ScaIL: Classifier Weights Scaling for Class Incremental Learning

Eden Belouadah, Adrian Popescu
Université Paris-Saclay, CEA, Département Intelligence Ambiante et Systèmes Interactifs,
91191 Gif-sur-Yvette, France
eden.belouadah,adrian.popescu@cea.fr

Abstract

Incremental learning is useful if an AI agent needs to integrate data from a stream. The problem is non trivial if the agent runs on a limited computational budget and has a bounded memory of past data. In a deep learning approach, the constant computational budget requires the use of a fixed architecture for all incremental states. The bounded memory generates imbalance in favor of new classes and a prediction bias toward them appears. This bias is commonly countered by introducing a data balancing step in addition to the basic network training. We depart from this approach and propose simple but efficient scaling of past classifiers' weights to make them more comparable to those of new classes. Scaling exploits incremental state statistics and is applied to the classifiers learned in the initial state of classes to profit from all their available data. We also question the utility of the widely used distillation loss component of incremental learning algorithms by comparing it to vanilla fine tuning in presence of a bounded memory. Evaluation is done against competitive baselines using four public datasets. Results show that the classifier weights scaling and the removal of the distillation are both beneficial.

1. Introduction

Artificial agents are often deployed in dynamic environments in which information often arrives in streams [22]. For instance, this is the case of a robot which operates in an evolving environment or of a face recognition app which needs to deal with new identities. In such settings, an incremental learning (IL) algorithm is needed to increase the recognition capacity when integrating new data. There was recently a strong regain of interest for IL with the adaptation of deep learning methods [2, 5, 26, 28]. Incremental learning is non-trivial if the artificial agents have limited computational and memory budgets. If the memory of past classes is bounded or unavailable, the system underfits past data when new information is integrated and catastrophic forgetting [19] occurs. Since a joint optimization of computational and memory requirements is hard, if not impossible, existing IL algorithms focus on one of these two aspects. In a first scenario [2, 18, 28, 30], the number of deep model parameters is allowed to grow and no memory is used. In a second scenario [5, 7, 10, 26, 34], the deep architecture is fixed and a memory is introduced for past class exemplars to alleviate the effect of catastrophic forgetting. These algorithms update deep models by adapting the fine tuning procedure to include classification and distillation losses.

We focus on this second scenario and introduce $ScaIL$, a method which reduces the bias in favor of new classes by exploiting the classifier weights of past classes as learned in their initial state with all class data available. Since past class classifiers are learned in different previous IL states, they are reshaped to be usable in the current state. Their scaling uses aggregate statistics from the current and initial states. $ScaIL$ is illustrated in Figure 1 with a toy example which includes an initial and two incremental states. In addition to the bounded exemplar memory $\mathcal{B}$, $ScaIL$ requires the use of a compact memory $\mathcal{I}$ which stores the classifier weights from the initial states of past classes. A second contribution, of practical importance, is to simplify the deep model update across incremental states. The widely used distillation loss term [5, 7, 10, 26, 34] is ablated here and model updates are done with vanilla fine tuning.

Evaluation is done with four public datasets and three values for the number of incremental states $\mathcal{Z}$ and the exemplar memory $\mathcal{B}$, the two key components of class IL algorithms. $ScaIL$ is compared to strong baselines from literature and to new ones proposed here and the obtained results indicate that it has the best overall performance.

2. Related Works

Incremental learning is an open research topic which recently witnessed a regain of interest with the use of deep learning algorithms. We discuss three groups of methods which focus on different parts of IL. Due to limited space, only a representative subset of methods is described.

A first group increases the number of parameters of deep architectures to accommodate new classes. Growing a Brain [30] increases the depth and/or the width of the

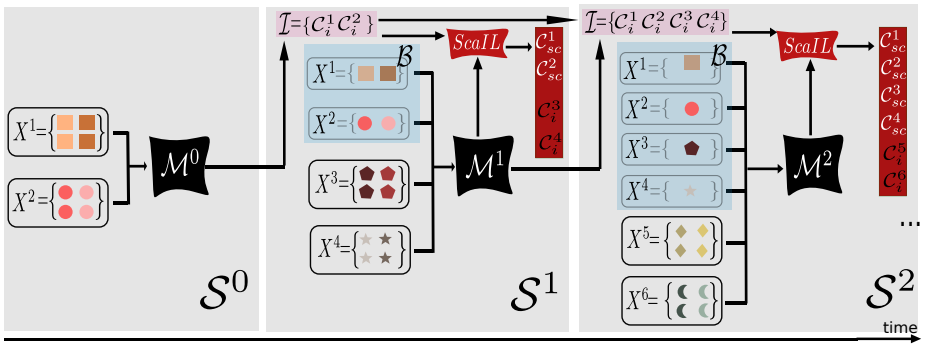

Figure 1: Illustration of $ScaIL$. States are noted $\mathcal{S}^k$, image data X^j, deep models $\mathcal{M}^k$ and classifier weights C_i^j and C_{sc}^j, where: j is the class label, i is the initial state in which the classifier was learned with all data and sc means that the classifier was scaled using $ScaIL$. We represent three states $\mathcal{S}^0$, $\mathcal{S}^1$ and $\mathcal{S}^2$ which recognize $2, 4$ and 6 classes respectively. The bounded memory (light blue), is fixed at $\mathcal{B} = 4$ past classes exemplars. As the training advances, the data imbalance between past and new classes grows due to bounded $\mathcal{B}$ and the prediction bias in favor of new classes becomes more prominent. $ScaIL$ reduces this bias by making classifier weights of past and new classes more comparable by using a small memory $\mathcal{I}$ which stores initial classifiers C_i^j. In each IL state, $ScaIL$ replaces the raw classifiers of past classes provided by the model $\mathcal{M}^k$ by C_{sc}^j, a scaled version of C_i^j, the initial classifier. Since $ScaIL$ combines classifiers learned in different IL states, initial classifiers are reshaped using aggregate statistics from the current and the initial states. The classifiers for newly learned classes are left as learned by the current model $\mathcal{M}^k$. *Best viewed in color.*

layers to integrate new classes. In Progressive Neural Networks [28], a new network is added for each new task and lateral connections are used between all networks to share the representation. $PackNet$ [18] uses weight pruning techniques to free up redundant network parameters. When new classes arrive, the freed up parameters are attributed to the new task. The number of parameters grows slowly but only a limited number of new tasks can be added. These algorithms are a good choice if the complexity of deep models can grow across incremental states. However, they increase the model memory footprint and slow down inference, especially for a large number of incremental states.

A second, less frequent group, is based on fixed representations. Here, the feature extractor network does not evolve across IL states. $FearNet$ [11] is biologically inspired by the functioning of human brain. The incremental learning process is implemented with three networks which model short and long term memory and a decision network to choose the activated network. The main drawback of $FearNet$ is that memory grows in a nearly linear fashion across IL states because detailed statistics about past classes are needed. $DeeSIL$ [3] is an adaptation of transfer learning to a class IL context [13, 15]. It learns a fixed representation on the first state and deploys a set of SVMs to increment recognition capacity afterwards. The reported top-5 accuracy on ILSVRC [27] with a bounded memory $\mathcal{B} = 20000$ exemplars is 74.7%. A fixed representation is tested in [25] but its past classes are unnecessarily relearned only with exemplars in each IL states and results are suboptimal. The main advantage of fixed representations is that all positive examples can be used for all classes since the deep model does not evolve across incremental states. However, performance depends heavily on the quality of the initial representation. If the representation is learned on small dataset or if the new classes are significantly different from the initial ones, the generalization capacity is low.

A third influential group of algorithms updates deep models across incremental states using an adapted fine tuning procedure. These algorithms are inspired by Learning-without-Forgetting (LwF) [16], which introduces a distillation loss term to handle catastrophic forgetting in absence of a memory of past classes. This term encourages the network to reproduce the same outputs for past classes in the current state as in past ones. We discuss some representative adaptations of distillation to IL hereafter. $iCaRL$ [26] implements LwF using binary cross-entropy loss, which operates independently on class outputs. The use of this loss is not clearly justified but it is assumed to cope with class imbalance [17]. Also, the authors modify the distillation term to use sigmoids instead of the standard softened softmax targets. $iCaRL$ adds the following steps for an efficient adaptation to an IL context: (1) exploit a bounded memory for past classes, (2) select exemplars using a herding mechanism which approximates the real class mean [31] and (3) replace the outputs of the deep models by a Nearest-Exemplars-Mean (NEM) external classifier, an adaptation of nearest-class-mean [20], to tackle class imbalance. $iCaRL$ top-5 accuracy reaches 62.5% on ILSVRC [27] dataset with a memory of $\mathcal{B} = 20000$ exemplars. The authors also conclude that vanilla fine tuning is not fitted for IL with bounded memory. However, their main experiment tests $iCaRL$ with memory and vanilla fine

tuning without memory and the comparison is not fair. In an additional experiment, they compare the two methods only on a small scale dataset and, while $iCaRL$ remains superior, the gap between the two methods is much smaller. End-to-end incremental learning [5] uses a distillation component which is closer to the original definition from [9]. The authors exploit standard cross-entropy loss and their basic distilled network has performance similar to that of $iCaRL$. A balanced fine tuning step is added to tackle data imbalance and data augmentation is also used. As a result, the method gains 7 points over $iCaRL$ on ILSVRC with $\mathcal{B} = 20000$. In [10], the authors show that the use of higher temperature to soften distillation helps to some extent. Very recently, the authors of [34] introduced a multi-model and multi-level distillation for IL. The method incorporates knowledge from all previous incremental states and not only from the latest one. A performance improvement of 3 to 5 points over $iCaRL$ is reported. Generative Adversarial Networks were also considered as a mean to generate image exemplars for past classes instead of storing them directly [7]. While the approach is appealing, the quality of generated images is still insufficient. A combination of generated and real images was necessary to slightly enhance performance over $iCaRL$. BiC [32] is a very recent approach that handles catastrophic forgetting by adding a linear model after the last fully connected layer to correct the bias towards new classes. We add BiC to the results table for SotA completeness. Approaches from this group tend to cope well with the integration of new data but retraining the network at each incremental step is costly.

3. Proposed Method

3.1. Class IL Problem Formalization

We focus on IL with constant model complexity, $\mathcal{Z}$ incremental states and a bounded memory $\mathcal{B}$ of past classes. The proposed formalization is adapted from [5, 7, 26]. We note: $\mathcal{S}^k$ - the incremental state, N_k - the number of classes in $\mathcal{S}^k$, $\mathcal{X}^{N_k}$ - the training dataset in $\mathcal{S}^k$, $\mathcal{M}^k$ - the deep model and $\mathcal{C}^{N_k}$ - the classifier weights layer. The initial state $\mathcal{S}^0$ includes a dataset $\mathcal{X}^{N_0} = \{X^1, X^2, ..., X^{N_0}\}$ with $N_0 = P_0$ classes. $X^j = \{x_1^j, x_2^j, ..., x_{n_j}^j\}$ is the set of n_j training examples for the j^{th} class. An initial model $\mathcal{M}^0 : \mathcal{X}^{N_0} \rightarrow \mathcal{C}^{N_0}$ is trained to recognize N_0 classes using all data from $\mathcal{X}^{N_0}$. P_k new classes need to be integrated in each incremental state $\mathcal{S}^k$, with $k > 0$. Each IL step updates the previous model $\mathcal{M}^{k-1}$ into the current model $\mathcal{M}^k$ which recognizes $N_k = P_0 + P_1 + ... + P_k$ classes in incremental state $\mathcal{S}^k$. All data of the P_k new classes are available but only a bounded exemplar memory $\mathcal{B}$ of the N_{k-1} past classes is allowed. If memory allocation is balanced, each past class is represented by $\frac{\mathcal{B}}{N_{k-1}}$ exemplars. We note $\mathcal{M}^k : \mathcal{X}^{N_k} \rightarrow \mathcal{C}^{N_k}$ the model which transforms the $\mathcal{X}^{N_k}$ dataset into a set of raw classifiers $\mathcal{C}^{N_k} = \{C_k^1, C_k^2, ..., C_k^{N_{k-1}}, C_k^{N_{k-1}+1}, ..., C_k^{N_k}\}$. The classifier weights learned in state $\mathcal{S}^k$ for the j^{th} class are written $C_k^j = \{w^1(C_k^j), w^2(C_k^j), ..., w^D(C_k^j)\}$, where D is the size of the features extracted from the penultimate layer of $\mathcal{M}^k$.

3.2. Classifier Weights Scaling

Incremental learning algorithms strive to approach the performance of full learning, in which the entire training set is available for all classes at all times. When a bounded set of past exemplars is stored, a prediction bias toward new classes appears due to the data imbalance in their favor. This bias is illustrated in Figure 2(a) with the difference between mean raw predictions for past and new classes after incrementally fine tuning the $ILSVRC$ dataset [27] with $\mathcal{B} = 5000$ past exemplars. The average score difference in favor of new classes over all incremental states is 6.45 points. The prediction gap is due to the stronger activations of classifier weights for new classes compared to past classes, as illustrated by the blue and red curves from Figure 2(b). It is thus tempting to try to reshape the classification layers of past and new classes in order to make them more comparable. A simple way to do this is to add a normalization layer to the current deep model and we provide results with such a baseline (FT^{L2}) in Section 4.

$ScaIL$ attempts to approximate full learning by exploiting past classifiers as learned in their initial state, with all images available. Since the deep models evolve during the incremental process, a transformation of the initial classifiers is needed for them to be usable in the current incremental state. The method is illustrated in Figure 1.

The main differences with existing IL algorithms which exploit a bounded memory are: (1) the introduction of a second memory $\mathcal{I}$ to store initial past class classifiers and (2) the ablation of the distillation loss. Note that the size of $\mathcal{I}$ is orders of magnitude smaller than that of $\mathcal{B}$ since it only stores hundreds of floating point values per class instead of exemplar images. The immediate advantage of the method is that initial classifiers of past data are learned with all data. Initial classifiers learned with all images are stronger than the past classifiers learned only with exemplars in the current state. This is clearly visible in Figure 2(b) from the comparison of past classifiers weights as learned in the current state (red) and the weights of the same classifiers learned in states $\mathcal{S}^0$ and $\mathcal{S}^1$ (black and green). We also note the activations of new classes become weaker as the incremental learning process advances. The new classes from state $\mathcal{S}^0$ (black) are the strongest, followed by new classes from $\mathcal{S}^1$ (green) and those from $\mathcal{S}^2$ (blue).

The main challenge associated to $ScaIL$ is to combine classifiers originating from deep models learned in different IL states. The reuse of initial classifiers in later incremental states is made possible by fine tuning process with a

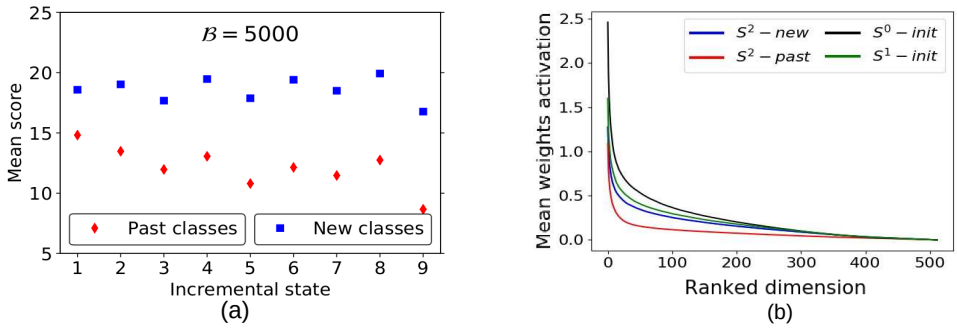

Figure 2: (a) - Raw prediction scores (before softmax) of vanilla fine tuning for the ILSVRC dataset with $\mathcal{B} = 5000$ past exemplars and a total of $\mathcal{Z} = 10$ states. Incremental states from 1 to 9 are represented. The initial state $\mathcal{S}^0$ is non-incremental and is not shown. (b) - Ranked mean weight activations of new and past classes in state $\mathcal{S}^2$ (blue and red) and mean weight activations of $\mathcal{S}^2$ past classes as initially learned in $\mathcal{S}^0$ (black) and $\mathcal{S}^1$ (green). *Best viewed in color.*

memory of the past. This process results in a partial preservation of the feature space even if the deep model evolves. In the supplementary material, we show that classifier reuse across states is impossible in absence of memory during IL model updates. $ScaIL$ reshapes initial classifiers from $\mathcal{I}$ in order to make them comparable to those of newly learned classes in the feature space defined by the current state's deep model. The scaling is based on weights statistics computed for initial models in each incremental state (Equation 1). Before computing the means in the equation, the weights of each initial classifier are ranked by their absolute value. The use of absolute values is necessary since classifier weights activations can be positive or negative.

$$\mu_i^{rank} = \frac{1}{P_i} \times \sum_{j=1}^{P_i} |w^{rank}(C_i^j)| \tag{1}$$

μ_i^{rank} is the mean of the weights ranked $rank$, with $1 \leq rank \leq D$, for the P_i classes initially learned in each past state $\mathcal{S}^i$, with $0 \leq i < k$. Figure 2(b) shows that classifiers of each past state have different statistical distributions. To make class predictions from different states comparable, it is necessary to compute μ_i^{rank} separately for each state. If $k = i$, we compute μ_k^{rank}, the mean of classifier weights for new classes from the current state $\mathcal{S}^k$, which is also their initial state. Note that each mean is computed using weights situated at the same rank for each classifier. For instance, μ_k^1 and μ_k^D will aggregate respectively the maximum and minimum weights of newly learned classes in $\mathcal{S}^k$.

$ScaIL$ transforms the past classifier weights as learned in their initial state using Equation 2. $w_{sc}^h(C_{sc}^j)$ is the scaled version of $w^h(C_i^j)$, the h^{th} dimension of the initial classifier C_i^j of the j^{th} past class. These weights are scaled using the ratio between the mean activation of new classes and that of past classes in their initial state. In Equation 2, each weight w^h is scaled using the mean activations of its corresponding rank, returned by function $r(\cdot)$, in the current and initial states $\mathcal{S}^k$ and $\mathcal{S}^i$. For instance, if the first weight ($h = 1$) of the classifier C_i^j is ranked 9^{th}, it will be scaled using the mean activations to the ninth dimension of the mean ranked activations μ_k^9 and μ_i^9 respectively. This is done in order to preserve the relative importance of each classifier weight. Figure 2(b) shows that $\mu_i^{rank} > \mu_k^{rank}$ for a given rank $rank$. Consequently, $ScaIL$ scaling reduces the weights of the j^{th} class learned in its initial state to make it more comparable to classifiers of new classes from the current state. The scaled classifier for each past class of the current state $\mathcal{S}^k$ is written as $C_{sc}^j = \{w_{sc}^1(C_{sc}^j), w_{sc}^2(C_{sc}^j), ..., w_{sc}^D(C_{sc}^j)\}$. The $ScaIL$ classification layer for $\mathcal{S}^k$ combines scaled classifiers for past classes and original classifiers for new classes. It can be written as $\mathcal{C}_{sc}^{N_k} = \{C_{sc}^1, ..., C_{sc}^{N_{k-1}}, C_k^{N_{k-1}+1}, ..., C_k^{N_k}\}$. The features learned in $\mathcal{S}^k$ are fed into this scaled classification layer instead of the original one provided by $\mathcal{M}^k$.

$$w_{sc}^h(C_{sc}^j) = \frac{\mu_k^{r(h)}}{\mu_i^{r(h)}} \times w^h(C_i^j) \tag{2}$$

Note that only scores of the top-10 past classes are scaled as they code more information, the scores of the remaining past classes are set to zero. The choice of this value is experimental.

We illustrate the effect of $ScaIL$ on the prediction scores in Figure 3. Past classes have a slightly larger mean classification score in the first states and a lower one in subsequent states. While not completely aligned, the predictions of past and new classes in $ScaIL$ are much more balanced compared to those of raw fine tuning results from Figure 2(a).

4. Experiments

4.1. Datasets

Experiments are done with four public datasets. This evaluation is more comprehensive than the usual one pro-

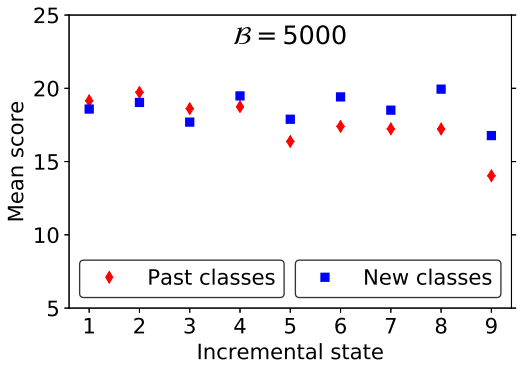

Figure 3: Prediction scores after scaling for the ILSVRC dataset [27] with $\mathcal{B} = 5000$ exemplars and $\mathcal{Z} = 10$ states.

posed in [5, 26], which includes two datasets only. We use:

- ILSVRC [27]: a subset of ImageNet [6] designed for object recognition and used in the popular ImageNet LSVRC challenges. For comparability, we retain the same configuration (order of classes and train/test splits) as [26]. This version of ILSVRC contains 1000 classes, with 1,23 million training and 50,000 test images.
- VGGFace2 [4]: face recognition dataset including over 9000 identities in its full version. Due to the heavy computation associated to IL, we select the 1000 identities with the largest number of training images. The resulting dataset has 491,746 training and 50,000 test images. Face detection was done using MTCNN [33] which was applied to each image prior to training and test phases.
- Google Landmarks [21] (Landmarks below): landmark recognition dataset whose full version includes over 30000 classes. We select the top 1000 classes and experiment with 374,367 training and 20,000 test images.
- CIFAR-100 [14]: object recognition dataset including 100 classes, with 500 training and 100 test images each.

4.2. Methodology and Baselines

The experimental setup used here is inspired from [5, 26]. The size of bounded memory $\mathcal{B}$ and the number of incremental states $\mathcal{Z}$ are the two most important parameters in IL with memory. We use three different values for each of them while fixing the value of the other parameter.

Memory Management. $\mathcal{B}$ size is varied to evaluate the robustness of the tested methods with memory availability. We fix the number of states $\mathcal{Z} = 10$ and run experiments with a memory which amounts to approximately $2\%, 1\%, 0.5\%$ of the full training sets. Memory sizes are thus $\mathcal{B} = \{20000, 10000, 5000\}$ for ILSVRC, $\mathcal{B} = \{10000, 5000, 2500\}$ for VGGFace2, $\mathcal{B} = \{8000, 4000, 2000\}$ for Landmarks and $\mathcal{B} = \{1000, 500, 250\}$ for CIFAR-100. Whenever a new incremental state is added, memory is updated by inserting exemplars of new classes and reducing exemplars of past classes in order to fit the maximum size.

Incremental States. The number of incremental states is the second key component of IL with memory and we evaluate its variation. We fix the memory to $\mathcal{B} = 0.5\%$ and test with $\mathcal{Z} = \{20, 50\}$ in addition to $\mathcal{Z} = 10$. The lowest memory size was selected since it is the most interesting configuration when memory budget is smallest.

Exemplar selection. A herding mechanism [31], called Nearest-Exemplars-Mean (NEM) was introduced in $iCaRL$ for exemplar selection [26]. BiC uses the same herding mechanism. For this, we provide results for $ScaIL$ with and without herding. $ScaIL^{herd}$ is directly comparable with $iCaRL$ and BiC.

Evaluation measures. To facilitate comparability, each configuration is evaluated with the widely used top-5 accuracy [27]. Each algorithm is tested in a large number of configurations and it is important to propose a summarized performance score. Inspired by works such as [25, 29], we introduce a global score computed with Equation 3. G_{IL} measures the performance gap between each algorithm and an upper bound method. This upper-bound is represented by $Full$ a non-incremental learning with all data available.

$$G_{IL} = \frac{1}{T} \times \sum_{t=1}^{T} \frac{acc_t - acc_{Full}}{acc_{Max} - acc_{Full}} \tag{3}$$

where: T - number of tested configurations; acc_t - top-5 score for each configuration (individual values of each row of Table 1); acc_{Full} - the upper-bound accuracy of the dataset ($Full$ in Table 1); acc_{Max} - the maximum theoretical value obtainable for the measure ($acc_{Max} = 100$ here).

G_{IL} estimates the average behavior of each algorithm with respect to the upper bound. The denominator is introduced to avoid a disproportionate influence of individual datasets in the aggregate score. G_{IL} is necessarily a negative number and the closer its value to zero, the better the method is. An ideal method, which reaches the upper bound value in all configurations, gives $G_{IL} = 0$. More details about G_{IL} are discussed in the supplementary material.

Baselines. Experiments have been conducted with strong baselines which are either inspired from existing IL literature or introduced here because relevant to $ScaIL$:

- $iCaRL$ [26] - method using fine tuning with classification and distillation losses to prevent catastrophic forgetting and NEM classification to counter data imbalance.
- BiC [32] - introduces a bias correction layer to address the imbalance responsible for catastrophic forgetting.
- $DeeSIL$ [3] - fixed representation IL method which freezes the network after the initial non-incremental state

and trains an SVM per class. While similar to the fixed representation from [26], an important difference concerns the fact that in [26] past classifiers are retrained only with exemplars in each IL state. Instead, as allowed by the frozen network, SVMs are trained in the initial state of each class and then reused.

- FT - vanilla fine tuning. Unlike existing IL algorithms which use distillation [5, 7, 10, 12, 22, 34], only classification loss is used. States are initialized with weights from previous model and all the network layers are allowed to evolve. Exemplars are selected randomly. FT is the backbone for all subsequent baselines and for $ScaIL$.
- FT^{NEM} - version of FT in which the classification is done using exemplars instead of the outputs of the deep model. FT^{NEM} is equivalent to a version of $iCaRL$ in which the distillation loss component is ablated.
- FT^{BAL} - FT followed by a balanced fine tuning proposed by [5] to reduce the effect of imbalance. FT^{BAL} is equivalent to a version of end-to-end IL [5] in which the distillation loss component is ablated.
- FT^{L2} - adds an L2-normalization layer to the raw classifier weights $\mathcal{C}^{N_k}$ given by model $\mathcal{M}^k$ to reduce bias in favor of new classes in current state $\mathcal{S}^k$.
- FT_{init} - the initial classifiers C_i^j of each past class replace the classifiers learned only with the past classes exemplars in $\mathcal{S}^k$. No transformation is applied to C_i^j. This is an ablation of the mean-related statistics from $ScaIL$.
- FT_{init}^{L2} - version of FT_{init} in which all classifiers are $L2$-normalized to make them more comparable.

Implementation. ResNet-18 is used as backbone architecture for all methods. For $iCaRL$ and BiC, we use the public TensorFlow [1] implementations provided by authors with their hyperparameters. FT and its variants are implemented in PyTorch [23]. The choice of hyperparameters is largely inspired by the original paper of ResNet-18 [8] and by end-to-end incremental learning [5]. To discard a potential influence of the deep learning framework, we trained FT for one ILSVRC configuration with $\mathcal{Z} = 10$ and $\mathcal{B} = 0.5\%$ using Tensorflow. The obtained performance is similar to that reported with Pytorch. $DeeSIL$ SVMs are implemented using scikit-learn [24] and their parameters are optimized on the training data since it is hard to hold out validation data in IL. More implementation details are provided in the supplementary material.

4.3. Discussion of results

Confirming the conclusions of [26], $iCaRL$ has the best overall performance for CIFAR-100 in Table 1. For the three larger datasets, the FT consistently outperforms $iCaRL$. Overall, FT more than halves the gap with $Full$ compared to $iCaRL$ ($G_{IL} = -6.40$ vs. $G_{IL} = -16.75$). The comparison to end-to-end IL [5], which achieves 69.4% top-5 accuracy for ILSVRC with $\mathcal{B} = 2\%$ is equally favorable to FT[1]. Since one important difference between FT and existing IL methods is the use of distillation, we analyze its role separately in Subsection 4.4.

The FT-based methods all have a positive contribution. FT^{NEM} and FT^{BAL} which are inspired by $iCaRL$ [26] and end-to-end IL [5] improve over FT by less than 0.5 G_{IL} points. FT^{L2}, the L2-normalized version of the classifiers from the current IL state, provides a gain of 1.23 G_{IL} points compared to FT. Somewhat surprisingly, the direct concatenation of initial classifier weights from different states in FT_{init} also improves performance over FT by over 1 point. However, its performance for individual configurations is much more contrasted than that of FT^{L2}. FT_{init} has low results for the two object recognition datasets, which are on average more difficult than face and landmark recognition tasks. FT_{init}^{L2} adds $L2$-normalization to FT_{init} classifiers and ranks fourth among all methods tested, with 1.73 G_{IL} improvement over FT. The best overall result is obtained with $ScaIL^{herd}$, which improves FT performance by 2.69 points. The difference between $ScaIL$ and FT_{init}^{L2} in terms of G_{IL} is not large but still interesting. $ScaIL$ has the most stable behavior among all those tested. In fact, its performance on the three large datasets is most interesting for the smallest $\mathcal{B}$ values. This is the most challenging case and also the most interesting in practice since it requires a reduced memory for past data. The increase of the number of incremental state results in a drop of performance for all methods. With equal memory $\mathcal{B}$, the worst results are obtained for $\mathcal{Z} = 50$ states, followed by $\mathcal{Z} = 20$ and $\mathcal{Z} = 10$. This finding confirms the results reported in [5] and [26]. It is probably an effect of a larger number of incremental rehearsal steps which are applied for larger $\mathcal{Z}$. Again, $ScaIL$ is the method which is the least affected by the change of the number of incremental states.

Contrarily to the conclusion of [5], the herding mechanism in $ScaIL^{herd}$ has positive effect compared to random selection of exemplars in $ScaIL$. Results show that, while BiC [32] is better for a lower number of incremental states ($\mathcal{Z} = 10$), $ScaIL$ has better behavior for a larger number of states. Equally important, $ScaIL$ performance is less affected by the reduction of the memory size and its performance is globally better for $B = 0.5\%$, this leads to a better G_{IL} score for $ScaIL$. Finally, the need of BiC for a validation set to parametrize the bias correction layer makes it nonfunctional if no memory of the past is available.

The performance gap between $Full$ learning and IL is

[1]Note that a complete set of results is not presented for end-to-end IL [5]. This method was not fully tested because we were not able to reproduce the results presented by the authors since the original implementation is based on Matlab, a non-free environment to which we don't have access.

States	$\mathcal{Z}=10$												$\mathcal{B}=0.5\%$								G_{IL}
Dataset	ILSVRC			VGGFace2			Landmarks			CIFAR-100			ILSVRC		VGGFace2		Landmarks		CIFAR-100		
$\mathcal{B}$	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	Z=20	Z=50	Z=20	Z=50	Z=20	Z=50	Z=20	Z=50	
$iCaRL^{herd}$	62.5	61.4	60.9	83.9	81.4	78.2	82.5	80.5	76.2	85.1	83.7	83.2	56.2	42.9	72.7	52.3	72.4	54.2	**73.2**	**55.7**	-16.75
BiC^{herd}	**85.5**	**82.8**	**79.7**	**97.3**	**96.6**	**95.7**	**97.9**	**97.3**	**96.6**	**88.8**	**87.6**	**83.5**	74.6	63.9	92.3	85.3	**94.7**	**90.5**	50.5	19.6	-4.03
$DeeSIL$	74.5	74.3	74.2	92.6	92.5	92.2	93.9	93.6	92.9	66.5	65.2	63.7	69.0	58.0	87.2	78.9	90.6	84.8	63.4	42.5	-7.10
FT	77.0	70.1	60.0	96.0	94.1	90.7	95.8	93.2	89.1	80.0	73.7	63.3	64.5	59.2	90.8	86.5	87.8	85.5	59.9	49.4	-6.40
FT^{NEM}	79.4	74.5	69.6	95.7	94.1	91.0	95.2	92.7	88.8	82.4	77.4	68.4	72.7	63.4	91.8	87.8	88.1	86.0	64.5	51.0	-6.01
FT^{BAL}	81.3	78.0	72.3	96.4	95.0	92.2	96.3	94.3	90.0	73.0	65.0	56.1	70.5	61.1	91.7	86.5	87.8	85.3	57.1	50.0	-5.98
FT^{L2}	81.4	77.6	72.1	96.5	95.1	92.4	96.2	94.4	91.4	81.8	77.2	69.1	73.4	66.7	92.8	88.8	89.8	87.1	63.2	49.9	-5.17
FT_{init}	68.9	66.5	61.2	95.9	95.3	94.5	96.5	95.0	92.7	79.3	77.3	73.7	53.4	39.0	**95.1**	90.3	90.6	87.5	60.7	40.1	-5.23
FT^{L2}_{init}	78.4	75.7	73.3	95.9	95.3	94.5	96.5	95.0	92.7	83.0	79.2	72.7	72.0	66.0	**95.1**	90.2	90.7	87.6	64.3	42.5	-4.67
$ScaIL$	81.0	78.2	75.1	96.4	95.6	94.5	96.9	95.3	92.7	84.6	81.1	74.9	73.9	68.3	94.5	90.5	90.7	88.2	67.9	47.7	-4.41
$ScaIL^{herd}$	82.0	79.8	76.6	96.5	95.8	95.2	97.3	96.0	94.0	85.6	83.2	79.1	**76.6**	**70.9**	95.0	**92.4**	92.6	90.4	69.8	51.0	**-3.71**
$Full$	92.3			99.2			99.1			91.2			92.3		99.2		99.1		91.2		-

Table 1: Top-5 average accuracy (%). Following [5], accuracy is averaged only for incremental states (i.e. excluding the initial, non-incremental state). The sizes of past memory $\mathcal{B}$ and number of IL states are varied to evaluate the robustness of algorithms. $Full$ is the non-incremental upper-bound performance obtained with all data available. The methods whose names include $herd$ exploit herding while the others are based on random exemplar selection. *Best results are in bold.*

naturally higher for more complex tasks, such as object recognition, compared to face and landmark recognition. For the last two tasks, classes have a more coherent visual representation and fewer examples are needed for a comprehensive representation of them. In the simplest configurations reported here ($\mathcal{Z}=10$, $\mathcal{B}=2\%$), the best IL algorithms are less than three points behind $Full$ for faces and landmarks. For such specialized tasks, incremental learning seems thus applicable in practice without a very significant performance loss. The situation is different for more complex tasks, such as object recognition, where significant progress is needed before IL algorithms approach the performance of classical learning.

An additional result concerns $DeeSIL$, the fixed representation method. Here, it is globally better than $iCaRL$, a finding which is at odds with the results originally reported in [26]. The difference is explained by the use of all data for each class, while past class training was unnecessarily restricted to $\mathcal{B}$ exemplars in [26]. FT outperforms $DeeSIL$ by less than 1 G_{IL} point. For $\mathcal{Z}=10$, $DeeSIL$ has very low dependence on the bounded memory size and could be also used in absence of past exemplars memory. Naturally, its performance drops for larger $\mathcal{Z}$ values because the initial model is learned with fewer classes but remains interesting.

4.4. Effect of distillation in IL

The use of knowledge distillation in incremental learning with bounded memory was pioneered in iCaRL [26], which extends the work on IL without memory from [8]. Distillation was largely adopted afterwards [5, 7, 10, 12, 22, 34] as a way to reduce the effect of catastrophic forgetting. This adoption was based on one experiment presented in [26] which compared the performance of iCaRL and fine tuning only on the CIFAR-100 dataset and with a single memory size. In Table 1, we report a similar finding for this dataset. For CIFAR-100, FT is probably less effective because it uses hard targets for loss minimization. These targets encode very sparse information for the small dataset available. In contrast, distillation exploits soft targets which encode more information [9] and is thus more fitted to work with small datasets. The results for $\mathcal{Z}=10$ states with different values of $\mathcal{B}$ support the above observation since the difference in favor of $iCaRL$ grows as $\mathcal{B}$ is reduced.

However, distillation hurts performance for all configurations tested for the three larger datasets, where FT has consequently better performance than $iCaRL$. The use of network outputs as soft targets for distillation was noted to produce a classification bias for past classes both in the original knowledge distillation paper [9] and in an incremental context [10]. A common assumption of distillation-based IL algorithms, first made in [8], is that the process starts with a powerful pretrained model which is trained on a large and balanced dataset. Under this condition, the soft targets used by the distillation loss are efficient to transfer knowledge to the next incremental state. Our hypothesis is that distillation tends to reinforce the errors due to data imbalance in the previous incremental state. In practice, if the distillation component is fed with soft targets whose predictions are wrong, it will push the classifier toward wrong classes. To verify this hypothesis, we present an analysis of correct and erroneous predictions for past and new classes in Table 2 for vanilla fine tuning (FT) and fine tuning with distillation used as backbone in $iCaRL$ ($FT^{distill}$). Results are shown only for ILSVRC with $\mathcal{Z}=10$ states and $\mathcal{B}=5000$ exemplars but trends are similar for other configurations. The bias toward new classes, expressed by $e(p,n)$ errors is similar with and without distillation. The correct predictions for new classes are also in a comparable range, although lower for $FT^{distill}$. This indicates that the data imbalance toward new classes has rather comparable effect

		Incremental states								
		$\mathcal{S}^1$	$\mathcal{S}^2$	$\mathcal{S}^3$	$\mathcal{S}^4$	$\mathcal{S}^5$	$\mathcal{S}^6$	$\mathcal{S}^7$	$\mathcal{S}^8$	$\mathcal{S}^9$
FT	$c(p)$	2117	2995	3415	3875	3653	4451	4558	5003	3119
	$e(p,p)$	156	450	807	1363	1842	2710	2626	3932	2388
	$e(p,n)$	2727	6555	10778	14762	19505	22839	27816	31065	39493
	$c(n)$	4151	4322	4103	4141	4267	4304	4247	4378	4248
	$e(n,n)$	809	638	875	828	716	674	743	595	741
	$e(n,p)$	40	40	22	31	17	22	10	27	11
$FT^{distill}$	$c(p)$	850	1008	1355	1355	1195	1344	1419	1543	1562
	$e(p,p)$	472	1746	3700	4999	6904	8246	10771	13400	14556
	$e(p,n)$	3678	7246	9945	13646	16901	20410	22810	25057	28882
	$c(n)$	3645	3834	3597	3607	3744	3754	3605	3766	3662
	$e(n,n)$	1043	793	928	905	785	776	828	692	751
	$e(n,p)$	312	373	475	488	471	470	567	542	587

Table 2: Top-1 ILSVRC correct and wrong classifications for vanilla fine tuning (FT), fine tuning with distillation ($FT^{distill}$) with $\mathcal{Z} = 10$ and $\mathcal{B} = 5000$. p and n stand for past and new classes. c and e indicate correct and erroneous classifications. $e(p,p)$ is to be read as past class examples wrongly predicted as other past classes. $e(p,n)$ is to be read as past class examples wrongly predicted as new classes. Note that top-1 performance is used because the proposed analysis is impossible for top-5 accuracy.

regardless of the use of distillation. The performance difference between the two methods is due mainly to confusions between past classes expressed by $e(p,p)$. They are roughly three times more frequent for $FT^{distill}$ compared to FT in Table 2. Equally important, while distillation is supposed to preserve accuracy for past classes, it clearly does not since the amount of correctly recognized past examples grows very steadily in $FT^{distill}$.

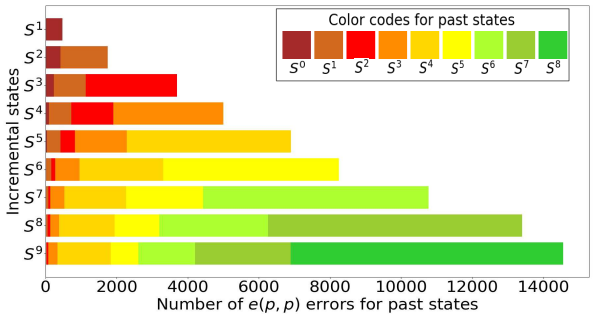

Figure 4: Detail of past-past errors $e(p,p)$ for individual states of $FT^{distill}$ on ILSVRC with $\mathcal{Z} = 10$ and $\mathcal{B} = 5000$. We note that, in each state, a majority of errors are due to the latest past state as a result of learning its associated state with an imbalanced training set. *Best viewed in color.*

In Figure 4, we present the distribution of $e(p,p)$ errors among individual past states for $FT^{distill}$. Since test data is balanced among states, the distribution of errors should also be approximately so. Instead, Figure 4 shows that a majority of past test data for state $\mathcal{S}^k$ are predicted as belonging to classes which were new when first learned in $\mathcal{S}^{k-1}$. This result confirms that class imbalance has an important role for the distillation component of the loss, similarly to its influence on the classification component. It is also noticeable that, except for $\mathcal{S}^5$, the number of error grows for more recent past states. Along with imbalance, the number of rehearsals after the initial learning of the class also plays an important role in terms of distillation-related errors.

Our findings indicate that vanilla fine tuning is preferable to distillation-based fine tuning as backbone for large scale IL with memory. Further distillation related experiments are presented in the supplementary material.

5. Conclusion

We introduced $ScaIL$, a simple but effective IL algorithm which combines classifiers learned in different IL states to reduce catastrophic forgetting. It keeps the number of parameters of the network constant across IL states and requires a second memory whose size is negligible compared to that of the exemplars memory. The method is compared to strong state-of-the-art methods, with their improvements based on distillation ablation and with new baselines which exploit initial classifiers. $ScaIL$ provides performance improvement over published results and is also better than the new baselines. Our method is also the most stable over the different memory and IL states values tested.

A consequent part of the performance improvement is due to the ablation of the distillation in IL algorithms. While widely used, we find that distillation is only useful for small scale datasets. Our analysis indicates that a performance drop appears for large scale datasets with memory when distillation is used. The drop is notably due to the inherently imbalanced character of datasets available in IL.

Comprehensive experiments were run on four public visual datasets with three memory sizes and three numbers of incremental states. We introduced an aggregated score to get an overview of performance in the different configurations tested. This experimental protocol can be reused to validate future works. To facilitate reproducibility, the code and dataset details are publicly available at: `https://github.com/EdenBelouadah/class-incremental-learning`.

The presented results reduce the performance gap between IL algorithms and non-incremental learning but the difference is still important, particularly for harder tasks. Class IL with bounded memory remains an open problem and new research is needed to make it usable in practice without significant performance loss. Future work will aim to: (1) improve vanilla FT while keeping model complexity and memory budget bounded, (2) explore new ways to handle data imbalance and (3) tackle real life situations where streamed data are partially or completely unlabeled.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang. Tensorflow: A system for large-scale machine learning. *CoRR*, abs/1605.08695, 2016.

[2] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[3] E. Belouadah and A. Popescu. Deesil: Deep-shallow incremental learning. In *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part II*, pages 151–157, 2018.

[4] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, Xi'an, China, May 15-19, 2018*, pages 67–74, 2018.

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, pages 241–257, 2018.

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. 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*, pages 248–255, 2009.

[7] C. He, R. Wang, S. Shan, and X. Chen. Exemplar-supported generative reproduction for class incremental learning. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 98, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[9] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[10] K. Javed and F. Shafait. Revisiting distillation and incremental classifier learning. *CoRR*, abs/1807.02802, 2018.

[11] R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[12] D. Kim, J. Bae, Y. Jo, and J. Choi. Incremental learning with maximum entropy regularization: Rethinking forgetting and intransigence. *CoRR*, abs/1902.00829, 2019.

[13] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? *CoRR*, abs/1805.08974, 2018.

[14] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[15] I. Kuzborskij, F. Orabona, and B. Caputo. From N to N+1: multiclass transfer incremental learning. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 3358–3365, 2013.

[16] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2018.

[17] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007, 2017.

[18] A. Mallya and S. 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*, pages 7765–7773, 2018.

[19] M. Mccloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[20] T. Mensink, J. J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2624–2637, 2013.

[21] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pages 3476–3485. IEEE Computer Society, 2017.

[22] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569, 2018.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems Workshops*, NIPS-W, 2017.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[25] S. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516, 2017.

[26] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[28] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[29] Y. Tamaazousti, H. Le Borgne, C. Hudelot, M. E. A. Seddik, and M. Tamaazousti. Learning more universal representations for transfer-learning. *arXiv:1712.09708*, 2017.

[30] Y. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[31] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 1121–1128, 2009.

[32] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 374–382, 2019.

[33] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process. Lett.*, 23(10):1499–1503, 2016.

[34] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. S. Davis. M2KD: multi-model and multi-level knowledge distillation for incremental learning. *CoRR*, abs/1904.01769, 2019.