PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning

Gregoire Petit, Adrian Popescu, Eden Belouadah, David Picard, Bertrand Delezoide

PLASTIL: PLASTIC AND STABLE EXEMPLAR-FREE CLASS-INCREMENTAL LEARNING

Grégoire Petit[1,2], Adrian Popescu[1], Eden Belouadah[3], David Picard[2], Bertrand Delezoide[3]
[1]Université Paris-Saclay, CEA, LIST, F-91120, Palaiseau, France
[2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
[3]Datakalab, 114 Bd Malesherbes, 75017 Paris, France
[4]Amanda, 34 Avenue Des Champs Elysées, F-75008, Paris, France
{gregoire.petit, adrian.popescu}@cea.fr,eb@datakalab.com
david.picard@enpc.fr,bertrand.delezoide@amanda.com

ABSTRACT

Plasticity and stability are needed in class-incremental learning in order to learn from new data while preserving past knowledge. Due to catastrophic forgetting, finding a compromise between these two properties is particularly challenging when no memory buffer is available. Mainstream methods need to store two deep models since they integrate new classes using fine-tuning with knowledge distillation from the previous incremental state. We propose a method which has similar number of parameters but distributes them differently in order to find a better balance between plasticity and stability. Following an approach already deployed by transfer-based incremental methods, we freeze the feature extractor after the initial state. Classes in the oldest incremental states are trained with this frozen extractor to ensure stability. Recent classes are predicted using partially fine-tuned models in order to introduce plasticity. Our proposed plasticity layer can be incorporated to any transfer-based method designed for exemplar-free incremental learning, and we apply it to two such methods. Evaluation is done with three large-scale datasets. Results show that performance gains are obtained in all tested configurations compared to existing methods.

1 INTRODUCTION

Class-incremental learning (CIL) enables the adaptation of artificial agents to dynamic environments in which data occur sequentially. CIL is particularly useful when the training process is performed under memory and/or computational constraints (Masana et al., 2021). However, it is really susceptible to catastrophic forgetting, which refers to the tendency to forget past information when learning new data (Kemker et al., 2018; Mccloskey & Cohen, 1989). Most recent CIL methods (Douillard et al., 2020; Hou et al., 2019; Javed & Shafait, 2018; Rebuffi et al., 2017; Wu et al., 2019) use fine-tuning with knowledge distillation (Hinton et al., 2015) from the previous model to preserve past information. Knowledge distillation has been progressively refined (Hou et al., 2019; Javed & Shafait, 2018; Wu et al., 2021; Yu et al., 2020; Zhou et al., 2019) to improve CIL performance. An alternative approach to CIL is inspired by transfer learning (Razavian et al., 2014). These methods use a feature extractor which is frozen after the initial CIL state (Belouadah & Popescu, 2018; Hayes et al., 2020; Hayes & Kanan, 2020; Rebuffi et al., 2017). They become competitive in exemplar-free CIL, a difficult setting due to a strong effect of catastrophic forgetting (Masana et al., 2021). The main challenge is to find a good plasticity-stability balance because fine-tuning methods favor plasticity, while transfer-based methods only address stability.

In this work, we tackle exemplar-free CIL (EFCIL) by combining the two types of approaches described above. Building on the strong performance of transfer-based methods (Belouadah et al., 2021; Hayes & Kanan, 2020), we introduce a plasticity component by partially fine-tuning models for recent classes. The results from Figure 1 show that our method gives a better global accuracy compared to $DeeSIL$ (Belouadah & Popescu, 2018) and $LUCIR$ (Hou et al., 2019), two representative methods focused on stability and plasticity, respectively. Accuracy is presented separately for past and new classes for existing methods to examine the plasticity-stability balance offered by each method. $LUCIR$ has optimal plasticity (best accuracy of new classes), while $DeeSIL$ has optimal stability (best accuracy for past classes). However, the performance of both methods is strongly degraded on the complementary dimensions. Our method is close to $LUCIR$ in terms of plasticity and to $DeeSIL$ in terms of stability. Consequently, it ensures a better balance between these two properties of EFCIL.

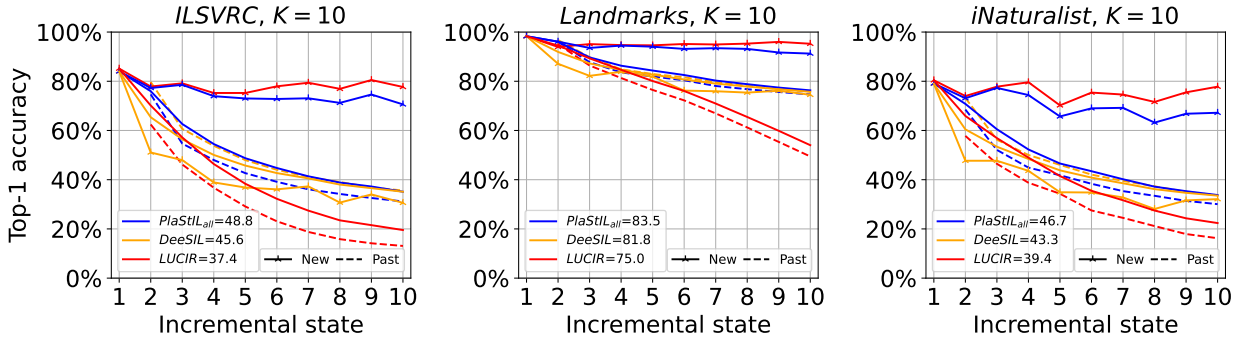

Figure 1: Accuracy of past and new classes in exemplar-free CIL for three large-scale datasets with $K = 10$ incremental states. $LUCIR$ (Hou et al., 2019) uses distillation to preserve past knowledge and favors plasticity. $DeeSIL$ (Belouadah & Popescu, 2018) transfers features from the initial frozen model to all subsequent states and focuses on stability. *PlaStIL* offers a better plasticity-stability balance. Note that the proportion of past classes increases as the incremental process advances and so does their weight in global accuracy.

PlaStIL is inspired by transfer learning but adds a partial fine-tuning component to boost plasticity. It is applicable to any transfer-based method and we exemplify it with $DSLDA$ (Hayes & Kanan, 2020) and $DeeSIL$ (Belouadah & Popescu, 2018). We introduce a hybrid classification layer which combines classification weights learned with the initial model for past classes and with the fine-tuned models for recent classes. We evaluate the proposed approach on three datasets which contain 1000 classes each. The number of incremental states is varied to assess the robustness of the tested methods. Results show that performance gains are obtained by adding the proposed plasticity component to transfer-based methods. Equally interesting, important performance improvements are obtained over distillation-based methods, which are the mainstream methods deployed to tackle CIL (Hou et al., 2019; Javed & Shafait, 2018; Rebuffi et al., 2017; Smith et al., 2021; Wu et al., 2021). We will open-source the code to facilitate reproducibility.

2 Related Work

Incremental learning is interesting when artificial agents need to learn under memory or computational constraints (Masana et al., 2021; Parisi et al., 2019; Rebuffi et al., 2017). The main challenge in CIL is to tackle the catastrophic forgetting phenomenon (Kemker et al., 2018; Mccloskey & Cohen, 1989). A suitable balance between plasticity and stability of the learned models is sought (Mermillod et al., 2013). Plasticity and stability are needed in order to accommodate new data and preserve previously learned knowledge, respectively (Chaudhry et al., 2018). As noted in a recent survey (Masana et al., 2021), a large majority of CIL-related works use a memory buffer which stores samples of past classes in order to improve overall performance. Replaying these samples facilitates the preservation of past knowledge, thus making the incremental learning process akin to imbalanced learning (Belouadah et al., 2021). However, the assumption that past samples are available is strong and limits the applicability of CIL. A growing research effort was devoted to exemplar-free CIL (EFCIL) (Belouadah et al., 2021; Masana et al., 2021). The plasticity-stability dilemma is particularly challenging without memory since the effects of catastrophic forgetting are stronger in this case (Belouadah et al., 2020; Rebuffi et al., 2017; Smith et al., 2021; Wu et al., 2021).

Survey papers such as (Lange et al., 2019; Masana et al., 2021) analyze different types of continual learning methods which are usable in exemplar-free CIL. Parameter-isolation methods, such as HAT (Serra et al., 2018) or $PackNet$ Mallya & Lazebnik (2018) were designed for task-incremental learning, a setting in which the task ID is known at inference time. They learn task-specific masks to reduce catastrophic forgetting. However, they are impractical in task-agnostic scenarios since the simultaneous evaluation of all tasks is not possible and specific forward passes are needed for each task (Masana et al., 2021). Regularization-based methods are a popular solution to EFCIL and they fall into two subcategories, namely data-focused or prior-focused approaches. Existing works (Lange et al., 2019; Masana et al., 2021; Petit et al., 2023) showed that data-focused methods outperform prior-focused on in EFCIL scenarios. Consequently, we discuss data-focused methods and use representative examples of them in experiments. According to Belouadah et al. (2021); Masana et al. (2021), most methods update learned models in each IL state using fine-tuning for plasticity and different flavors of knowledge distillation (Hinton et al., 2015) for stability. Alternatively, a few works (Belouadah & Popescu, 2018; Dhamija et al., 2021; Hayes et al., 2020; Hayes & Kanan, 2020) use an initial representation throughout the incremental process. We discuss the merits and limitations of both approaches below and position our contribution with respect to them.

Distillation-based methods are inspired by LwF (Li & Hoiem, 2016), an adaptation of knowledge distillation (Hinton et al., 2015) to an incremental context. The authors of Dhar et al. (2018) add an attention mechanism to the distillation loss to preserve information from past classes and obtain an improvement over LwF. Since its initial use for exemplar-based CIL in $iCaRL$ (Rebuffi et al., 2017), distillation was refined and complemented with other components to improve the plasticity-stability compromise. $LUCIR$ (Hou et al., 2019) applies distillation on feature vectors instead of raw classification scores to preserve the geometry of past classes, and an inter-class separation to maximize the distances between past and new classes. LwM (Dhar et al., 2018) adds an attention mechanism to the distillation loss to preserve information from base classes. An interesting solution is proposed in Yu et al. (2020), where the feature drift between incremental steps is estimated based on features of samples associated to new classes. However, this method has a large footprint since it needs a large multi-layer perceptron and also stores past features to learn the transformation. A feature transformation method is designed for task-incremental learning and adapted for CIL by predicting the task associated to each test sample (Verma et al., 2021). The authors of Smith et al. (2021) combined feature drift minimization and class separability to improve distillation. The authors of Wu et al. (2021) proposed an approach which stabilizes the fine-tuned model, adds reciprocal adaptive weights to weigh past and new classes in the loss, and introduces multi-perspective training set augmentation. They reported important gains in exemplar-free CIL compared to $LUCIR$ (Hou et al., 2019) and SDC (Yu et al., 2020) using a protocol in which half of the dataset is available initially (Hou et al., 2019). Distillation is widely used but a series of studies question its usefulness (Belouadah et al., 2021; Masana et al., 2021; Prabhu et al., 2020), especially for large-scale datasets. One explanation for the lack of scalability of distillation is that inter-class confusion becomes too strong when the number of past classes is high. Another challenge is that distillation needs to store the previous deep model to preserve past knowledge. The total footprint of these methods is the double of the footprint of the backbone model used.

A second group of methods learns a deep representation in the initial state and uses it as feature extractor throughout the CIL process. They are inspired by transfer learning (Tan et al., 2018) and favor stability since the initial representation is frozen. Usually, these methods learn shallow external classifiers on top of the initial deep representation. The nearest class mean (NCM) (Mensink et al., 2013) was introduced in Rebuffi et al. (2017), linear SVCs (Cortes & Vapnik, 1995) were used in Belouadah & Popescu (2018); Petit et al. (2023) and extreme value machines (Rudd et al., 2017) were tested by Dhamija et al. (2021). $REMIND$ (Hayes et al., 2020) uses a vector quantization technique to save compressed image representations which are later reconstructed for memory consolidation by training model tops. The main difference with our proposal is that the past is represented via compressed image representations instead of model tops. $DSLDA$ (Hayes & Kanan, 2020) updates continuously a class-specific mean vector and a shared covariance matrix. The predicted label is the one having the closest Gaussian in the feature space defined by these vectors and matrix. These methods are simple and suited for exemplar-free CIL, particularly for large-scale datasets where they outperform distillation-based methods (Belouadah et al., 2021; Masana et al., 2021). Equally important, they only use the initial model and thus have a smaller footprint. Their main drawbacks are the genericity of the initial representation and the sensitivity to strong domain variations. Their performance drops if a small number of classes is initially available (Belouadah et al., 2021) and if the incremental classes have a large domain shift with classes learned initially (Lange et al., 2019). The robustness of the initial representation can be improved is an assumption is made that a model pretrained with a large amount of data is available and that its features are transferable to the incremental datasets (Hayes et al., 2020). ESN (Wang et al., 2023) is an interesting method which was proposed very recently, and makes this assumption, similarly to $REMIND$ and $DSLDA$. ESN leverages a pretrained transformer model, trains classifiers per state and then merges them by combining a temperature-controlled energy metric, an anchor-based energy self-normalization strategy and a voting-based inference augmentation strategy to ensure impartial and robust EFCIL predictions. Here, we experiment without a large pretrained model in order to cover cases where there is a large domain drift between the pretraining and the incremental datasets.

3 Proposed Method

3.1 Problem Formalization

The CIL process is divided in K states, with n classes learned in each state. In EFCIL, no past data can be stored for future use. The predictions associated to observed classes are noted p. We write the structure of a deep model as:

$$\mathcal{M} = \{\mathcal{B}, \mathcal{T}, \mathcal{W}\} \tag{1}$$

with: $\mathcal{M}$ - the full model; $\mathcal{B}$ - the model base which includes the initial layers; $\mathcal{T}$ - the model top which includes the subsequent layers up to the classification one; $\mathcal{W}$ - the classification layer which provides class predictions.

Assuming that the CIL process includes K states, the objective is to learn K models in order to incorporate all classes which arrive sequentially. The incremental learning process can be written as:

$$\mathcal{M}_1 \rightarrow \mathcal{M}_2 \rightarrow ... \rightarrow \mathcal{M}_k \rightarrow ... \rightarrow \mathcal{M}_{K-1} \rightarrow \mathcal{M}_K \tag{2}$$

Each incremental model needs to integrate newly arrived data, while also preserving past knowledge. Assuming that the current state is $k \geq 2$, the majority of existing CIL methods (Castro et al., 2018; Hou et al., 2019; Rebuffi et al., 2017; Smith et al., 2021; Wu et al., 2021; 2019) fine-tunes the entire current model $\mathcal{M}_k$ by distilling knowledge from $\mathcal{M}_{k-1}$. We note w_k^1 to $w_k^{(k-1)\times n}$ the classifier weights of the past states 1 to $k-1$, and $w_k^{(k-1)\times n+1}$ to $w_k^{k\times n}$ the classifier weights of the new state k.

Their classification layer is written as:

$$\mathcal{W}_k^{ft} = \{w_k^1, ..., w_k^n, ..., w_k^{(k-1)\times n}, w_k^{(k-1)\times n+1}, ..., w_k^{k\times n}\} \tag{3}$$

They are all trained using $\mathcal{T}_k$, the model top learned in the k^{th} state. The $\mathcal{W}_k^{ft}$ layer is biased toward new classes since it is learned with all samples from the current state, but only with the representation of past classes stored in $\mathcal{M}_{k-1}$ (Hou et al., 2019; Rebuffi et al., 2017; Wu et al., 2019). This group of methods focuses on CIL plasticity at the expense of stability (Belouadah et al., 2021; Masana et al., 2021).

Transfer-based methods (Belouadah & Popescu, 2018; Hayes et al., 2020; Hayes & Kanan, 2020; Petit et al., 2023) freeze the feature extractor $\mathcal{F}_1 = \{\mathcal{B}_1, \mathcal{T}_1\}$ after the initial non-incremental state. All the classes observed during the CIL process are learned with $\mathcal{F}_1$ as features extractor. The classification layer can be written as:

$$\mathcal{W}_1^{fix} = \{w_1^1, ..., w_1^n, ..., w_1^{(k-1)\times n}, w_1^{(k-1)\times n+1}, ..., w_1^{k\times n}\} \tag{4}$$

All classifier weights from Eq. 4 are learned with image features provided by $\mathcal{F}_1$, the feature extractor learned initially, inducing a bias toward initial classes. It is suboptimal for classes learned in states $k \geq 2$ because their samples were not used to train $\mathcal{F}_1$. These methods focus on CIL stability at the expense of plasticity (Masana et al., 2021).

3.2 *PlaStIL* Description

PlaStIL is motivated by recent studies which question the role of distillation in CIL, particularly for large-scale datasets (Belouadah et al., 2021; Masana et al., 2021; Prabhu et al., 2020). Instead of $\mathcal{M}_{k-1}$ needed for distillation, *PlaStIL* uses two or more model tops $\mathcal{T}$ which have an equivalent number of parameters at most. *PlaStIL* is inspired by feature transferability works (Neyshabur et al., 2020; Yosinski et al., 2014) which show that higher layers of a model, included in $\mathcal{T}$, are the most important for successful transfer learning. Consequently, the initial layers $\mathcal{B}$ are frozen and shared throughout the incremental process. A combination of model tops which includes $\mathcal{T}_1$, the one learned in the first incremental state, and those of the most recent state(s) is used in *PlaStIL*. Similar to transfer-based CIL methods (Belouadah & Popescu, 2018; Hayes et al., 2020; Hayes & Kanan, 2020; Petit et al., 2023), $\mathcal{T}_1$ ensures stability for classes first encountered in past states for which a dedicated top model is not available. Different from existing methods, model top(s) are available for the most recent state(s), thus improving the overall plasticity of *PlaStIL*. The number of different model tops which can be stored instead of $\mathcal{M}_{k-1}$ depends on the number of higher layers which are fine-tuned in each incremental state. The larger the number of layers in $\mathcal{T}$, the larger its parametric footprint is and the lower the number of storable model tops will be.

The method is illustrated in Figure 2 with a toy example which includes $K = 4$ IL states, with $n = 2$ new classes per state and which assumes that up to three model tops can be stored. Up to the third state, *PlaStIL* stores a model top per state and corresponding classifier weights are learned for each model top. In the fourth state, one of the model tops needs to be removed in order to keep the parameters footprint bounded. Consequently, $\mathcal{T}_2$ is removed and the initial model top $\mathcal{T}_1$ is used. Note that $\mathcal{T}_1$ is used to learn classifier weights for all classes when they occur initially. These initial weights are stored for usage in later incremental states in order to cover all past classes for which dedicated model tops cannot be stored. The storage of the initial weights generates a small parameters overhead but its size is small and does not increase over time. If the classifier weights are d-dimensional, the number of supplementary parameters is $n \times d$. Moreover, this overhead can be easily compensated by the choice of number of parameters in $\mathcal{T}$ and the number of such model tops which are stored. In Figure 2, initial classifier weights w_1^3 and w_1^4 are first learned in state 2 but only used in state 4, when $\mathcal{T}_2$ is no longer available. In state 4, $\mathcal{T}_1$ is used for classes which first occurred in states 1 and 2 (classifiers weights w_1^1 to w_1^4). $\mathcal{T}_3$ is reused along with its classifier weights w_3^5 and w_3^6, learned for the classes which were learned in state 3. Finally, classifier weights of new classes w_4^7 and w_4^8 are learned with $\mathcal{T}_4$. This results into a hybrid classification weights layer which is defined as:

$$\mathcal{W}_k^{hyb} = \{w_1^1, ..., w_1^{j\times n}, ..., w_{j+1}^{j\times n+1}, ..., w_{j+1}^{(j+1)\times n}, ..., w_k^{(k-1)\times n+1}, ..., w_k^{k\times n}\} \tag{5}$$

where we assume that $k - j + 1$ models can be stored, with $2 \leq j \leq k$; the blocks of classes learned with features from different model tops are color coded.

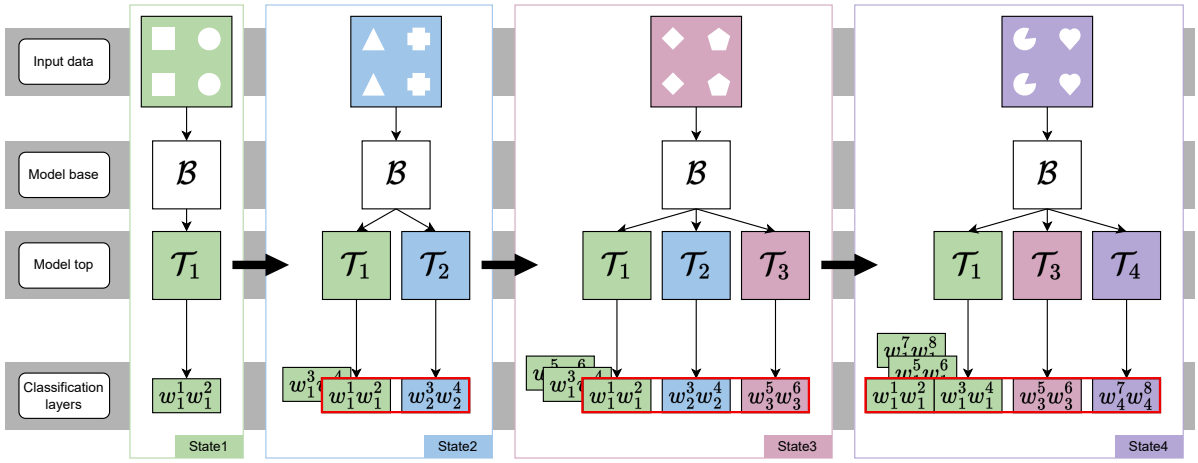

Figure 2: *PlaStIL* overview using a toy example with $K = 4$ CIL states and $n = 2$ new classes learned per state. The global memory footprint is equivalent to that of distillation-based methods, but this memory is used differently. We assume that a model base and at most three model tops can be used. A base $\mathcal{B}$, is learned in the initially and then frozen, as is $\mathcal{T}_1$ which is needed to ensure stability. Initial classifier weights are trained using $\mathcal{T}_1$ in each state and reserved for future use. Classifier weights which are actually used in each state are highlighted in red. In state 4, the recent model tops ($\mathcal{T}_4$ and $\mathcal{T}_3$) are included to ensure plasticity. Classifier weights w_4^7 and w_4^8, associated to the new classes from state 4 are learned with features provided by $\mathcal{T}_4$. w_3^5 and w_3^6 were learned with $\mathcal{T}_3$ features in state 3, when they were new. w_1^1 to w_1^4 were learned with $\mathcal{T}_1$ features, in states 1 and 3. w_1^5 to w_1^8 are reserved for future use. $\mathcal{T}_2$ is discarded to keep the total memory footprint of *PlaStIL* bounded. *Best viewed in color.*

In Equation 5, classifier weights of the first j incremental states are learned with the features provided by initial model top $\mathcal{T}_1$. Those of the most recent states ($j+1$ to k) are learned with features provided by model tops $\mathcal{T}_{j+1}$ to $\mathcal{T}_k$. An advantage of the layer from Equation 5 is that it ensures a good balance between stability, via $\mathcal{T}_1$ and plasticity, via $\mathcal{T}_{j+1}$ to $\mathcal{T}_k$. The number of storable model tops varies inversely with the number of layers that they include. We report results with three top depths in Section 4. A choice between internal and external classifiers has to be made for the implementation of this classification layer. Experiments from Belouadah et al. (2021) indicate that external classifiers are easier to optimize when transferring features from $\mathcal{M}_1$ to subsequent incremental states.

*PlaStIL*is primarily intended for a CIL scenario under the assumption that the parametric budget should not increase over time. If memory is allowed to grow over time, the method could store a larger number of model tops. The model top creation and removal policy could be adapted depending on the continual learning scenario being explored. For instance, if classes were grouped semantically, as it is the case in task-incremental learning, it might be better to create model tops for states which include classes which are most dissimilar from those of the initial model. Another interesting scenario assumes that past classes can be revisited, with new samples of them arriving later in the incremental process. In this case, model tops could be created for the current state if the amount of samples for revisited classes is smaller than those associated to existing tops. Such a top creation policy would be based on the assumption that the less revisiting there is, the more likely a new top would be due to the fact that the current state includes more novelty. In practice, the total number of stored model tops will depend on the total budget available on the device. When the memory budget is reached, one of the selection strategies listed above can be applied depending on the characteristics of the continual learning process. In Subsection 4.5, we show that the creation of model tops for the most recent states is a good solution for class incremental learning. While interesting, the adaptation of *PlaStIL*to other continual learning scenarios is out of the immediate scope and is thus left for future work.

4 Experiments

We evaluate *PlaStIL* with three large-scale datasets designed for different visual tasks. We compare it to a representative set of EFCIL methods. We vary the total number of states K using $K \in \{5, 10, 20\}$ because the length of the CIL process has strong effects on EFCIL performance (Masana et al., 2021). The evaluation metric is the top-1 accuracy averaged over all incremental states. Following a common practice in CIL (Castro et al., 2018; Hou et al., 2019; Wu et al., 2019), the performance on the initial state is excluded because it is not incremental.

4.1 DATASETS

We thus select large-scale datasets which provide a more realistic scenario for evaluation compared to medium-scale ones which are still used (Smith et al., 2021; Wu et al., 2021). We run experiments with:

- $ILSVRC$ (Russakovsky et al., 2015) - the well-known subset of ImageNet (Deng et al., 2009) built for the eponymous competition and also used in CIL (Castro et al., 2018; Hou et al., 2019; Rebuffi et al., 2017; Wu et al., 2019). The training and testing sets are composed of 1,231,167 and 50,000 images, respectively.
- $Landmarks$ - a subset of a landmarks recognition dataset (Noh et al., 2017) which includes a total of over 30000 classes. We select the 1000 classes having the largest number of images. The training and testing sets are composed of 374,367 and 20,000 images, respectively.
- $iNaturalist$ - a subset of the dataset used for the iNaturalist challenge (Van Horn et al., 2018). The full version includes 10000 fine-grained classes for natural species. We sample 1000 classes from different super-categories to obtain a diversified subset. The training and testing sets are composed of 300,000 and 10,000 images, respectively.

4.2 STATE-OF-THE-ART METHODS

We compare *PlaStIL* with the following existing methods:

- LwF (Rebuffi et al., 2017) - is a CIL version of the initial method from Li & Hoiem (2016). It tackles forgetting using a distillation loss.
- SIW (Belouadah et al., 2020) - uses a vanilla *FT* backbone and tackles catastrophic forgetting by reusing the past classifiers learned when these classes were first learned.
- $LUCIR$ (Hou et al., 2019) - adapts distillation to feature vectors instead of raw scores to preserve the geometry of past classes and also pushes for inter-class separation. Note that while initially proposed for CIL with memory, this method showed strong performance in EFCIL too (Belouadah et al., 2021).
- $SPB\text{-}M$ (Wu et al., 2021) - is a recent method which focuses on balancing plasticity and stability in exemplar-free CIL. We report results with the multi-perspective variant, which has the best overall performance in Wu et al. (2021). $SPB\text{-}M$ provides very competitive performance compared to $LUCIR$ when half of the dataset is initially available.
- $PASS$ (Zhu et al., 2021) - uses prototypes of past classes in combination with distillation in order to counter catastrophic forgetting.
- $REMIND$ Hayes et al. (2020) - encodes knowledge about past classes by storing compressed image representations. It is compared with our method by allocating the amount of storage used for model tops in our method to compressed representations of past samples.
- $FeTrIL$ (Petit et al., 2023) - is a very recent method which focuses on generating pseudo-features for past classes by using a geometric translation of features of similar new classes.
- $DSLDA$ (Hayes & Kanan, 2020) - is based on Gaussian functions defined in the features space by specific mean class vectors and a covariance matrix which is shared among all classes. This method is interesting since its classification layer provides an efficient inter-class separability mechanism.
- $DeeSIL$ (Belouadah & Popescu, 2018) - freezes the initial model uses linear SVCs (Cortes & Vapnik, 1995) for the final layer, which is trained independently for each state.

The first four methods fine-tune models incrementally. $REMIND$ trains model tops using a compressed replay buffer and is very relevant for comparison here. $DSLDA$ and $DeeSIL$ are transfer-based, and *PlaStIL* can be applied to them. They were implemented using their original optimal parameters. Whenever the original experimental settings were different from the ones used here, the correct functioning of the baselines was carefully checked. The obtained accuracy was coherent with the results reported in the original papers and/or in comparative studies such as Belouadah et al. (2021); Masana et al. (2021) in all cases. See details about the reproduced results in the appendix.

We experiment with three versions of *PlaStIL* designed to ensure that its parameters footprint is equivalent to (or lower than) that of distillation-based methods. We assume that the incremental process is in the k^{th} state and test:

- *PlaStIL*$_1$ - fine-tunes model tops $\mathcal{T}$ limited to the last convolutional layer of ResNet-18, which includes approximately 21.45% of the model parameters. Consequently, we can fit $\mathcal{T}_1$, $\mathcal{T}_{k-3}$, $\mathcal{T}_{k-2}$, $\mathcal{T}_{k-1}$ and $\mathcal{T}_k$ in memory.
- *PlaStIL*$_2$ - fine-tunes $\mathcal{T}$ which includes the last two convolutional layers of ResNet-18, which includes approximately 42.9% of the model parameters. We can fit $\mathcal{T}_1$, $\mathcal{T}_{k-1}$ and $\mathcal{T}_k$ in the allowed parameters memory.
- *PlaStIL*$_{all}$ - trains all the layers of the current model in k^{th} state and we can only use $\mathcal{T}_1$ and $\mathcal{T}_k$ in each IL state.

PlaStIL variant test different variants of the compromise between the number of model tops and their depth. *PlaStIL*$_1$ fine-tunes only the last convolutional layer of model tops and maximizes the number of such storable models. *PlaS-*

CIL Method	mem on disk	*ILSVRC*			*Landmarks*			*iNaturalist*		
		K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
FT (*lower bound*)	44.59MB	26.6	18.3	12.2	31.3	21.0	13.4	25.6	17.5	11.4
LwF (Rebuffi et al., 2017) (CVPR'17)	44.59MB	24.0	21.1	17.4	36.9	34.7	28.0	23.9	21.5	16.3
SIW (Belouadah et al., 2020) (BMVC'20)	44.59MB	38.3	35.2	26.8	66.4	55.7	41.4	38.6	30.9	17.2
$LUCIR$ (Hou et al., 2019) (CVPR'19)	89.18MB	50.4	37.4	24.4	89.5	75.0	50.5	54.9	39.4	24.8
SPB-M (Wu et al., 2021) (ICCV'21)	89.18MB	38.9	37.3	30.4	81.6	70.4	57.1	46.7	39.6	29.8
$PASS$ (Zhu et al., 2021) (CVPR'21)	89.18MB	39.4	35.9	29.8	65.0	55.1	42.3	48.0	40.9	31.8
$REMIND$ (Hayes et al., 2020) (ECCV'20)	89.18MB	52.2	44.8	35.9	83.3	77.5	72.2	50.6	39.4	31.3
$FeTrIL$ (Petit et al., 2023) (WACV'23)	44.59MB	54.7	49.1	**42.8**	86.1	81.2	77.7	52.9	45.3	**38.7**
$DSLDA$ (Hayes & Kanan, 2020) (CVPRW'20)	45.59MB	51.3	45.4	39.2	82.7	78.5	74.5	49.7	42.1	34.8
w/ *PlaStIL*$_1$	93.44MB	56.8	50.1	42.2	86.8	82.1	76.1	53.8	45.6	36.6
w/ *PlaStIL*$_2$	87.52MB	58.3	50.6	42.5	87.8	82.1	76.0	56.1	46.2	36.9
w/ *PlaStIL*$_{all}$	91.18MB	57.7	49.8	41.9	86.9	81.3	75.5	56.2	46.3	37.4
$DeeSIL$ (Belouadah & Popescu, 2018)	44.59MB	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ *PlaStIL*$_1$	88.44MB	58.6	**51.8**	41.9	92.1	**86.4**	**78.1**	56.8	**47.5**	36.4
w/ *PlaStIL*$_2$	84.52MB	**59.2**	50.2	39.7	**92.2**	85.1	76.7	57.3	46.9	36.0
w/ *PlaStIL*$_{all}$	89.18MB	57.7	48.6	39.0	90.7	83.4	75.3	**58.2**	47.1	35.6
$Joint$ (*upper bound*)			73.0			97.4			75.6	

Table 1: Average top-1 accuracy with three numbers of states K per dataset. *PlaStIL* is applied on top of $DeeSIL$ and $DSLDA$. **Best results - in bold**, second best - underlined.

tIL$_{all}$ provides optimal transfer since all layers are trained with new data but can accommodate only the current model. *PlaStIL*$_2$ provides a compromise between top depth and the number of storable models.

We also provide results for: (1) vanilla fine-tuning (FT) - a baseline that does not counter catastrophic forgetting at all, and (2) $Joint$ - an upper bound which consists of a standard training in which all data are available at once.

4.3 Implementation

A ResNet-18 (He et al., 2016) architecture was used as a backbone in all experiments. All methods were run with the published optimal parameters and minor adaptation of the codes to unify data loaders: FT (Belouadah et al., 2020), LwF (Rebuffi et al., 2017), $LUCIR$ (Hou et al., 2019), SIW (Smith et al., 2021), $DSLDA$ (Hayes & Kanan, 2020), $DeeSIL$ (Belouadah & Popescu, 2018) and $FeTrIL$ (Petit et al., 2023). SPB-M (Wu et al., 2021) has no public implementation and we reimplemented the method. We verified its correctness by comparing the accuracy obtained with our implementation (60.1) to the original one (59.7) for $ILSVRC$ split tested by the authors (Wu et al., 2021). All methods were implemented in PyTorch (Paszke et al., 2019), except for LwF which uses the Tensorflow (Abadi et al., 2015) implementation of LwF from Rebuffi et al. (2017) because it provides better performance compared to later implementations Javed & Shafait (2018); Wu et al. (2019). The training procedure from Hayes et al. (2020) was used to obtain initial models for all transfer-based methods. These initial models were trained for 90 epochs, with a learning rate of 0.1, a batch size of 128, and a weight decay of 10^{-4}. We used stochastic gradient descent for optimization and divided the learning rate by 10 every 30 epochs. Detailed parameters are presented in the appendix.

4.4 Main results

The results presented in Table 1 show that all *PlaStIL* variants improve over the transfer-based methods to which they are added for all tested datasets and CIL configurations. The gains are generally higher for $K = 5$ states, but remain consistent for the $K = \{10, 20\}$. For instance, *PlaStIL*$_1$ gains 6.2, 6.4 and 4.4 points for $ILSVRC$ split into 5, 10, 20 states, respectively. The best overall performance is obtained with *PlaStIL*$_1$, followed by *PlaStIL*$_2$ and *PlaStIL*$_{all}$ applied on top of $DeeSIL$. A combination of recent model tops which fine-tune only the last convolutional layer is best here. Our method appled to $DeeSIL$provides best performance for 5 and 10 incremental states. Gains are equally interesting for $DSLDA$, and particularly for $K = 20$. This baseline has better performance than $DeeSIL$for all three datasets when $K = 20$. The application of *PlaStIL*on top of $DSLDA$leads to slightly better performance compared to the version built on top of $DeeSIL$for $ILSVRC$ (42.5 vs. 41.9) and $iNaturalist$ (37.4 vs 36.4). The better behavior of $DSLDA$ for longer incremental sequences is explainable since this method features a global inter-class separability component. In contrast, $DeeSIL$ only separates classes within each state and its discriminative power is reduced when each state includes a low number of classes.

Distillation-based methods have lower performance compared to transfer-based methods for the tested large-scale datasets. This result is coherent with previous findings regarding scalability problems of distillation (Belouadah et al.,

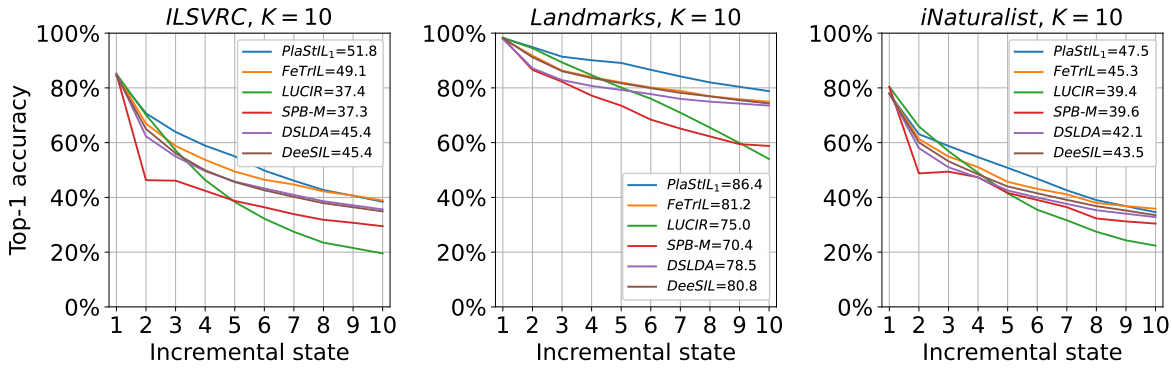

Figure 3: Incremental accuracy across all states for $K = 10$. Plots for for $K = 5$ and $K = 20$ are presented at Figure 7 in the appendix. Plots are presented for the best methods from Table 1.

2021; Masana et al., 2021; Prabhu et al., 2020). The difference between *PlaStIL* applied to $DeeSIL$ and $DSLDA$ and distillation-based methods is very consequent. It is in the double-digits range compared to $LUCIR$, the best distillation-based method, for five configurations out of nine tested. This difference reaches a maximum value of 27.6 top-1 accuracy points for $Landmarks$ with $K = 20$ states and a minimum of 2.7 points for the same dataset with $K = 5$ states. $SPB\text{-}M$ is a recent method which compares very favorably with $LUCIR$ when half of the dataset is allowed in the initial state (Wu et al., 2021). However, its behavior is globally similar to that of $LUCIR$, with performance gains for $K = 20$ states and losses for $K = 5$. This happens because $SPB\text{-}M$ is more dependent on the representativeness of the initial model compared to $LUCIR$ since it features a strong stability component. LwF and SIW, the two other methods which update models in each incremental state have even lower performance than $LUCIR$ and $SPB\text{-}M$. The comparison with $REMIND$ is also favorable in all tested configurations. This indicates that, given an identical memory budget, the use of model tops is more effective than the use of a replay buffer made of compressed samples of the past . *PlaStIL* is also better than $FeTrIL$ on 7 configurations out of 9, and ranks similarly (0.3% and 0.7%) on the remaining two. It can be explained by the fact that on the long run the pseudo-feature generator of $FeTrIL$ favors the stability. However, on shorter runs, for $K = 5$ states, *PlaStIL* beats it by 4.5%, 6.1% and 5.3%. All EFCIL methods need a supplementary budget to ensure the plasticity-stability balance. The takeaway from the comparison of *PlaStIL* with mainstream methods is that this budget is much better spent on partially fine-tuned model tops than on storing the previous model needed for distillation.

The results per dataset show that $ILSVRC$ and $iNaturalist$ are harder to solve compared to $Landmarks$. The gains obtained by *PlaStIL* are smaller for $Landmarks$ since the progress margin is more reduced. The performance gap between the evaluated methods and $Joint$ is large. The fact that the gap widens with the number of incremental states has specific explanations for the two types of methods. Past knowledge becomes harder to preserve when the number of fine-tuning rounds increases in $LUCIR$, $SPB\text{-}M$, SIW, LwF. For transfer-based methods, past knowledge is encoded using a weaker feature extractor. *PlaStIL* reduces the gap with $Joint$, but the results reported here show that exemplar-free class-incremental learning remains a very challenging problem.

We propose a more detailed presentation of the incremental accuracy in Figure 3. These results confirm the large gap between the proposed methods and distillation-based ones. $SPB\text{-}M$ has lower performance at the start of the process, but then becomes better than $LUCIR$, because of the representativeness of the initial model, as previously explained.

4.5 Method analysis

We conduct an analysis of *PlaStIL* in terms of model footprint and number of operations needed to infer test image predictions. These experiments are run using *PlaStIL* applied on top of $DeeSIL$ since this variant has the best overall results. They are important in order to highlight the merits and limitations of the proposed method.

Model footprint. Incremental learning algorithms are particularly useful in memory-constrained environments. Their model footprint is thus an important characteristic. Distillation-based methods require the storage of models $\mathcal{M}_k$ and $\mathcal{M}_{k-1}$ to preserve past knowledge. Transfer-based methods only need $\mathcal{M}_1$, but they optimize stability at the expense of plasticity. The three versions of *PlaStIL*, whose performance is presented in Table 1, store $\mathcal{B}_1$, the initial model base and a variable number of model tops. Each of the four recent model tops used by *PlaStIL*$_1$ fine-tunes the last convolutional layer of ResNet-18 (He et al., 2016), which accounts for 21.45% of total number of the model's

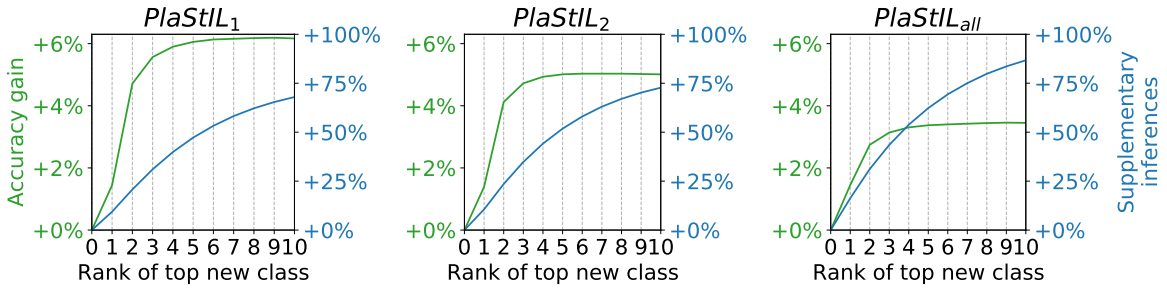

Figure 4: Top-1 accuracy gains obtained with the three variants of *PlaStIL* applied to $DeeSIL$ with different thresholds for the rank of the top new class among the predictions generated with Equation 4. Results are shown for $ILSVRC$ with $K = 10$ states. The corresponding percentage of supplementary inferences needed for each threshold is also plotted. Interesting gains are obtained starting with a recent class predicted in the second position, which requires approximately 25% of supplementary inferences for *PlaStIL*$_1$. *Best viewed in color.*

parameters. The parametric footprint of *PlaStIL*$_1$ is thus lower than that of distillation-based methods. *PlaStIL*$_2$ has the same footprint as *PlaStIL*$_1$ since it stores two tops which account for 42.9% of ResNet-18 parameters each. As an ablation of *PlaStIL*, we also present results with a single model top, regardless of its fine-tuning depth .Naturally, *PlaStIL*$_{all}$ obtains the best results in this configuration, but interesting gains are still obtained with model tops which fine-tune one or two final convolutional layers in *PlaStIL*$_1$ and *PlaStIL*$_2$, respectively.

Inference complexity. The classification layer defined in Equation 5 is fed with features from the initial and the updated model top(s), for past and recent classes, respectively. By default, *PlaStIL* inferences requires an extraction of features for all used model tops. This supplementary inferences cost varies for *PlaStIL* variants due to the different number of parameters in the model top(s) that they use. This supplementary computational cost can be reduced if predictions are first computed using the initial model only, as defined in Equation 4. Then, subsequent model tops are used only if one of their classes is strongly activated for the test image. A top is used for inferences only if at least one of its associated classes is ranked among the top classes of the of the classification layer from Equation 4. The closer to 1 this rank threshold is, the smaller the added computational cost will be since fewer tops are likely to be used for inference. However, a restriction to small ranks might also discard useful model tops and thus reduce the positive effect of *PlaStIL*. Evaluation is done with top ranks of new class in $\mathcal{W}_k^{fix}$ between 1 and 10 to examine the trade-off between inference complexity and performance. The obtained accuracy, as well as the added inference cost, for the three variants of *PlaStIL* applied over $DeeSIL$ are presented in Figure 4. The results show that performance gains relative to $DeeSIL$ raise sharply. The best balance between performance gains and added costs is obtained for *PlaStIL*$_1$. This is explained by the fact that $\mathcal{T}$ have the lowest number of parameters for this variant. Their activation can be done in a finer manner, resulting in a reduced overall inference cost. Interesting *PlaStIL* gains are obtained starting with a recent class being ranked second position by $DeeSIL$. The accuracy curve becomes practically flat if a recent class is ranked beyond the third position by the baseline. The results presented in Figure 4 provide further support to the fact that *PlaStIL*$_1$ is the most appropriate choice as plasticity layer added on top of $DeeSIL$.

Choice of model tops. The main experiments used model tops created for the most recent incremental states. Other top creation and removal policies are possible, and we compare the one proposed here with an oracle which performs an optimal selection of tops. The oracle selects model tops associated to different states so as to maximize the average incremental accuracy of the CIL process by aggregating the accuracy of different incremental states computed on the test set. The results from Table 2 indicate that the creation of tops for the most recent states provides a performance level which is close to the optimal one achievable by the oracle. This is probably explained by the fact that the evaluated CIL scenarios use a random assignment of classes to states. Other selection strategies might be more appropriate if the assumptions made about the order of arrival of classes or data were different, as discussed in Subsection 3.2.

5 Conclusion

We proposed a new method which adds a plasticity layer to transfer-based methods in exemplar-free CIL. Plasticity is improved by training dedicated model tops which fine-tune a variable number of deep models layers for one or more recent incremental states. The predictions of the different model tops used by our method are integrated in a hybrid classification layer. Model tops improve accuracy compared to the transfer-based method to which they are added to in order improve plasticity. These improvements are obtained by introducing supplementary parameters so

CIL Method	*ILSVRC*			*Landmarks*			*iNaturalist*		
	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
DeeSIL (Belouadah & Popescu, 2018)	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ *PlaStIL*$_1$	58.6	51.8	41.9	92.1	86.4	78.1	56.8	47.5	36.4
w/ *PlaStIL*$_1$+oracle	58.6	51.8	42.1	92.1	86.4	78.7	56.8	47.8	36.8
w/ *PlaStIL*$_2$	59.2	50.2	39.7	92.2	85.1	76.7	57.3	46.9	36.0
w/ *PlaStIL*$_2$+oracle	59.3	50.3	40.1	92.2	85.2	77.3	57.5	47.1	36.5
w/ *PlaStIL*$_{all}$	57.7	48.6	39.0	90.7	83.4	75.3	58.2	47.1	35.6
w/ *PlaStIL*$_{all}$+oracle	57.9	48.7	39.3	90.7	83.7	75.8	58.3	47.5	36.0

Table 2: Comparison of *PlaStIL* with a version that knowing the final composition of the different states will only fine-tune the relevant states (*PlaStIL* +oracle). The average gains are of +0.2% with *PlaStIL* +oracle.

as to have a total footprint which remains lower than that of distillation-based methods. The comparison of these methods with *PlaStIL* is clearly favorable to the latter. The takeaway is that the parameters allocated to the previous model in distillation-based approaches are better spent on partially fine-tuned model tops. This finding is aligned with those reported in recent comparative studies which question the usefulness of distillation in large-scale CIL with or without memory (Belouadah et al., 2021; Masana et al., 2021; Prabhu et al., 2020). We believe that future studies should consider transfer-based methods to assess progress in exemplar-free CIL in a fair and comprehensive manner. We plan to optimize the stability-plasticity compromise for EFCIL to further improve the encouraging results reported here. First, we will try to devise a classifier which predicts the most probable initial state for each test sample, following the proposal made in Verma et al. (2021). If successful, this prediction would reduce the classification space to individual states and remove the need for a hybrid classification layer. Second, we will investigate the use of pretrained representations learned with larger amounts of supervised or semi-supervised data, as proposed in Dhamija et al. (2021); Hayes et al. (2020); Wang et al. (2023). Such models can be seamlessly integrated in the *PlaStIL* pipeline.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Eden Belouadah and Adrian Popescu. Deesil: Deep-shallow incremental learning. *TaskCV Workshop @ ECCV 2018.*, 2018.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. Initial classifier weights replay for memoryless class incremental learning. In *British Machine Vision Conference (BMVC)*, 2020.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.

Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, pp. 241–257, 2018.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pp. 556–572. Springer, 2018.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255, 2009.

Akshay Raj Dhamija, Touqeer Ahmad, Jonathan Schwan, Mohsen Jafarzadeh, Chunchun Li, and Terrance E Boult. Self-supervised features improve open-world learning. *arXiv preprint arXiv:2102.07848*, 2021.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. *CoRR*, abs/1811.08051, 2018.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer vision-ECCV 2020-16th European conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, volume 12365, pp. 86–102. Springer, 2020.

Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 220–221, 2020.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pp. 466–483. Springer, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 831–839, 2019.

Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. *CoRR*, abs/1807.02802, 2018.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *CoRR*, abs/1909.08383, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, ECCV, 2016.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7765–7773, 2018.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification, 2021.

Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11): 2624–2637, 2013.

M Mermillod, A Bugaiska, and P Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504–504, 2013.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pp. 3476–3485. IEEE Computer Society, 2017.

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3911–3920, January 2023.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pp. 524–540. Springer, 2020.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Conference on Computer Vision and Pattern Recognition Workshop*, CVPRW, 2014.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. *arXiv preprint arXiv:2106.09701*, 2021.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pp. 270–279. Springer, 2018.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. *CoRR*, abs/2103.13558, 2021.

Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *2023 AAAI Conference*, 2023.

Guile Wu, Shaogang Gong, and Pan Li. Striking a balance between stability and plasticity for class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1124–1133, 2021.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 374–382, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 6980–6989. IEEE, 2020.

Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S. Davis. M2KD: multi-model and multi-level knowledge distillation for incremental learning. *CoRR*, abs/1904.01769, 2019.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5871–5880, June 2021.

A Appendix

In this appendix, we provide:

- implementation details for all tested approaches;
- details about datasets used in experiments;
- results for features transferability among datasets.
- discussion on the different disk-memory usages
- additional detailed comparative accuracies

A.1 Implementation details

As already mentioned in Section 4.3, we used the authors' optimal parameters to run all baselines. A ResNet-18 model (He et al., 2016) and an SGD optimizer with $momentum = 0.9$ are used for all methods. We explicitly list the learning parameters of each method hereafter:

A.1.1 Learning from scratch

This type of learning is used to train models of the initial state (because it is not incremental), and also $Joint$, the upper bound method where all classes are learned with all their data at once.

Following Belouadah et al. (2020), $Joint$ and the first models of FT and SIW (Belouadah et al., 2020) are run for 120 epochs using $batch\ size = 256$ and $weight\ decay = 0.0001$. The lr is set to 0.1 and is divided by 10 when the error plateaus for 10 epochs.

For $REMIND$ (Hayes et al., 2020), $Deep\text{-}SLDA$ (Hayes & Kanan, 2020), $DeeSIL$ (Belouadah & Popescu, 2018), $Fixed_{NCM}$ (Rebuffi et al., 2017), $FeTrIL$ (Petit et al., 2023) and $PlaStIL$ (ours), we follow Hayes et al. (2020) and run the model for 90 epochs using $lr = 0.1$, $batch\ size = 128$, $weight\ decay = 0.00001$. The lr is set to its initial value decayed by 10 every 30 epochs. The lr is constrained to do not decrease beneath 0.001.

For $LUCIR$, LwF, the first model is trained in the same manner than subsequent models (detailed below), following the authors of Hou et al. (2019); Rebuffi et al. (2017).

A.1.2 Incremental Learning

Here, we describe the hyper-parameters used to train the models incrementally for model-update-based methods.

- FT (Belouadah et al., 2020) - IL models are trained for 35 epochs with $batch\ size = 256$, $momentum = 0.9$ and $weight\ decay = 0.0001$. The learning rate is set to $lr = 0.1/t$ at the beginning of each incremental state ($t \geq 2$) and is divided by 10 when the error plateaus for 5 consecutive epochs.
- LwF (Rebuffi et al., 2017) - all models are trained for 60 epochs using $lr = 1.0$, $batch\ size = 128$, and $weight\ decay = 0.0001$. The learning rate is divided by 5 at epochs 20, 30, 40, and 50.
- $LUCIR$ (Hou et al., 2019) - all models are trained for 90 epochs using $lr = 0.1$, $batch\ size = 128$ and $weight\ decay = 0.0001$. The lr is divided by 10 at epochs 30 and 60. The method-specific parameters are the same as those from the original paper (Hou et al., 2019) and can also be found once we release the codes and configuration files.
- SIW (Belouadah et al., 2020) - is trained using the same hyper-parameters of FT following the authors.

Init. dataset	*Landmarks*	*iNaturalist*	*ILSVRC*	*iNaturalist*	*Landmarks*	*ILSVRC*
IL dataset	*ILSVRC*		*Landmarks*		*iNaturalist*	
Deep-SLDA (Hayes & Kanan, 2020)	19.3	29.2	67.8	60.1	16.1	35.2
DeeSIL (Belouadah & Popescu, 2018)	21.6	29.6	70.5	61.9	16.2	36.1
DeeSIL w/ *PlaStIL*$_1$	**28.1**	**33.8**	**76.8**	**68.3**	**21.4**	**39.2**
Joint	72.98		97.41		75.60	

Table 3: Average top-1 incremental accuracy in a dataset transfer learning configuration. Results are given for transferring between all pairs of initial and target datasets. All experiments are run with $K = 10$ states. **Best results are in bold**.

Dataset	#Train	#Test	μ(Train)	σ(Train)
ILSVRC (Russakovsky et al., 2015)	1,231,167	50,000	1231.2	70.2
Landmarks (Noh et al., 2017)	374,367	20,000	374.4	103.8
iNaturalist (Van Horn et al., 2018)	300,00	10,000	300.0	0.0

Table 4: Summary of datasets. μ is the mean number of train images per class and σ is the standard deviation

- *SPB-M* (Wu et al., 2021) - all models are trained for 90 epochs using $lr = 0.1$, $batch\ size = 128$ and $weight\ decay = 0.0001$. The lr is divided by 10 at epochs 30 and 60. The method-specific parameters are the same as those from the original paper (Wu et al., 2021) and can also be found once we release the codes and configuration files. Note that *SPB-M* (Wu et al., 2021) has no available code, we implemented it by modifying significantly the code of *LUCIR* (Hou et al., 2019), and verified its correctness on the results the authors provided in their paper.
- *REMIND* (Hayes et al., 2020) - method-specific parameters are the same as those from the original paper, and run from the available code, using the advised version of pytorch

We reimplemented the method, and verified its correct functioning using the protocol from the original paper. If the paper is accepted we will publish the code of every implementation we wrote in order to facilitate reproducibility.

A.2 Datasets details

The datasets used in evaluation are designed for three visual classification tasks: object, natural species and landmark recognition. Their main statistics are in tables 4.

A.3 Feature transferability.

The results from Table 1 show that transfer-based methods work well when features are transferred within the same dataset. In Table 3, we examine the behavior of *PlaStIL* in a transfer scenario which involves a domain gap. The transfer is done between all pairs of initial and incremental datasets. The results show that *PlaStIL*$_1$ is more resilient to transfer compared to the transfer-based baselines. The best results are obtained with *ILSVRC* as initial models and *iNaturalist* and *Landmarks* as incremental datasets. This is intuitive since *ILSVRC* contains more diversified concepts and produces more generic features. It also has more samples per class compared to the other two datasets and its feature extractor is more robust. The accuracy obtained with a transfer from *ILSVRC* is comparable with that of best distillation-based methods from Table 1. The results with *iNaturalist* as initial dataset are lower, but still interesting. Performance is lower when the domain shift between the initial and the target datasets is more important and if the initial model is learned on domain-specific data. This is, for instance, the case when *Landmarks* is used to train the initial model since this dataset only includes geographic landmarks. Note that transfer would be more efficient if larger initial datasets were used to reinforce the initial representation, as proposed in Belouadah & Popescu (2018); Hayes & Kanan (2020). However, for fairness, our focus is on transfer experiments which use the same number of initial classes as in Table 1.

A.4 Disk-memory usage.

The results of Figure 6 demonstrate that as the number of tops increases, the accuracy of the model improves. However, it is important to note that this improvement in accuracy comes at the cost of increased memory usage on disk. This trade-off between accuracy and resource usage is a crucial consideration for optimizing model performance in practice in an exemplar-free setting.

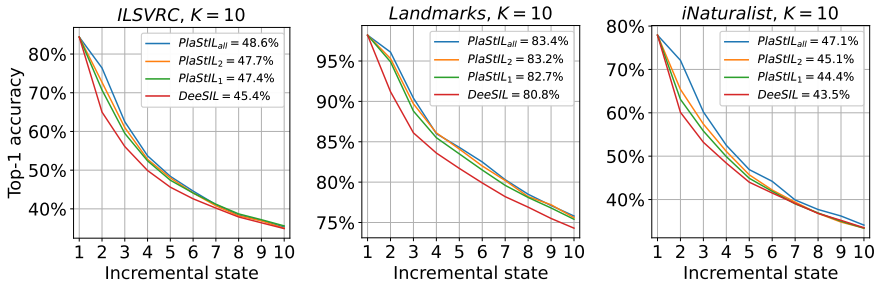

Figure 5: Top-1 incremental accuracy of three versions of *PlaStIL* applied to $DeeSIL$ when using a single model top with variable fine-tuning depth. $DeeSIL$ is a limit case in which the whole feature extractor is frozen. *Best viewed in color.*

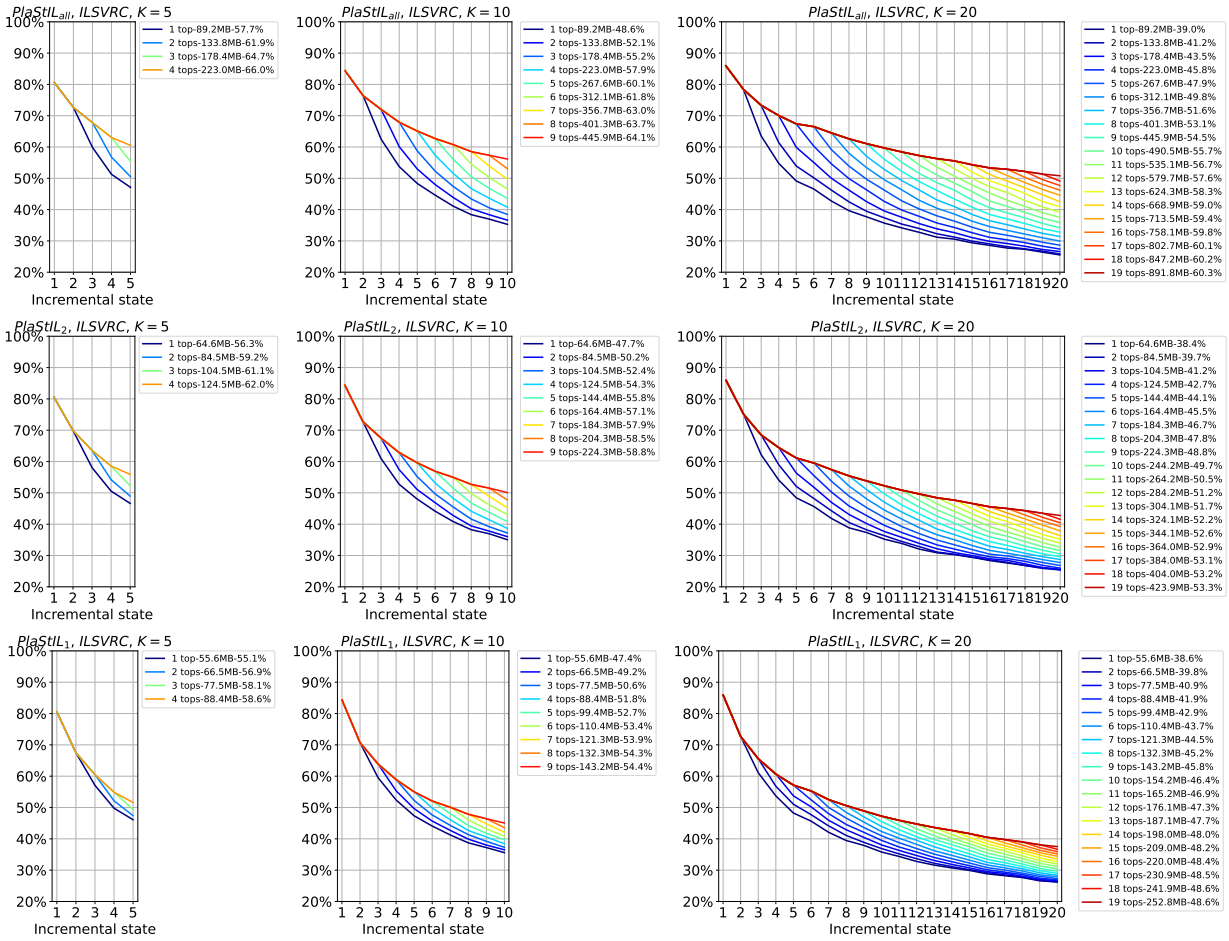

Figure 6: Effect of varying the number of additional tops on incremental accuracy, on $ILSVRC$ with $K \in \{5, 10, 20\}$. As the number of tops increases, the accuracy of the model improves. However, this comes at the cost of increased memory usage on disk. *Best viewed in color.*

Results from Table 5 show that *PlaStIL*strategy of storing model tops suits better the experiments than storing compressed representation vectors, even for a larger memory on disk.

CIL Method	mem on disk	*ILSVRC*			*Landmarks*			*iNaturalist*		
		K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
REMIND (Hayes et al., 2020) (ECCV'20)	89.18MB	52.2	44.8	35.9	83.3	77.5	72.2	50.6	39.4	31.3
REMIND (Hayes et al., 2020) (ECCV'20)	133.78MB	52.3	44.9	36.2	83.4	79.3	75.8	50.9	43.8	35.0
REMIND (Hayes et al., 2020) (ECCV'20)	222.96MB	52.3	44.4	**39.7**	84.2	81.0	78.0	53.7	46.5	**37.8**
DeeSIL (Belouadah & Popescu, 2018)	44.59MB	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ *PlaStIL*$_1$	88.44MB	58.6	**51.8**	41.9	92.1	**86.4**	**78.1**	56.8	**47.5**	36.4
w/ *PlaStIL*$_2$	84.52MB	**59.2**	50.2	**39.7**	**92.2**	85.1	76.7	57.3	46.9	36.0
w/ *PlaStIL*$_{all}$	89.18MB	57.7	48.6	39.0	90.7	83.4	75.3	**58.2**	47.1	35.6

Table 5: Average top-1 accuracy with three numbers of states K per dataset, comparison with *REMIND* with different budgets for the storage of their compressed vectors (1, 2 and 4 times the size of a ResNet18 on disk). **Best results - in bold**, second best - underlined.

A.5 Detailed comparative accuracies.

In this section, we presented the incremental accuracy across all states for $K = 5$ and $K = 20$. These results confirm the large gap between the proposed methods and distillation-based ones.

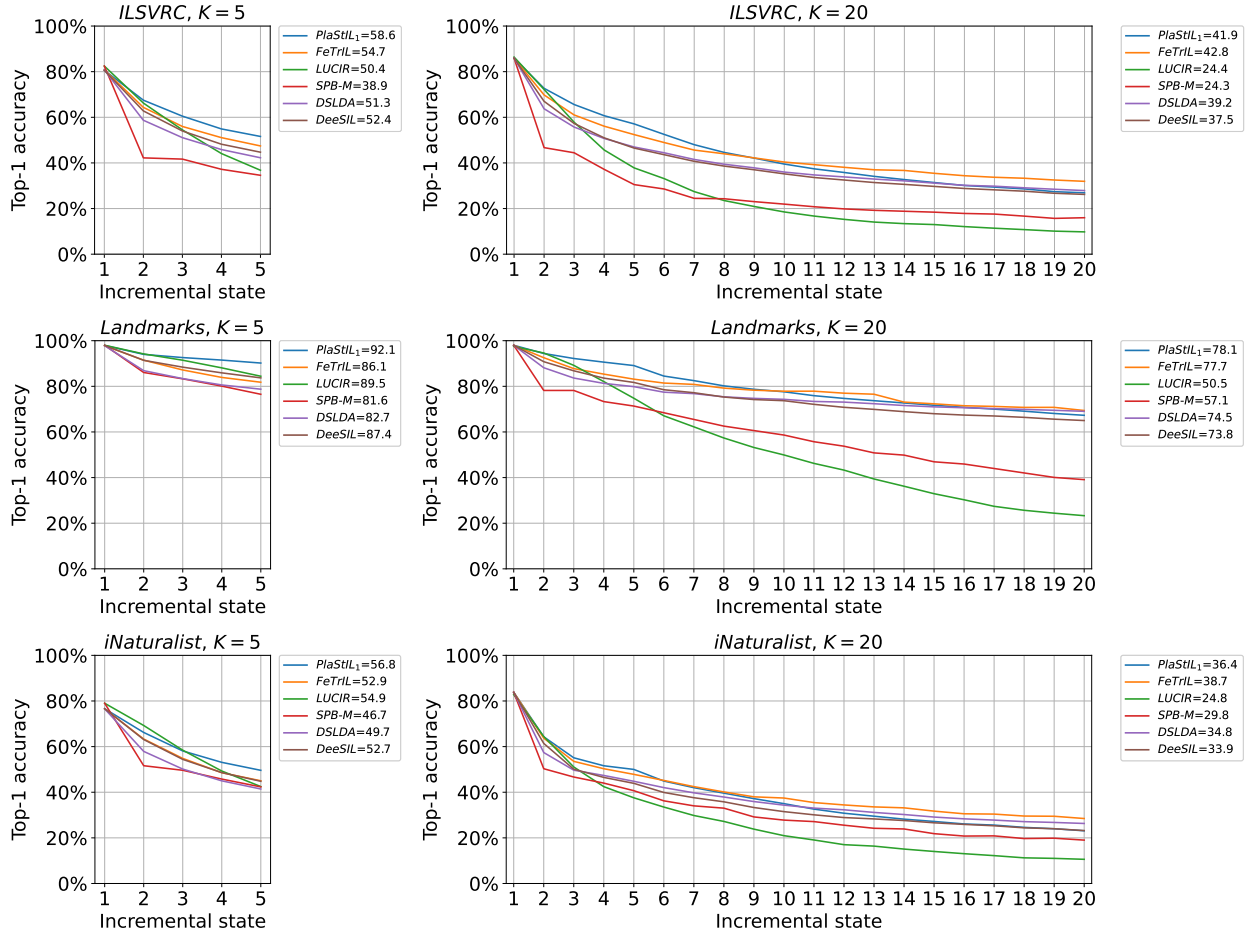

Figure 7: Incremental accuracy across all states for $K = 5$ and $K = 20$. Plots for $K = 10$ are presented in Figure 3 in the main paper. Plots are presented for the best methods from Table 1. *Best viewed in color.*