FFNB: Forgetting-Free Neural Blocks for Deep Continual Learning

Hichem Sahbi
hichem.sahbi@sorbonne-universite.fr
Haoming Zhan
haoming.zhan@sorbonne-universite.fr

Sorbonne University, CNRS, LIP6
F-75005, Paris
France

Abstract

Deep neural networks (DNNs) have recently achieved a great success in computer vision and several related fields. Despite such progress, current neural architectures still suffer from catastrophic interference (a.k.a. forgetting) which obstructs DNNs to learn continually. While several state-of-the-art methods have been proposed to mitigate forgetting, these existing solutions are either highly rigid (as regularization) or time/memory demanding (as replay). An intermediate class of methods, based on dynamic networks, has been proposed in the literature and provides a reasonable balance between task memorization and computational footprint.

In this paper, we devise a dynamic network architecture for continual learning based on a novel forgetting-free neural block (FFNB). Training FFNB features on new tasks is achieved using a novel procedure that constrains the underlying parameters in the null-space of the previous tasks, while training classifier parameters equates to Fisher discriminant analysis. The latter provides an effective incremental process which is also optimal from a Bayesian perspective. The trained features and classifiers are further enhanced using an incremental "end-to-end" fine-tuning. Extensive experiments, conducted on different challenging classification problems, show the high effectiveness of the proposed method.

1 Introduction

Deep learning is currently witnessing a major success in different computer vision tasks including image and video classification [15]. The purpose of deep learning is to train convolutional or recurrent neural networks that map raw data into suitable representations prior to their classification [30, 31]. However, the success of these networks is highly dependent on the availability of large collections of labeled training data that capture the distribution of the learned categories. In many practical scenarios, mainly those involving streams of data, large collections *covering the inherent variability of the learned categories* are neither available nor can be holistically processed. Hence, training deep networks should be achieved as a part of a *lifelong* process, a.k.a. continual or incremental learning [37, 43, 49, 50].

The traditional mainstream design of deep networks is based on back propagation and stochastic gradient descent. The latter collects gradients through mini-batches and updates network parameters in order to learn different categories. However, in lifelong learning,

tasks involve only parts of data/categories, and this potentially leads to catastrophic forgetting (CF) defined as the inability of a learning model to "memorize" previous tasks when handling new ones. In cognitive science, CF is considered as an extreme case of the stability-plasticity dilemma [1, 32] where excessive plasticity causes an easy fit to new knowledge and less on previous ones. This is also related to concept (or distribution) drift [9] that may happen when a learning model keeps ingesting data [14, 44].

Whereas in most of the learning models (especially shallow ones [6, 34, 41, 51, 52]), CF could be overcome, its handling in deep networks is still a major challenge and existing solutions can only mitigate its effect. Indeed, CF results from the high non-linearity and entanglement of gradients when achieving back-propagation in deep networks (in contrast to shallow ones). Existing straightforward solutions bypass this effect by storing huge collections of data and *replaying* the learning process using all these collections; whereas *replay* is highly effective, it is known to be time and memory demanding and may result into resource saturation even on sophisticated hardware devices. Other solutions, with less time and memory footprint (e.g., regularization) can only mitigate the effect of CF. Another category of methods, based on dynamic networks provides a suitable balance between resource consumption and task memorization, and gathers the advantage of the two aforementioned categories of methods (namely replay and regularization) while discarding their inconveniences at some extent. Our proposed solution, in this work, is also built upon dynamic networks and allows mitigating CF with a reasonable growth in the number of training parameters.

2 Related work

Early work in continual learning mitigates CF by constraining model parameters to keep previous knowledge while learning new tasks. These techniques include regularization, replay and dynamic networks. Elastic weight consolidation [21] is one of the early regularization methods based on the Fisher information (see also [27, 38]). Other criteria, including synaptic intelligence [59], regularize parameters according to their impact on the training loss using gating mechanisms [46] and weight pruning [29]. Knowledge distillation [11] has also been investigated to build cumulative networks that merge previous tasks with current ones [26]. These methods include Incremental Moment Matching [23] which merges networks by minimizing a weighted Kullback–Leibler divergence and Learning without Memorizing [4] that relies on attention mechanisms [45]. Gradient episodic memory [28] relies on a memory budget but proceeds differently by regularizing and projecting the gradient of the current task onto the gradients of the previous ones. A variant in [8] extends further this regularization by averaging gradients through all the visited tasks while the method in [2] combines limited representative memory with distillation. Other models seek to leverage extra knowledge including unlabeled data [22, 60] (which are independent from the targeted tasks) or biases (due to imbalanced distributions) between previous and current tasks to further enhance generalization [12, 53].

The second category of methods (namely replay) consists in leveraging original or generated pseudo data as exemplars for continual learning. ICaRL [36] is one of these methods which extracts image exemplars for each observed class depending on a predefined memory budget. Pseudo-replay models [19] including deep generative networks [10, 18, 20, 39, 47] have also been investigated in the literature as alternative solutions that prevent storing exemplars. The particular method in [24] combines an explicit and an implicit memory where the former captures features from observed tasks and the latter corresponds to a discrimina-

tor and a generator similarly to deep generative replay. Other continual learning approaches employ coresets [33] to characterize the key information from different tasks.

Closely related to our contribution, dynamic networks proceed by adapting the topology of the trained architectures either at a macroscopic or microscopic level [56]. Macroscopically, progressive networks [40] define parallel cascaded architectures where each sub-network characterizes a specific task. Each layer propagates its output *not only* in the same sub-network but also through all the sub-networks of the subsequent tasks. PathNet [7] extends the topology of progressive networks, using evolutionary algorithms, to learn new connections between previous and current tasks. Random path selection networks [55] push this concept further by learning potential skip-connections among parallel sub-networks using random search. Microscopically, existing methods dynamically expand networks using thresholds on loss functions over new tasks and retrain the selected weights to prevent semantic drift [57]. Reinforced continual learning [55] employs a controller to define a strategy that expands the architecture of a given network while the learn-to-grow model [25] relies on neural architecture search [62] to define optimal architectures on new tasks. Other models [5], inspired by the process of adult neurogenesis in the hippocampus, combine architecture expansion with pseudo-rehearsal using auto-encoders. Our contribution in this paper proceeds differently compared to the aforementioned related work: tasks are incrementally handled by learning the parameters of a particular sub-network (referred to as FFNB) *in the null-space* of the previous tasks leading to *stable representations* on previously visited categories and discriminating representations *both* on previous and current categories. A bound is also provided that models the loss due to CF; this bound vanishes under particular settings of the null-space, activations and weight decay regularization. All these statements are corroborated through extensive experiments on different classification problems.

3 Problem formulation

Considering $\mathcal{X}$ as the union of input data (images, etc.) and $\mathcal{Y}$ their class labels drawn from an existing but unknown probability distribution $P(X,Y)$. The general goal is to train a network $f : \mathcal{X} \rightarrow \mathcal{Y}$ that assigns a label $f(X)$ to a given sample X while minimizing a generalization risk $R(f) = P(f(X) \neq Y)$. The design of f is usually achieved by minimizing an objective function (or loss) on a *fixed* set $\mathcal{T} = \{(x_i, y_i)\}_i$ including all the training data and their labels; this scheme is known as multi-task learning [16, 17]. In contrast, we consider in this work a different setting which learns f incrementally, i.e., only a subset of $\mathcal{T}$ (denoted $\mathcal{T}_t$) is available at a given cycle t (see for instance algorithm 1 in supplementary material). In what follows $\mathcal{T}_t$ will be referred to as *task*.

Let $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$ be a collection of tasks; this set is not necessarily a partition of $\mathcal{T}$, and each $\mathcal{T}_t$ may include one or multiple classes. Learning f incrementally may lead to CF; the latter is defined as the inability of f to remember (or correctly classify) previously seen tasks either due to a *distribution-shift* (i.e., when tasks correspond to the same classes but drawn from different distributions) or to a *class-shift* (i.e., when tasks correspond to disjoint classes). We consider only the *latter* while the *former* (closely related to domain adaptation) is out of the scope of this paper.

3.1 Dynamic networks and catastrophic forgetting

Without a loss of generality, we consider T as an overestimated (maximum) number of tasks visited during a continual learning process. We also consider f as a convolutional

network whose fully connected (FC) layers are dynamically updated. This FC sub-network of f, also referred to as FFNB (see Table. 7), corresponds to feature maps and classification layers whose widths $\{d_\ell\}_\ell$ (or dimensionalities) are dynamically expanded as tasks evolve. This dimensionality expansion makes data, belonging to the current and the previous tasks, increasingly separable. Let $\mathbf{X}_t \in \mathbb{R}^{d_0 \times n_t}$ denote the data matrix of a given task t (with $n_t = |\mathcal{T}_t|$), the maps of these FC layers are recursively defined as $\psi_\ell(\mathbf{X}_t) = g\big(\mathbf{W}_\ell\ \psi_{\ell-1}(\mathbf{X}_t)\big)$ with $\psi_\ell(.) \in \mathbb{R}^{d_\ell}$, $\ell \in \{1,\ldots,L\}$ and $\psi_0(\mathbf{X}_t) = \mathbf{X}_t$. Here $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is a matrix of training parameters and g a nonlinear activation. Considering this dynamic network, a band of parameters is assigned to each new task and trained "end-to-end" by back-propagating the gradient of a *task-wise* loss. These parameters are afterwards updated while those assigned to the previous tasks remain unchanged; see again algorithm 1 in supp material.

As data belonging to the previous tasks are dismissed, this straightforward "end-to-end" learning of the current task's parameters relies only on data in $\mathcal{T}_t$, and thereby the underlying classifier may suffer from insufficient generalization. Moreover, as no updates are allowed on the parameters of $\mathcal{T}_1,\ldots,\mathcal{T}_{t-1}$, it follows that *neither* generalization *nor* CF are appropriately handled on the previous tasks; indeed, as the parameters $\{\mathbf{W}_{\ell,t}\}_\ell$ of a given task t evolve, there is no guarantee that the outputs $\{\psi_\ell\}_\ell$ remain unchanged on data of the previous tasks, leading to changes in the underlying classification output ψ_{L+1} and thereby to CF (see later experiments). One may consider the statistics of the previous tasks (e.g., means and covariances of the data across different FC layer maps) and discriminatively learn the parameters associated to the current task (see later section 3.2.3). Nevertheless, prior to this step, one should be cautious in the way learning is achieved with those statistics, as the latter should remain stable as tasks evolve. Indeed, even when those statistics are available, one cannot update the *parameters* of the previous tasks as the *latter* disrupt in turn those statistics and no data are available in order to re-estimate them on the previous tasks. Hence, before making updates on the network parameters, feature maps should be stabilized on the previous tasks as introduced subsequently.

3.2 Proposed method

We introduce in this section an alternative solution which sill relies on dynamic networks but considers different FC layers and training procedure. Our framework incrementally learns the parameters $\{\mathbf{W}_{\ell,t}\}_\ell$ of the current task $\mathcal{T}_t$ in the null-space of the previous tasks $\mathcal{T}_1,\ldots,\mathcal{T}_{t-1}$ while maintaining the dynamic outputs $\{\psi_\ell\}_\ell$ of all the FC/FFNB layers almost unchanged (or at least stable) on $\mathcal{T}_1,\ldots,\mathcal{T}_{t-1}$. This approach, as described subsequently, learns *new tasks* incrementally and mitigates CF on *previous ones* while maintaining high generalization *on both*.

3.2.1 FFNB features

Learning the parameters of the current task should guarantee: (i) the *consistency* of the network predictions w.r.t. the underlying ground-truth, and (ii) the *stability* of the feature maps of the FC layers on the previous tasks. The first constraint is implemented by minimizing a hinge loss criterion while the second one is guaranteed by constraining the parameters of a new task to lie in the null-space $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$ of previous tasks data; in this notation, $\mathcal{P} = \{1,\ldots,t-1\}$ and $\mathbf{X}_\mathcal{P}$ refers to the matrix of data in $\mathcal{T}_1,\ldots,\mathcal{T}_{t-1}$. As shown subsequently, stability is implemented by learning the parameters of a new task in a residual

subspace spanned by the axes of principal component analysis (PCA)[1] applied to $\psi_\ell(\mathbf{X}_\mathcal{P})$.

Let $\Phi^{\ell,t}$ be the matrix of eigenvectors (principal axes of PCA) associated to data in $\psi_\ell(\mathbf{X}_\mathcal{P})$; in what follows, unless stated otherwise, we write $\Phi^{\ell,t}$ simply as Φ. Assuming these data centered, the principal axes are obtained by diagonalizing a covariance matrix incrementally defined as $\sum_{r=1}^{t-2} \psi_\ell(\mathbf{X}_r)\psi_\ell(\mathbf{X}_r)^\top + \psi_\ell(\mathbf{X}_{t-1})\psi_\ell(\mathbf{X}_{t-1})^\top$. The eigenvectors $\{\Phi_d\}_d$ in Φ constitute an orthonormal basis sorted following a decreasing order of the underlying eigenvalues. Let p be the smallest number of dimensions which concentrate most of the statistical variance. The vector of parameters associated to the current task t in $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$ is

$$\mathbf{W}_{\ell,t} := \sum_{d=p+1}^{d_{\ell-1}} \alpha_{\ell,t}^d \Phi_d^\top, \tag{1}$$

and training the latter equates to optimizing $\alpha_{\ell,t} = (\alpha_{\ell,t}^{p+1}, \ldots, \alpha_{\ell,t}^{d_{\ell-1}})^\top$. Let E denote a loss function associated to our classification task; considering the aforementioned reparametrization of $\mathbf{W}_{\ell,t}$, the gradient of the loss is now updated using the chain rule as $\frac{\partial E}{\partial \alpha_{\ell,t}} = \frac{\partial E}{\partial \mathbf{W}_{\ell,t}} \frac{\partial \mathbf{W}_{\ell,t}}{\partial \alpha_{\ell,t}}$ being $\frac{\partial E}{\partial \mathbf{W}_{\ell,t}}$ the original gradient obtained using back-propagation as provided with standard deep learning frameworks (including PyTorch and TensorFlow) and $\frac{\partial \mathbf{W}_{\ell,t}}{\partial \alpha_{\ell,t}}$ being a Jacobian matrix. The latter — set with the $(d_{\ell-1}-p)$ residual PCA eigenvectors — is used to maintain $\mathbf{W}_{\ell,t}$ in the feasible set, i.e., $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$. Considering this update scheme, the following proposition shows the consistency of the training process when handling CF.

Proposition 1 *Let $g : \mathbb{R} \to \mathbb{R}$ be a L-Lipschitz continuous activation (with $L \leq 1$). Any η-step update of $\mathbf{W}_{\ell,t}$ in $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$ using (1) satisfies $\forall r \in \mathcal{P}$*

$$\left\|\psi_\ell^{\eta-1}(\mathbf{X}_r) - \psi_\ell^0(\mathbf{X}_r)\right\|_F^2 \leq B$$

$$\text{with} \qquad B = \sum_{\tau=1}^{\eta-1}\sum_{k=0}^{\ell-1} \left(\left\|\alpha_{\ell-k,t}^\tau\right\|_F^2 . \left\|\beta_{\ell-k-1,r}^\tau\right\|_F^2 + \left\|\alpha_{\ell-k,t}^{\tau-1}\right\|_F^2 . \left\|\beta_{\ell-k-1,r}^{\tau-1}\right\|_F^2\right) . \prod_{k'=0}^{k-1} \left\|\mathbf{W}_{\ell-k',\mathcal{P}}^\tau\right\|_F^2, \tag{2}$$

being $\psi_\ell^0(\mathbf{X}_r)$ (resp. $\psi_\ell^{\eta-1}(\mathbf{X}_r)$) the map before the start (resp. the end) of the iterative update (gradient descent on current task $\mathcal{T}_t$), $\beta_{\ell,r}^\tau$ the projection of $\psi_\ell^\tau(\mathbf{X}_r)$ onto $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$ at any iteration τ, $\{\mathbf{W}_{\ell,r}^\tau\}_\ell$ the network parameters at τ, and $\|.\|_F$ the Frobenius norm.

Details of the proof are omitted and can be found in the supplementary material. More importantly, the bound in Eq. 2 suggests that FFNB layers endowed with L-Lipschitzian activations (e.g., ReLU) and low statistical variance in $\mathcal{N}_S(\psi_\ell(\mathbf{X}_\mathcal{P}))$ make CF contained. Eq. 2 also suggests that one may use weight decay (on $\{\alpha_{\ell,t}\}$) to regularize the parameters $\{\mathbf{W}_{\ell,t}\}_{t,\ell}$ leading to a tighter bound B, and again contained CF. Note that Eq. 2 is an increasing function of ℓ, so shallow FC layers suffer less from CF compared to deeper ones. However, controlling the norm of $\{\mathbf{W}_{\ell,t}\}_{t,\ell}$ (and p in Eq. 1) effectively mitigates the effect of CF (due to the depth) and maintains generalization. As a result, all the statistics (mean and covariance matrices) used to estimate the eigenvectors of PCA and also to update the classifier parameters (in section 3.2.3) remain stable. In short, Eq. 1 provides an effective way to "memorize" the previous tasks[2].

[1] with the smallest statistical variance.

[2] Note that Eq. 1 leads to almost orthogonal parameters through successive tasks (with shared residual components), and this provides an effective way to leverage both "shared multi-task" and "complementary" informations despite learning incrementally.

3.2.2 Initialization

We introduce a suitable initialization of the feature map parameters $\{\alpha_{\ell,t}\}_{\ell,t}$ which turns out to be effective during optimization (fine-tuning). We cast the problem of setting $\{\alpha_{\ell,t}\}_{\ell,t}$ (or equivalently $\{\mathbf{W}_{\ell,t}\}_{\ell,t}$; see Eq. 1) as a solution of the following regression problem

$$\min_{\alpha_{\ell,t}} \frac{\gamma}{2}\big\|\alpha_{\ell,t}\big\|_F^2 \;+\; \frac{1}{2}\big\|\mathbf{C}_{\ell,t} - \alpha_{\ell,t}^\top\, \Phi^\top\; \psi_\ell(\mathbf{X}_t)\big\|_F^2, \tag{3}$$

here $\mathbf{C}_{\ell,t} \in \mathbb{R}^{d_\ell \times n}$ (with $n = \sum_{r\leq t} |\mathcal{T}_r|$) is a predefined coding matrix whose entries are set to $+1$ iff data belong to current task t and 0 otherwise. One may show that optimality conditions (related to the gradient of Eq. 3) lead to the following solution

$$\alpha_{\ell,t} = \big(\gamma \mathbf{I} + \Phi\; \psi_\ell(\mathbf{X}_t)\; \psi_\ell(\mathbf{X}_t)^\top\; \Phi^\top\big)^{-1} \Phi\; \psi_\ell(\mathbf{X}_t)\; \mathbf{C}_{\ell,t}^\top. \tag{4}$$

As the setting of $\alpha_{\ell,t}$ relies only on current (mono) task data, it is clearly sub-optimal and may affect the discrimination power of the learned feature maps. In contrast to the above initialization, we consider another (more effective multi-task) setting using all $\{\mathcal{T}_r\}_{r\in\mathcal{A}}$ (with $\mathcal{A} = \mathcal{P}\cup\{t\}$), and without storing all the underlying data. Similarly to Eq. (4), we derive

$$\alpha_{\ell,t} = \big(\gamma \mathbf{I} + \Phi \sum_{r\in\mathcal{A}} [\psi_\ell(\mathbf{X}_r)\; \psi_\ell(\mathbf{X}_r)^\top]\; \Phi^\top\big)^{-1}\; \Phi \sum_{r\in\mathcal{A}} [\psi_\ell(\mathbf{X}_r)\; \mathbf{C}_{\ell,r}^\top]. \tag{5}$$

This equation can still be evaluated incrementally (without forgetting) while leveraging multiple tasks $\mathcal{T}_1,\ldots,\mathcal{T}_t$, and without explicitly storing the whole data in $\{\psi_\ell(\mathbf{X}_r)\}_r$ and $\{\mathbf{C}_{\ell,r}\}_r$.

3.2.3 FFNB classifiers

The aforementioned scheme is applied in order to learn the parameters of the feature maps while those of the classifiers are designed differently. The setting of these parameters is based on Fisher discriminant analysis (FDA) which has the advantage of being achievable incrementally by storing only the means and the covariance matrices associated to each task. Given current and previous tasks (resp. $\mathcal{T}_t$ and $\mathcal{T}_r$), FDA approaches the problem by modeling the separable FFNB features (as designed earlier) as gaussians with means and covariances $(\mu_t^{L-1}, \Sigma_t^{L-1})$, $(\mu_r^{L-1}, \Sigma_r^{L-1})$ respectively. Following this assumption, the Bayes optimal decision function corresponds to the log likelihood ratio test. One may show that its underlying separating hyperplane $(\mathbf{W}_{L,(t,r)}, b)$ maximizes the following objective function

$$\max_{\mathbf{W}_{L,(t,r)}} \frac{\big(\mathbf{W}_{L,(t,r)}^\top \big(\mu_t^{L-1} - \mu_r^{L-1}\big)\big)^2}{\mathbf{W}_{L,(t,r)}^\top \big(\Sigma_t^{L-1} + \Sigma_r^{L-1}\big)\; \mathbf{W}_{L,(t,r)}}, \tag{6}$$

with its shrinkage estimator solution being

$$\begin{array}{lcl} \mathbf{W}_{L,(t,r)} & = & \big(\Sigma_t^{L-1} + \Sigma_r^{L-1} + \varepsilon \mathbf{I}\big)^{-1} \big(\mu_t^{L-1} - \mu_r^{L-1}\big) \\ b & = & -\mathbf{W}_{L,(t,r)}^\top \big(\mu_t^{L-1} + \mu_r^{L-1}\big). \end{array} \tag{7}$$

Under heteroscedasticity (i.e., $\Sigma_t^{L-1} \neq \Sigma_r^{L-1}$), these covariance matrices allow normalizing the scales of classes leading to better performances as shown later in experiments. Considering the pairwise class parameters $\{\mathbf{W}_{L,(t,r)}\}_{r\in\mathcal{P}}$, one may incrementally derive the underlying pairwise classifiers as

$$\psi_L^{(t,r)}(.) = \tanh\big(\mathbf{W}_{L,(t,r)}\; \psi_{L-1}(.)\big), \tag{8}$$

and the output of the final classifier $\psi^t_{L+1}(.)$ (see again Table. 7) is obtained by pooling all the pairwise scores $\{\psi^{(t,r)}_L(.)\}_{t,r}$ through $r \in \mathcal{P}$ resulting into

$$\psi^t_{L+1}(.) = \sum_{r \in \mathcal{P}} \psi^{(t,r)}_L(.). \tag{9}$$

As shown later in experiments (see supp material), these incremental aggregated "one-vs-one" classifiers outperform the usual "one-vs-all" softmax while mitigating CF. This also follows the learned separable FFNB features which make these classifiers highly effective.

3.2.4 "End-to-end" fine-tuning

End-to-end fine-tuning of the whole network involves only the parameters of the feature map and the classification layers associated to the current task, as the other parameters cannot be updated without knowing explicitly the data. Note that the convolutional layers are kept fixed: on the one hand, these layers capture low-level features, which are common to multiple tasks and can therefore be pre-trained offline. On the other hand, retraining the convolutional layers may disrupt the outputs of the feature maps and hence the classifiers. Details of the whole "end-to-end" incremental learning are described in algorithm 2 in supp material.

4 Experimental validation

We evaluate the performance of our continual learning framework on the challenging task of action recognition, using the SBU and FPHA datasets [8, 58]. SBU is an interaction dataset acquired (under relatively well controlled conditions) using the Microsoft Kinect sensor; it includes in total 282 moving skeleton sequences (performed by two interacting individuals) belonging to 8 categories. Each pair of interacting individuals corresponds to two 15 joint skeletons and each joint is encoded with a sequence of its 3D coordinates across video frames [31]. The FPHA dataset includes 1175 skeletons belonging to 45 action categories which are performed by 6 different individuals in 3 scenarios. Action categories are highly variable with inter and intra subject variability including style, speed, scale and viewpoint. Each skeleton includes 21 hand joints and each joint is again encoded with a sequence of its 3D coordinates across video frames [31]. In all these experiments, we use the same evaluation protocol as the one suggested in [8, 58] (i.e., train-test split[3]) and we report the average accuracy over all the *visited* classes of actions[4].

4.1 Setting and performances

The whole network architecture is composed of a *spatial graph convolutional* block similar to [24] appended to *FFNB*. The *former* includes an aggregation layer and a dot product layer while the *latter* consists of our feature map and classification layers [42]. During incremental learning, all the parameters are fixed excepting those of the current task (in FFNB) which are allowed to vary. For each task, we train the network parameters (FC layers) as described earlier for 250 epochs per task with a batch size equal to 50, a momentum of 0.9 and a learning rate (denoted as $\nu(t)$) inversely proportional to the speed of change of

[3] excepting that training data belonging to different classes are visited incrementally.

[4] Due to space limitation, extra experiments and comparisons can be found in the supplementary material.

PCA dim \ Tasks (1 class / $\mathcal{T}_t$)	$\mathcal{T}_1(1)$	$\mathcal{T}_2(2)$	$\mathcal{T}_3(3)$	$\mathcal{T}_4(4)$	$\mathcal{T}_5(5)$	$\mathcal{T}_6(6)$	$\mathcal{T}_7(7)$	$\mathcal{T}_8(8)$
$p=15$	100.00	100.00	96.42	97.36	95.34	87.50	84.48	81.53
$p=25$	100.00	100.00	100.00	97.36	95.34	85.41	82.75	83.07
$p=35$	100.00	100.00	96.42	97.36	95.34	85.41	82.75	81.53
$p=45$	100.00	100.00	100.00	100.00	97.67	89.58	89.65	**84.61**
$p=50$	100.00	100.00	96.42	97.36	93.02	85.41	81.03	73.84
$p=55$	100.00	100.00	100.00	97.36	90.69	87.50	75.86	69.23
Incremental (FFNB, algorithm 1)	100.00	50.00	28.57	26.31	18.60	16.66	13.79	12.30
Multi-task (all, upper bound)	—	—	—	—	—	—	—	**90.76**
Network size (# of param)	2165	2372	2619	2924	3305	3780	4367	5084
Average training time per epoch (in seconds)	0.1326	0.1369	0.1411	0.1475	0.1519	0.1651	0.1629	0.1737

Table 1: Impact of retained PCA dimension on the performances of incremental learning (the max nbr of dimensions is 60); for each column (task $\mathcal{T}_t$), performances are reported on the union of all the visited classes (i.e., $[1-t]$).

PCA dim \ Tasks (5 classes/$\mathcal{T}_t$)	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_6$	$\mathcal{T}_7$	$\mathcal{T}_8$	$\mathcal{T}_9$
	$[1-5]$	$[6-10]$	$[11-15]$	$[16-20]$	$[21-25]$	$[26-30]$	$[31-35]$	$[36-40]$	$[41-45]$
$p=15$	65.07	57.14	55.44	57.52	58.51	56.07	59.42	62.10	62.26
$p=30$	65.07	55.55	56.99	57.52	56.96	56.84	59.42	60.74	61.39
$p=45$	68.25	57.14	60.10	58.30	56.03	55.03	60.97	62.30	62.60
$p=60$	68.25	55.55	62.17	60.61	63.15	62.53	65.41	64.84	66.08
$p=75$	68.25	58.73	64.76	63.70	62.84	61.24	64.96	66.01	**67.47**
$p=90$	69.84	61.90	67.87	62.93	60.06	59.94	61.41	63.28	62.78
$p=105$	65.07	62.69	68.39	63.32	58.51	58.65	60.97	62.50	62.43
$p=120$	61.90	60.31	65.80	62.54	57.27	51.93	56.54	55.27	56.86
Incremental (FFNB, algo 1)	19.04	9.52	6.21	4.63	3.71	3.10	2.66	2.34	2.08
Multi-task (all, upper bound)	—	—	—	—	—	—	—	—	**84.17**
Network size (# of param)	5549	8784	14819	25904	44289	72224	111959	165744	235829
Avg. training time per epoch (in s)	2.0373	2.0241	2.0686	2.1463	2.2393	2.3318	2.3992	2.5082	2.5755

Table 2: Impact of retained PCA dimension on the performances of incremental learning (the max nbr of dimensions is 168); for each column (task $\mathcal{T}_t$), performances are reported on the union of all the visited classes (i.e., $[1-5t]$).

the current task loss; when this speed increases (resp. decreases), $\nu(t)$ decreases as $\nu(t) \leftarrow \nu(t-1) \times 0.99$ (resp. increases as $\nu(t) \leftarrow \nu(t-1)/0.99$). All these experiments are run on a GeForce GTX 1070 GPU device (with 8 GB memory) and no data augmentation is achieved.

Tables. 1 and 2 show the behavior of our continual learning model w.r.t. p the number of dimensions kept in PCA. From these results, it becomes clear that 45 dimensions on SBU (75 on FPHA) capture most of the statistical variance of the previous tasks ($\gg 95\%$ in practice) and enough dimensions are hence reserved to the current task. These dimensions make the learned FFNB stable on the previous tasks while also being effective on the current one. These tables also show a comparison of incremental and multi-task learning baselines (which learn the convolutional block and FFNB "end-to-end" using the whole ambient space). Note that multi-task learning performances are available only at the final task as this baseline requires all the tasks. From these tables, it is clear that the multi-task baseline obtains the best performance, however, our proposed method reaches a high accuracy as well in spite of being incremental while the second (incremental) baseline behaves almost as a random classifier. Tables 1, 2 also show network size and training time as tasks evolve while tables 8, 9, 10, 11 (in the supp material) show extra-tuning w.r.t. respectively the number of layers and *band-sizes*; the latter correspond to the *number of added neurons* per layer and per task.

4.2 Ablation study and comparison

We also study the impact of each component of our continual learning model on the performances when taken separately and jointly. From the results in tables 3 and 4, the use of the

Null-space	Heteroscedasticity	Init	$\mathcal{T}_1(1)$	$\mathcal{T}_2(2)$	$\mathcal{T}_3(3)$	$\mathcal{T}_4(4)$	$\mathcal{T}_5(5)$	$\mathcal{T}_6(6)$	$\mathcal{T}_7(7)$	$\mathcal{T}_8(8)$
✗	✗	rand	100.00	85.00	67.85	55.26	44.18	37.50	34.48	29.23
✗	✗	mono	100.00	90.00	53.57	36.84	27.90	22.91	25.86	26.15
✗	✗	multi	100.00	90.00	71.42	68.42	51.16	45.83	46.55	43.07
✗	✓	rand	100.00	100.00	96.42	86.84	51.16	54.16	63.79	56.92
✗	✓	mono	100.00	50.00	53.57	26.31	11.62	12.50	20.68	10.76
✗	✓	multi	100.00	100.00	100.00	47.36	53.48	39.58	44.82	44.61
✓	✗	rand	100.00	90.00	53.57	44.73	37.20	31.25	31.03	30.76
✓	✗	mono	100.00	95.00	92.85	94.73	81.39	68.75	60.34	50.76
✓	✗	multi	100.00	100.00	100.00	97.36	90.69	87.50	82.75	75.38
✓	✓	rand	100.00	50.00	57.14	50.00	25.58	29.16	34.48	30.76
✓	✓	mono	100.00	100.00	96.42	94.73	72.09	54.16	50.00	55.38
✓	✓	multi	100.00	100.00	100.00	**100.00**	**97.67**	**89.58**	**89.65**	**84.61**

Table 3: Ablation study (with pretraining and fine-tuning, here $p = 45$); for each column (task $\mathcal{T}_t$), performances are reported on the union of all the visited classes (i.e., $[1-t]$).

Null-space	Heteroscedast.	Multi	$\mathcal{T}_1$ $[1-5]$	$\mathcal{T}_2$ $[6-10]$	$\mathcal{T}_3$ $[11-15]$	$\mathcal{T}_4$ $[16-20]$	$\mathcal{T}_5$ $[21-25]$	$\mathcal{T}_6$ $[26-30]$	$\mathcal{T}_7$ $[31-35]$	$\mathcal{T}_8$ $[36-40]$	$\mathcal{T}_9$ $[41-45]$
✗	✗	✗	19.04	11.11	9.32	5.79	7.43	5.68	6.43	5.85	6.08
✗	✗	✓	71.42	65.07	59.06	59.84	62.84	60.72	59.86	58.20	56.52
✗	✓	✗	19.04	11.11	6.21	5.40	4.02	3.10	3.32	2.73	2.08
✗	✓	✓	69.84	56.34	54.92	54.44	57.89	60.46	64.07	64.84	65.39
✓	✗	✗	77.77	67.46	66.83	63.32	60.99	58.39	58.98	57.42	56.00
✓	✗	✓	76.19	67.46	64.76	62.93	62.84	61.24	59.42	58.00	54.95
✓	✓	✗	71.42	71.42	62.69	54.05	54.48	47.80	47.45	45.50	45.56
✓	✓	✓	68.25	58.73	64.76	63.70	62.84	61.24	64.96	66.01	**67.47**

Table 4: Ablation study (with pretraining and fine-tuning, here $p = 75$); for each column (task $\mathcal{T}_t$), performances are reported on the union of all the visited classes (i.e., $[1-5t]$).

null-space (in Eq. 1) provides a significant gain in performances and the impact of FDA covariance normalization (i.e., heteroscedasticity in Eq. 7) is also globally positive. This results from the learned features, in FFNB, which are designed to be *separable* without necessarily being *class-wise homogeneous*, and thereby their normalization provides an extra gain (as again shown through these performances)[5]. We also observe the positive impact of multi-task initialization w.r.t. mono-task and random initializations (Eq. 5 vs. 4). Finally, tables 12 and 13 (in supp material) show the impact of network pre-training and "end-to-end" fine-tuning on the performances (see again section 3.2.4). As observed, the best performances are obtained when both pre-training and fine-tuning are used.

4.3 Extra experiments: CIFAR100

We also evaluate the accuracy of our FFNB on the challenging CIFAR100 dataset which includes 60k images belonging to 100 categories; 50k images are used for training and 10k for testing. We use EfficientNet [48] as our CNN backbone. In all the results in table 5, "null-space + heteroscedasticity + multi-task initialization" settings are used, and the band-size is set to 1, number of layers in the feature map block to 2, size of minibatch to 32, the learning rate fixed to 10e-3 and neither weight decay nor momentum are used. Performances are measured using the standard "Average Incremental Acc" which is proposed in iCaRL [36], and defined as the average accuracy across all the visited tasks. Similarly to the standard evaluation protocol in iCaRL [36], the first 50 classes ([1-50]) are used to pretrain the

[5] It's worth noticing that our normalization is different from the usual batch-normalization, as the latter allows obtaining homogeneous features through neurons belonging to the same layer while our normalization makes features homogeneous through classes/tasks (see also empirical evidences in tables 22 and 23 of the supp material).

Tasks / Test classes	$\mathcal{T}_0$ [1 – 50]	$\mathcal{T}_1$ [51 – 55]	$\mathcal{T}_2$ [56 – 60]	$\mathcal{T}_3$ [61 – 65]	$\mathcal{T}_4$ [66 – 70]	$\mathcal{T}_5$ [71 – 75]	$\mathcal{T}_6$ [76 – 80]	$\mathcal{T}_7$ [81 – 85]	$\mathcal{T}_8$ [86 – 90]	$\mathcal{T}_9$ [91 – 95]	$\mathcal{T}_{10}$ [96 – 100]	*Average Incremental Acc.*
Top-50 classes	83.66	80.98	78.02	73.90	71.32	68.20	64.82	61.42	57.52	52.60	49.42	67.44
50 – 55		79.20	71.60	70.40	65.80	58.00	56.60	43.80	43.60	34.00	26.00	54.90
55 – 60			89.40	84.00	74.20	71.40	68.60	63.80	61.80	60.20	57.00	70.04
60 – 65				62.80	58.00	56.40	46.00	43.40	35.20	31.00	24.20	44.63
65 – 70					73.60	73.20	66.40	62.80	55.40	50.80	43.00	60.74
70 – 75						80.60	76.40	68.00	59.40	47.80	41.60	62.30
75 – 80	–	–					80.80	80.20	71.20	67.60	62.40	72.44
80 – 85			–	–				87.40	73.00	62.80	57.20	70.10
85 – 90					–	–			85.00	83.40	79.00	82.47
90 – 95							–	–		82.60	74.60	78.60
95 – 100									–	–	80.40	80.40
Average Task Acc.	83.66	80.82	78.43	73.55	70.34	68.11	65.19	62.56	58.88	55.06	51.98	68.05

Table 5: Results on *CIFAR100-B50-S10*. Here $B50$ stands for the 50 pretraining classes and $S10$ for tasks $\mathcal{T}_1,\ldots,\mathcal{T}_{10}$ which are learned incrementally (here $\mathcal{T}_1,\ldots,\mathcal{T}_{10}$ correspond to classes [51-100] while $\mathcal{T}_0$ is the pretraining task involving classes [1-50]). In this table, the symbol "–" stands for "accuracy not available" as classes are incrementally visited so training+test data, belonging to the subsequent tasks, are obviously not available beforehand.

Methods	#Params (M)	Avg. Acc.
Upper Bound [34]	11.2	79.91
iCaRL [34, 44]	11.2	58.59
UCIR [34, 44]	11.2	59.92
BiC [34, 44]	11.2	60.25
WA [34, 44]	11.2	57.86
PoDNet [34]	11.2	64.04 (63.19)
DDE (UCIR R20) [34]	11.2	62.36
DDE (PoDNet R20) [34]	11.2	64.12
DER(w/o P) [34]	67.2	72.81
DER(P) [34]	8.79	72.45
Ours	5.8	68.05

Table 6: Results on *CIFAR100-B50* (modified from Table 2 in DER [34] where numbers in blue refer to the results tested by the re-implementation in DER [34] and numbers in parentheses refer to the results reported in the original papers).

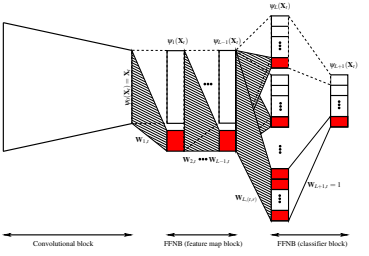

Table 7: This figure shows the whole architecture including a convolutional backbone and FFNB. For each new task $\mathcal{T}_t$ (one or multiple new classes), bands of neurons (shown in red) are appended to the dynamic layers $\psi_1(.),\ldots,\psi_{L-1}(.)$ of the feature map block, and only the underlying parameters (hatched) are trained in the null-space of the previous tasks. The classifier block includes two layers: in the first layer, bands of neurons (in red) are appended to $\psi_L(.)$ in order to model all the "one-vs-one" classifiers involving the new classes and all the (previously and newly) visited classes so far. Finally, a band of neurons (again in red) is appended to the second classification layer $\psi_{L+1}(.)$ in order to aggregate the scores of the "one-vs-one" classifiers (see again 3.2.3).

"EfficientNet" backbone, while the remaining 50 classes ([51-100]) are used for incremental task learning. Comparisons are shown in tables 5 and 6 w.r.t. different tasks and related work. These results show that our proposed method provides a reasonable balance between accuracy and the maximum number of training parameters w.r.t. these related methods.

5 Conclusion

We introduce in this work a novel continual learning approach based on dynamic networks. The strength of the proposed method resides in its ability to learn discriminating representations and classifiers incrementally while providing stable behaviors on the previous tasks. The proposed method is based on FFNBs whose parameters are learned in the null-space of the previous tasks, leading to stable representations and classifications on these tasks. Aggregated classifiers are also learned incrementally using Fisher discriminant analysis which also exhibits optimal behavior especially when the feature maps are appropriately learned and separable. Conducted experiments show the positive impact of each of the proposed components of our model. As a future work, we are currently investigating multiple aspects including replay-based methods that allow fine-tuning the networks using not only *current* task data but also *previous* ones sampled from a generative network.

References

[1] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[2] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[3] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[4] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019.

[5] Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Jonathan A Cox, Craig M Vineyard, Kristofor D Carlson, William M Severa, Conrad D James, and James B Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 526–533. IEEE, 2017.

[6] Anjan Dutta and Hichem Sahbi. High order stochastic graphlet embedding for graph-based pattern recognition. *arXiv preprint arXiv:1702.00156*, 2017.

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

[8] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. *In CVPR*, 2018.

[9] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (esann)*, 2016.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

[13] X. Hu, K. Tang, C. Miao, X-S. Hua, and H. Zhang. Distilling causal effect of data in class-incremental learning. In *arXiv preprint arXiv:2103.01737*, 2021.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Mingyuan Jiu and Hichem Sahbi. Deep kernel map networks for image annotation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1571–1575. IEEE, 2016.

[16] Mingyuan Jiu and Hichem Sahbi. Laplacian deep kernel learning for image annotation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1551–1555. IEEE, 2016.

[17] Mingyuan Jiu and Hichem Sahbi. Deep representation design from deep kernel networks. *Pattern Recognition*, 88:447–457, 2019.

[18] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.

[19] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[22] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2019.

[23] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017.

[24] Huaiyu Li, Weiming Dong, and Bao-Gang Hu. Incremental concept learning via online generative memory recall. *arXiv preprint arXiv:1907.02788*, 2019.

[25] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*, 2019.

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[27] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018.

[28] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[29] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[30] Ahmed Mazari and Hichem Sahbi. Deep temporal pyramid design for action recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2077–2081. IEEE, 2019.

[31] Ahmed Mazari and Hichem Sahbi. Mlgcn: Multi-laplacian graph convolutional networks for human action recognition. In *BMVC*, page 281, 2019.

[32] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.

[33] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[34] Quentin Oliveau and Hichem Sahbi. Learning attribute representations for remote sensing ship category classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(6):2830–2840, 2017.

[35] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems*, pages 12669–12679, 2019.

[36] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[37] Mark Bishop Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994.

[38] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748, 2018.

[39] Mohammad Rostami, Soheil Kolouri, James McClelland, and Praveen Pilly. Generative continual concept learning. *arXiv preprint arXiv:1906.03744*, 2019.

[40] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[41] Hichem Sahbi. A particular gaussian mixture model for clustering and its application to image retrieval. *Soft Computing*, 12(7):667–676, 2008.

[42] Hichem Sahbi. Learning laplacians in chebyshev graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2064–2075, 2021.

[43] Hichem Sahbi, Patrick Etyngier, Jean-Yves Audibert, and Renaud Keriven. Manifold learning using robust graph laplacian for interactive image search. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[44] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[45] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[46] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

[47] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[48] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *In International Conference on Machine Learning*, page 6105–6114, 2019.

[49] Sebastian Thrun. A lifelong learning perspective for mobile robot control. In *Intelligent Robots and Systems*, pages 201–214. Elsevier, 1995.

[50] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996.

[51] Ling Wang and Hichem Sahbi. Bags-of-daglets for action recognition. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1550–1554. IEEE, 2014.

[52] Ling Wang and Hichem Sahbi. Nonlinear cross-view sample enrichment for action recognition. In *European Conference on Computer Vision*, pages 47–62. Springer, 2014.

[53] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[55] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, pages 899–908, 2018.

[56] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. *arXiv preprint arXiv:2103.16788*, 2021.

[57] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[58] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras. Two-person interaction detection using body pose features and multiple instance learning. *In IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2012.

[59] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

[60] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. *arXiv preprint arXiv:1903.07864*, 2019.

[61] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S-T. Xia. Maintaining discrimination and fairness in class incremental learning. In *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 13208–13217, 2020.

[62] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.