. The PathNet algorithm [12] employs agents embedded in the neural network to identify which parts can be reused for new tasks. Task-relevant paths that evolved during the previous tasks can be frozen or partially reused in the following tasks. Shi et al. [37] introduced the BLIP approach, which preserves the information gain on model parameters provided by each task through a guided bit freezing. In particular, weight quantization is exploited to determine the weights to be frozen to prevent forgetting. Jung et al. [18] defined an algorithm that prevents the model from drifting by freezing the weights associated with the information learned in previous tasks, called *nodes*, while learning future tasks. Specifically, the loss function involves two group sparsity-based regularization terms that are used to define the importance of a node for carrying out the preceding tasks.

Yang et al. [44] proposed a progressive task-correlated layer freezing method to be used in the context of self-supervised continual learning (SSCL). More specifically, a task correlation ratio, based on the gradient projection norm, is defined to formally characterize the correlation between current and previous tasks.

In this work, we perform an empirical analysis of the effect of layer freezing on the backbone of state-of-the-art continual learning methods, studying the impact of freezing both at different depths of the network and at different "times", i.e., continual learning tasks. Then we present a novel approach, named *selective freezing*, to dynamically freeze layer weights based on a preliminary assessment of the new task, offering a potentially more efficient and flexible means of preserving knowledge between tasks.

3. Method

We hereby introduce and describe *selective freezing*, an approach aimed at gradually and adaptively freezing a subset of a model's parameters over a sequence of continual learning tasks, with the objective of encouraging feature reuse as well as reducing computational costs.

3.1. Problem formulation

Following the established literature, we pose continual learning as a supervised classification problem on a non-i.i.d. stream of data, with the assumption that *task boundaries*, marking changes in the data distributions, are known at training time. More formally, let $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ be a sequence of data streams, where each pair $(\mathbf{x}, y) \sim \mathcal{D}_i$ denotes a data point $\mathbf{x} \in \mathcal{X}$ with the corresponding class label $y \in \mathcal{Y}$; the sample distributions (in terms of both the data point distribution and the class label distribution) of different $\mathcal{D}_i$ and $\mathcal{D}_j$ may vary — for instance, class labels from $\mathcal{D}_i$ might be different from those from $\mathcal{D}_j$. Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}$, the objective of continual learning is to train f on $\mathcal{D}$, organized as a sequence of T tasks $\{\tau_1, \ldots, \tau_T\}$, under the constraint that, at a generic task τ_i, the model receives inputs sampled from the corresponding data distribution only, i.e., $(\mathbf{x}, y) \sim D_i$. The classification model may also keep a limited *memory buffer* $\mathbf{M}$ of past samples, to reduce forgetting of features from previous tasks. The model update step between tasks can be summarized as:

$$\langle f, \boldsymbol{\theta}_{i-1}, \mathbf{M}_{i-1} \rangle \xrightarrow{\mathcal{D}_i} \langle f, \boldsymbol{\theta}_i, \mathbf{M}_i \rangle, \tag{1}$$

where $\boldsymbol{\theta}_i$ and $\mathbf{M}_i$ represent the set of model parameters and the memory buffer at the end of task τ_i.

The training objective is to optimize a classification loss over the sequence of tasks (without losing accuracy on past tasks) by the model instance at the end of training:

$$\underset{\boldsymbol{\theta}_T}{\arg\min} \sum_{i=1}^{T} \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}_i} \Big[\mathcal{L}\Big(f\left(\mathbf{x}; \boldsymbol{\theta}_T\right), y \Big) \Big], \tag{2}$$

where $\mathcal{L}$ is a generic classification loss (e.g., cross-entropy), which a continual learning model attempts to optimize while accounting for model *plasticity* (the capability to learn current task data) and *stability* (the capability to retain knowledge of previous tasks) [28].

3.1.1 Selective freezing

In accordance to CLS theory [22, 27], we propose a method for parameter freezing to maximize stability and plasticity. Specifically, we propose to train the model at the beginning of each task for a limited number of iterations under varying parameter freezing settings, providing an opportunity to the model to find the optimal configuration that combines retaining of previous knowledge and learning of the new task.

Formally, we want to model the joint probability between task data $\mathcal{D}_i$, previous experience $\mathbf{M}_{i-1}$, model parameters $\boldsymbol{\theta}_i$ and a binary freezing mask $\mathbf{m}_i$, with the same dimensions as $\boldsymbol{\theta}_i$ and such that $m_{i,j} = 1$ indicates that parameter $\theta_{i,j}$ should be frozen:

$$P(\mathbf{x}, y, \boldsymbol{\theta}_i, \mathbf{m}_i) = P\left(y \mid \mathbf{x}, f\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i, \mathbf{m}_i) P(\mathbf{x}), \tag{3}$$

where $\mathbf{x}$ and y represent samples and labels from $\mathcal{D}_i \cup \mathbf{M}_{i-1}$. The first term of the decomposition of Eq. 3 is the likelihood of correct labels given the input and the model prediction, while the joint distribution $P(\boldsymbol{\theta}_i, \mathbf{m}_i)$ describes the relation between model parameters $\boldsymbol{\theta}_i$ and the freezing strategy defined by $\mathbf{m}_i$. Assuming the independence between $\boldsymbol{\theta}_i$ and $\mathbf{m}_i$, this distribution can be expressed as:

$$P(\boldsymbol{\theta}_i, \mathbf{m}_i) = P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{m}_i), \tag{4}$$

where

$$P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) = \prod_j \mathcal{N}\left(\theta_{i,j}; \theta_{i-1,j}, \sigma_i^2)\right)^{1-m_{i,j}}. \tag{5}$$

In this formulation, we model the distribution of each parameter $\theta_{i,j}$ as a Gaussian distribution depending on the corresponding mask value $m_{i,j}$, which removes a term from the overall probability when $m_{i,j} = 1$. Note that the mean of each parameter is set to $\theta_{i-1,j}$, i.e., its value at the end of the previous task (or to 0 for the first task, based on common initialization strategies).

In order to model $P(\mathbf{m}_i)$ in a practically feasible way, we employ some simplifying assumption based on the layered structure of deep learning models. Given $f = l_1 \circ l_2 \circ \ldots \circ l_L$, where each l_k represents a network layer with parameters $\boldsymbol{\theta}_{|k}$ and $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_{|1}, \ldots, \boldsymbol{\theta}_{|L}\right]$, let us similarly define $\mathbf{0}_{|k}$ and $\mathbf{1}_{|k}$ as two tensors with the same size as $\boldsymbol{\theta}_{|k}$, with all values set to 0 and 1, respectively. Then, we impose that possible values for $\mathbf{m}_i$ must be parameterized by a value l as follows:

$$\mathbf{m}_i(l) = \left[\mathbf{1}_{|1}, \ldots, \mathbf{1}_{|l}, \mathbf{0}_{|l+1}, \ldots, \mathbf{0}_{|L}\right] \vee \mathbf{m}_{i-1} \tag{6}$$

with $l \in \{1, \ldots, L\}$. In practice, parameters frozen at previous tasks must remain so at the current task, and a layer's parameters can only be frozen altogether if all previous layers are also frozen.

Given these constraints, our goal is to find the optimal binary mask $\mathbf{m}_i$ that maximizes the likelihood of the labels y given the inputs $\mathbf{x}$ from current task $\mathcal{D}_i$ and from long-term memory $\mathbf{M}_{i-1}$.This is expressed as the following optimization problem:

$$\arg\max_{\mathbf{m}_i, \boldsymbol{\theta}_i} P\left(y \mid \mathbf{x}, f\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{m}_i) P(\mathbf{x}), \tag{7}$$

where the optimization is over parameters $\boldsymbol{\theta}_i$ and all feasible binary masks $\mathbf{m}_i$. The choice of $\mathbf{m}_i$ is thus carried out by maximizing this likelihood through the optimization of the selective freezing loss function $\mathcal{L}_{\text{sf}}$:

$$\begin{aligned} \mathcal{L}_{\text{sf}} = & \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i}\left[\mathcal{L}\left(y, f\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right] \\ & + \alpha \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{M}_{i-1}}\left[\mathcal{L}\left(y, f\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right], \end{aligned} \tag{8}$$

where $\mathbf{m}_i$ varies as described above, and α is a weighing factor between data sources. It is important to notice that, while optimizing for $\mathbf{m}_i$ necessarily requires updating $\boldsymbol{\theta}_i$ as well (since freezing, per se, does not alter inference performance), the objective is to prepare the model by identifying the optimal set of parameters that should be kept from previous tasks in a way that ensures both knowledge retainment and room for plasticity. For this reason, optimization is carried out for a single epoch over $\mathcal{D}_i$. Note that the choice of $\mathcal{L}$ is arbitrary: the proposed formulation allows for plugging in any existing continual learning method, enhancing it with the proposed training strategy.

4. Experimental results

4.1. Datasets and metrics

In our experiments, in order to define a set of continual learning tasks $\{\tau_1, \ldots, \tau_T\}$, we employ the Seq-CIFAR10 [4] dataset, which is obtained splitting the CIFAR-10 dataset [21]. The original dataset consists of 10 classes organized into 6,000 32x32 color images per class. These classes are then divided into 5,000 images for training and 1,000 for test. Seq-CIFAR10 is composed of the original dataset, divided into $T = 5$ continual learning tasks, each comprising two classes.

For the experiments with a pre-trained backbone, the datasets used for the pre-training phase are ImageNet [9], CIFAR-100 [21] and ImageNet-10 [17].

The proposed approach and the related baselines have been evaluated in the standard *class-incremental learning* (Class-IL) setting, which is asked to gradually solve the complete problem while classes become available at different times. For the performance evaluation, we report the average classification accuracy over all dataset classes in the test set; we assume that the model has no access to task identity when classifying a given input.

Pretraining	**ImageNet-1k**		**CIFAR-100**		**ImageNet-10**	
Layer	**200**	**500**	**200**	**500**	**200**	**500**
l_1	69.84	80.31	55.63	67.42	67.87	74.98
l_2	70.18	78.44	56.07	67.39	69.52	75.09
l_3	69.27	78.95	57.96	68.15	70.02	74.34
l_4	70.34	80.08	54.79	66.51	68.02	74.17
l_5	70.10	80.85	54.62	67.29	71.82	75.40
l_6	69.06	76.69	53.66	63.56	71.64	75.48
l_7	62.02	75.68	43.08	55.39	71.49	77.23
l_8	10.13	9.09	7.92	10.18	71.44	74.47
not frozen	72.96	81.96	57.64	68.78	67.51	75.97

Table 1: Effect of layer freezing when using a pre-trained backbone with DER++. Results are computed at the end of the last task on Seq-CIFAR10, for different pre-training modalities, in the Class-IL setting.

Pretraining	**ImageNet**		**CIFAR-100**		**ImageNet-10**	
Layer	**200**	**500**	**200**	**500**	**200**	**500**
l_1	79.87	84.94	66.78	73.02	59.69	59.18
l_2	79.62	85.80	65.38	73.41	60.32	61.65
l_3	79.62	86.03	65.22	72.90	55.93	57.98
l_4	80.31	84.47	66.14	71.00	57.79	56.97
l_5	80.06	85.76	65.08	69.72	53.45	54.02
l_6	79.80	85.56	64.60	70.24	50.76	52.25
l_7	79.01	83.96	60.62	67.35	50.71	49.61
l_8	11.74	10.47	10.35	11.17	38.57	39.96
not frozen	80.70	85.11	64.94	73.70	59.33	65.66

Table 2: Effect of layer freezing when using a pre-trained backbone with ER-ACE. Results are computed at the end of the last task on Seq-CIFAR10, for different pre-training modalities, in the Class-IL setting.

4.2. Baselines

We carry out a set of experiments, freezing different portions of the backbone network according to different criteria, while employing the well-known continual learning DER++ [4] and ER-ACE [5] approaches. DER++ is a rehearsal approach that applies a form of knowledge distillation on buffered samples by encouraging the model to predict the corresponding logits, rather than the class directly (as done in the classic ER approach [7]). ER-ACE, instead, applies an *asymmetric* variant of the cross-entropy loss function, by limiting the computation of softmax scores on classes present in the current mini-batch (for elements from the current task), extended with previous-task classes for buffered samples only.

As a backbone for both approaches, we employ a ResNet-18 network [14], illustrated in Fig. 1. In standard implementations of the model[1], the feature extraction portion of the network consists of four *main blocks* (depicted with distinct colors in the figure), each of which includes two *basic blocks*; in turn, each basic block applies two convolutions with a residual connection. In the following experiments, we will treat each block as the smallest unit of freezing, named *layer*. Consequently, the network is divided into eight *layers*, aligning with the original structure of the basic blocks. The first convolution of the network is assumed to be part of the first layer.

Layer	**Task 1**	**Task 2**	**Task 3**	**Task 4**
l_1	62.32	64.14	65.35	64.24
l_2	58.75	63.56	63.09	64.80
l_3	60.63	64.24	64.53	65.58
l_4	57.58	62.75	66.46	63.34
l_5	54.66	62.28	65.18	65.05
l_6	54.65	62.93	64.43	63.75
l_7	46.70	55.11	54.26	54.03
l_8	31.95	49.65	52.36	38.62
not frozen			63.77	

Table 3: Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 200. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting.

Layer	**Task 1**	**Task 2**	**Task 3**	**Task 4**
l_1	70.00	72.20	73.94	73.28
l_2	67.47	69.59	73.86	73.52
l_3	67.70	70.83	73.00	74.31
l_4	64.36	69.81	71.85	74.01
l_5	62.26	69.28	73.63	72.98
l_6	60.87	69.62	71.82	73.40
l_7	52.36	60.32	68.57	59.57
l_8	44.98	54.90	40.52	49.67
not frozen			72.38	

Table 4: Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 500. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting.

4.3. Training details

In our experiment we follow [4] and adopt the same training settings. All models are trained by means of Stochastic Gradient Descent (SGD) with a fixed learning rate of 0.03. Training is performed separately for 50 epochs for each task. Training batches are composed by mixing data retrieved from the current task and previous samples from the replay buffer. Buffer sizes 200 and 500 were used in our experiments. The training dataset is normalized using the mean and the standard deviation values computed on data used for the training process. The data augmentation strategy includes random crop and horizontal flip.

[1]We refer to the PyTorch implementation.

Layer	**Task 1**	**Task 2**	**Task 3**	**Task 4**
l_1	58.67	64.02	54.81	63.88
l_2	62.09	62.31	66.38	64.12
l_3	59.78	63.53	63.47	66.12
l_4	57.75	64.36	66.00	65.65
l_5	57.84	62.24	65.93	63.41
l_6	58.73	63.07	64.56	66.11
l_7	55.10	62.67	51.89	55.07
l_8	46.04	53.45	47.03	49.07
not frozen			65.93	

Table 5: Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 200. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting.

Layer	**Task 1**	**Task 2**	**Task 3**	**Task 4**
l_1	68.27	71.22	70.96	71.91
l_2	68.27	70.19	71.17	70.99
l_3	67.96	70.82	71.09	71.15
l_4	65.04	70.44	71.39	70.99
l_5	64.83	69.76	69.27	72.11
l_6	62.08	69.21	70.67	72.54
l_7	58.49	67.47	60.14	61.09
l_8	50.89	55.98	50.98	58.08
not frozen			72.17	

Table 6: Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 500. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting.

For evaluating the freezing strategies, the optimization of Eq. 8 is carried out on a distinct portion of the training set, which we refer to as *validation set*. Specifically, for a given task τ_i, we optimize $\mathcal{L}_{\text{sf}}$ on the corresponding validation $\mathcal{D}_{i,\text{val}} \cup \mathbf{M}_{i-1,\text{val}}$, where $\mathcal{D}_{i,\text{val}}$ contains 10% of the training set $\mathcal{D}_i$, and $\mathbf{M}_{i-1,\text{val}}$ includes 10% of the buffer $\mathbf{M}_{i-1}$. Additionally, as alternative approaches, we also try using only the data from the current task ($\mathcal{D}_{i,\text{val}}$), or exclusively with the data contained in the buffer ($\mathbf{M}_{i-1,\text{val}}$).

All experiments were run on a single NVIDIA RTX 3090 GPU and implemented using the PyTorch framework.

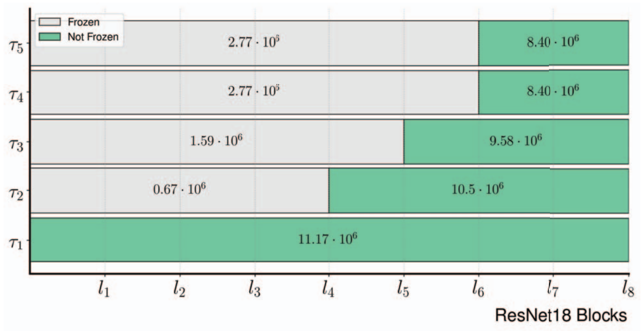

Figure 2: The prevalent freezing scheme when selective freezing is activated for DER++ trained on Seq-CIFAR10 over 10 runs. Number of parameters indicated within the bars.

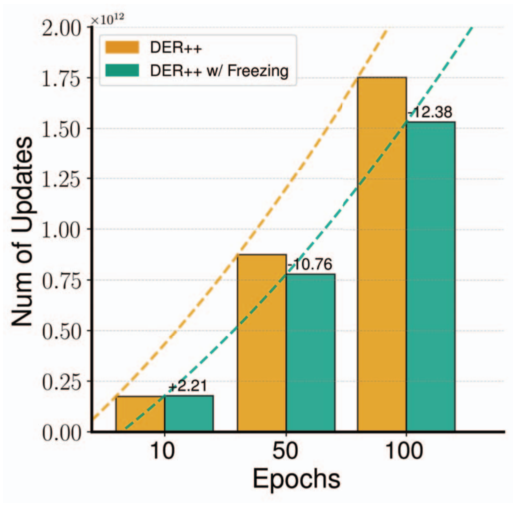

Figure 3: Parameter update count for the complete training of vanilla DER++ (in orange) compared to our selective freezing strategy (in green). The numbers above the green bars depict the percentage improvement points compared to the baseline.

4.4. Static freezing

We first present a set of experiments aimed at assessing the overall impact of different *static* freezing strategies (i.e., where the choice of which layers to freeze and the task at which they are frozen is performed *a priori*, independently of the model's performance). This preliminary analysis provides useful hints on how continual learning performance is affected by feature freezing in general; moreover, it allows us to establish a reference for comparison in our experiments with selective freezing.

We first analyze the effect of layer freezing with a pre-trained ResNet-18 backbone. A common setting of continual learning methods assumes that model features are trained from scratch, in order to assess a method's capability to learn strong features that support forward transfer and are not spurious or task-specific. In our scenario, it is interesting to evaluate the extent to which layer freezing interacts with the availability of pre-trained features: indeed, this setting should reduce the need for forward transfer (as we can assume that pre-trained features mitigate spurious feature learning) and robustness to forgetting (due to freezing). The results of these experiments are shown in Tab. 1 (for DER++) and Tab. 2 (for ER-ACE). We report the class-incremental learning (Class-IL) accuracy computed after the last task in Seq-CIFAR10, for buffer sizes 200 and 500, when freezing up to a certain layer (in the table, the row corresponding to l_i refers to when the i-th layer w.r.t. Fig. 1 is frozen, along with all previous layers); we take into account classification pre-training on three different datasets, CIFAR100, ImageNet-10[2] and ImageNet-1k [9]. It is interesting to note that the Class-IL accuracy does not decrease significantly when freezing up to layer l_6, in several cases even improving accuracy with respect to the non-frozen case. As mentioned above, this can be expected, since freezing pre-trained features helps reduce forgetting while reusing already good features, at the cost (in this case, negligible) of giving up learning task-specific low-level features. However, going deeper with freezing, performance drops when using pretrained features from CIFAR100 and ImageNet-1k, showing that a certain degree of feature customization may be in order. Interestingly, this drop is mitigated when pre-training on ImageNet-10 (see Tables 1 and 2). A possible explanation lies in the fact that ImageNet-10 only has ten classes: as such, it is tailored towards learning features that are less class-specific and that can be effectively leveraged by the final fully-connected layer, even when the entire backbone of the model is frozen.

The previous results show that layer freezing does not seem to incur a significant performance loss in a continual learning, thanks to pre-training. However, it is more interesting to see how this changes when the entire backbone is trained from scratch. To this aim, Tab. 3, 4, 5, 6 show the results for DER++ and ER-ACE without a pre-trained backbone. Accuracy is reported in terms of Class-IL for different buffer sizes, on Seq-CIFAR10. In detail, each table reports these metrics when the backbone is frozen up to a certain layer at the end of a certain task: during previous tasks, the entire backbone is updated. This setting thus takes into account not only the depth but also the *moment* at which backbone features are frozen, allowing us

[2]ImageNet-10 is available at: https://github.com/fastai/imagenette

to study the trade-off between freezing early (encouraging feature preservation but reducing plasticity) or late (encouraging adaptability to new tasks but risking catastrophic forgetting). As expected, in the absence of a pre-training stage, freezing tends to reduce model performance compared to full training of the entire backbone. A general trend can be identified for both DER++ and ER-ACE and for both tested buffer sizes: later freezing (with respect to the sequence of tasks) yields higher accuracy, as the model can work at its full capacity for a longer time; deeper freezing (with respect to the sequence of layers in the backbone), tends to reduce accuracy, for a similar reason, since the model is forced to reuse features that — unlike in the pre-training experiments — may not be representative for future tasks. Nevertheless, it is possible to find a "sweet spot" among the different freezing configurations, representing good compromises between accuracy and efficiency. For instance, while freezing after the first task may be too early to learn good features, freezing after the second task, even up to l_3, leads to a relatively small decrease in accuracy, on both the Class-IL and Task-IL metrics. This behavior appears to be consistent for DER++ and ER-ACE, for buffer sizes 200 and 500.

Validation set	**DER++**		**ER-ACE**	
	200	**500**	**200**	**500**
$\mathcal{D}_{i,\text{val}} \cup \mathbf{M}_{i-1,\text{val}}$	60.20	66.92	59.52	64.65
$\mathcal{D}_{i,\text{val}}$	62.20	69.89	58.52	66.26
$\mathbf{M}_{i-1,\text{val}}$	58.84	62.35	57.50	64.62
vanilla training	64.83	72.13	63.81	71.86

Table 7: Effect of selective freezing obtained using DER++ and ER-ACE methods. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting, for both 200 and 500 buffer sizes.

Validation set	**DER++**		**ER-ACE**	
	200	**500**	**200**	**500**
$\mathcal{D}_{i,\text{val}} \cup \mathbf{M}_{i-1,\text{val}}$	57.57	67.90	66.21	71.86
$\mathcal{D}_{i,\text{val}}$	58.11	67.56	64.84	71.02
$\mathbf{M}_{i-1,\text{val}}$	56.46	64.77	64.23	69.53
vanilla training	58.21	69.00	66.33	73.77

Table 8: Effect of selective freezing with DER++ and ER-ACE methods using a backbone pre-trained on the CIFAR-100 dataset. Results are computed at the end of the last task on Seq-CIFAR10 in the Class-IL setting, for both 200 and 500 buffer sizes.

4.5. Selective freezing

In this section, we present the results obtained while applying the proposed selective freezing strategy to DER++ and ER-ACE on the Seq-CIFAR10 dataset, for different buffer sizes. In detail, Tab. 7 reports Class-IL accuracy at the end of the final task with the three variants of selective freezing described in Sect. 4.3, i.e., when the validation set of freezing decision includes either training samples only (from the new task), or buffer samples only, or both.

It can be observed that generally the results obtained with selective freezing are slightly lower compared to the case of vanilla training, with a drop in accuracy in the range of 3 to 6 percentage points. The overall performance aligns with the optimal strategy identified through static freezing, which involves freezing after task 2 up to l_5. Moreover, it seems that training on new task data only (i.e., without including buffer samples) leads to improved performance, indicating that adaptation to new data during selective freezing plays a more important role than recalling past knowledge, which is instead handled during standard training.

Tab. 8 reports results from the same experiments, carried out starting from a pre-trained backbone. It is interesting to notice that, in this scenario, ER-ACE seems to benefit from pre-training significantly more than DER++. This may indicate that ER-ACE exhibits better properties at feature reuse and forward transfer; interestingly, this interpretation is also supported by the results shown in the following, according to which ER-ACE has a stronger tendency to selectively freeze larger parts of the backbone at earlier tasks.

We finally assess the computational efficiency of our selective freezing strategy on DER++ and ER-ACE compared to standard training. Fig. 2 illustrates the prevailing freezing scheme observed in DER++ during training on CIFAR10, over 10 different experimental runs, while Figure 3 shows the resulting efficiency in terms of the number of parameter updates required during the entire training process. Notably, the efficiency gain shows an increasing trend with the growth of the number of epochs per task. When the number of epochs is relatively low (e.g., 10), selective freezing leads to a higher number of updates compared to standard training. This is due to the fact that during the first epoch of the second task, up to 7 different model configurations are trained and evaluated in parallel. However, as the number of epochs increases, the overhead introduced during this phase becomes progressively more marginal, leading to a considerable efficiency gain of 12.34% with 100 epochs per task.

It is worth noting that the specific behaviors of freezing can also depend on the particular methods of continual learning employed in the experiments. As an example, ER-ACE tends to freeze more layers in the initial phase, showing a preference for a more conservative configuration right at the beginning of the second task, as depicted in Figure 4, while DER++ adopts a more gradual freezing strategy. Indeed, freezing more layers in an early stage has a positive impact on efficiency: for ER-ACE the efficiency gain ranges from 8.38% in the case of 10 epochs per task, to a remarkable 18.72% extending the number of epochs to 100, as shown in Figure 5.

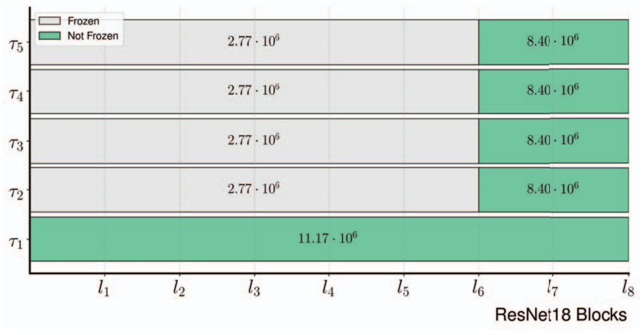

Figure 4: The prevalent freezing scheme when selective freezing is activated for ER-ACE trained on Seq-CIFAR10 over 10 runs. Number of parameters indicated within the bars.

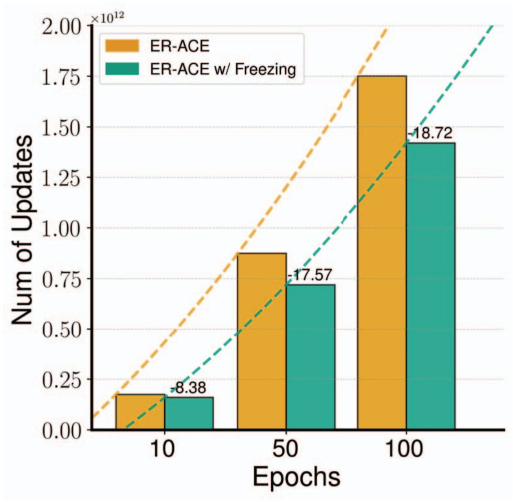

Figure 5: The number of parameter updates required to train the vanilla ER-ACE model (shown in orange) compared to the same count with our selective freezing strategy (in green). The numbers above the green bars represent the percentage improvement points compared to the baseline.

5. Conclusions

In this paper, we have presented a novel approach that performs selective layer freezing to address the challenges of continual learning and computational efficiency in deep learning models. Our findings highlight the potential for freezing a significant portion of the model without sacrificing accuracy: by dynamically identifying the optimal subset of frozen layers during training, we achieve a balance between plasticity on new tasks and stability on previous tasks. Experimental results demonstrate the competitiveness of our approach compared to manually-tuned freezing strategies. We have also quantitatively estimated the reduction in computation and energy requirements achieved through our freezing strategy, showcasing its potential for mitigating the environmental impact associated with large-scale deep learning models.

While the proposed approach is able to achieve promising results, its simplicity introduces certain limitations, that need to be addressed. First, our experiments focus on a specific network architecture, ResNet-18: while it is the *de facto* backbone for many continual learning approaches, it is important to assess the impact of selective freezing on different and deeper networks. Second, our analysis and experiments are conducted based on a specific data set and task setup: The effectiveness of our approach may vary with different task characteristics, data set sizes, and variation in task sequence.

In the future, we also aim to explore more sophisticated freezing strategies, for instance by introducing a finer selection of parameters, rather than working at the layer level. Secondly, investigating the interplay between freezing strategies and other techniques, such as regularization methods, knowledge distillation or architectural modifications, could provide additional insights into improving continual learning and computational efficiency.

Acknowledgements

This research was supported by the PNRR MUR project PE0000013-FAIR. Matteo Pennisi and Amelia Sorrenti are PhD students enrolled in the National PhD in Artificial Intelligence, course on Health and life sciences, organized by Università Campus Bio-Medico di Roma.

References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, 2019. 2

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient Based Sample Selection for Online Continual Learning. In *Advances in Neural Information Processing Systems*, 2019. 2

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General

Continual Learning: a Strong, Simple Baseline. In *Advances in Neural Information Processing Systems*, 2020. 2, 4, 5

[5] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New Insights on Reducing Abrupt Representation Change in Online Continual Learning. In *International Conference on Learning Representations Workshop*, 2022. 2, 4

[6] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *IEEE International Conference on Computer Vision*, 2021. 2

[7] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. In *International Conference on Machine Learning Workshop*, 2019. 2, 4

[8] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2009. 4, 6

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[12] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 2

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[16] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019. 2

[17] Jeremy Howard. Imagewang. 4

[18] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in neural information processing systems*, 33:3647–3658, 2020. 2

[19] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pages 10734–10750. PMLR, 2022. 2

[20] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. In *IEEE International Conference on Computer Vision*, 2021. 2

[21] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

[22] Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends Cogn Sci*, 20(7):512–534, Jul 2016. 3

[23] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ale*vs*. Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[24] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

[25] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018. 2

[26] Marc Masana, Tinne Tuytelaars, and Joost Van de Weijer. Ternary feature masks: zero-forgetting for task-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3570–3579, 2021. 2

[27] James L McClelland, Bruce L McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychol Rev*, 102(3):419–457, Jul 1995. 3

[28] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 1989. 1, 2, 3

[29] Divya Pandey, Madhoolika Agrawal, and Jai Shanker Pandey. Carbon footprint: current methods of estimation. *Environmental monitoring and assessment*, 178:135–160, 2011. 1

[30] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019. 2

[31] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 2021. 2

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[33] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 1990. 2

[34] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017. 2

[35] Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. 2

[36] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018. 2

[37] Yujun Shi, Li Yuan, Yunpeng Chen, and Jiashi Feng. Continual learning via bit-level information preserving. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 16674–16683, 2021. 2

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, 2017. 2

[39] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016. 2

[40] Gido M van de Ven and Andreas S Tolias. Three continual learning scenarios. In *Neural Information Processing Systems Workshops*, 2018. 2

[41] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019. 2

[42] Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263, 2021. 2

[43] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019. 2

[44] Li Yang, Sen Lin, Fan Zhang, Junshan Zhang, and Deliang Fan. Efficient self-supervised continual learning with progressive task-correlated layer freezing. *arXiv preprint arXiv:2303.07477*, 2023. 2