Dynamic Support Network for Few-shot Class Incremental Learning

Boyu Yang, Mingbao Lin, Yunxiao Zhang, Binghao Liu, Xiaodan Liang, Rongrong Ji, *Senior Member, IEEE*, Qixiang Ye, *Senior Member, IEEE*.

Abstract—Few-shot class-incremental learning (FSCIL) is challenged by catastrophically forgetting old classes and over-fitting new classes. Revealed by our analyses, the problems are caused by feature distribution crumbling, which leads to class confusion when continuously embedding few samples to a fixed feature space. In this study, we propose a Dynamic Support Network (DSN), which refers to an adaptively updating network with compressive node expansion to "support" the feature space. In each training session, DSN tentatively expands network nodes to enlarge feature representation capacity for incremental classes. It then dynamically compresses the expanded network by node self-activation to pursue compact feature representation, which alleviates over-fitting. Simultaneously, DSN selectively recalls old class distributions during incremental learning to support feature distributions and avoid confusion between classes. DSN with compressive node expansion and class distribution recalling provides a systematic solution for the problems of catastrophic forgetting and overfitting. Experiments on CUB, CIFAR-100, and miniImage datasets show that DSN significantly improves upon the baseline approach, achieving new state-of-the-arts. The code is publicly available.

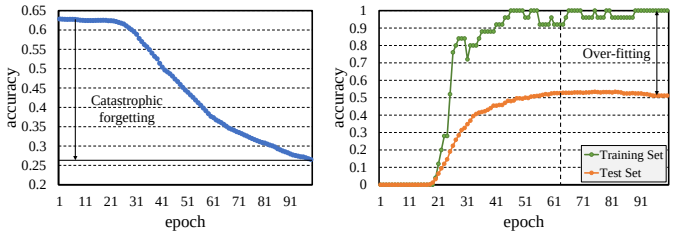
Index Terms—Few-shot Learning, Incremental Learning, Support Network, Compressive Network Expansion, Distribution Recalling.

1 Introduction

Great progress in visual recognition has been made over the past few years. This can be broadly attributed to advanced learning mechanisms and large-scale datasets with adequate supervision. However, machine learning mechanisms remain incomparable with cognitive learning, which not only constructs high-precision recognition capability upon few annotated samples but also can continually generalize such capability to novel things [1].

Few-shot class incremental learning (FSCIL) is an emerging machine learning paradigm inspired by cognitive learning [2]. Given base classes with sufficient training data and novel classes of few supervisions, FSCIL trains a representation model using old classes and then continually adapts the model to new classes. Both old classes and new classes need to be recognized during inference. However, FSCIL faces challenges which are beyond conventional learning paradigms. On the one hand, fine-tuning the model with new classes disturbs old-class feature distributions, which causes "catastrophic forgetting". On the other hand, training with only a few samples from new classes aggregates model bias toward old classes, which causes distribution crumbling and model over-fitting, Fig. 1.

Recent study managed to reduce model complexity and alleviated the over-fitting issue by squeezing network parameters [3]. Knowledge distillation [4] and evolving classifiers [5] facilitated memorizing old classes when generalizing models to new. The neural gas method [2] regularized feature space topology which in turn attenuated forgetting and resolved the problem of over-fitting, simultaneously. However, the problems caused by feature distribution crumbling (Fig. 2a) have been largely ignored. This oversight substantially hinders model scalability when continuously embedding new classes with few samples to a feature space trained upon old classes.



(a) Old class Performance (Test set) (b) New class Performance

Fig. 1. The forgetting and over-fitting issues in FSCIL.

In this study, we propose an adaptive yet minimally expanded network, referred to as the dynamic support network (DSN), to "support" feature distributions when performing FSCIL, Fig. 2b. During incremental learning, DSN tentatively expands network nodes to aggregate the representation capacity for new classes. New feature representation is then obtained by merging network outputs before ($i.e.$, representations learned from old classes) and after ($i.e.$, representations learned from new classes) node expansion. The expanded network is then compressed with a node self-activation mechanism to pursue compact feature representation which not only supports class distributions but also avoids over-fitting. Specifically, network outputs are used to predict an indicator vector which highlights the

- *B. Yang, B. Liu, Y. Zhang and Q. Ye are with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, 101408, China. Qixiang Ye is the corresponding author. E-mail: qxye@ucas.ac.cn.*
- *M. Lin is with the Youtu Laboratory, Tencent, Shanghai 200233, China.*
- *R. Ji is with the Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China.*
- *X. Liang is with the Sun Yat-Sen University, Guangzhou 510006, China.*

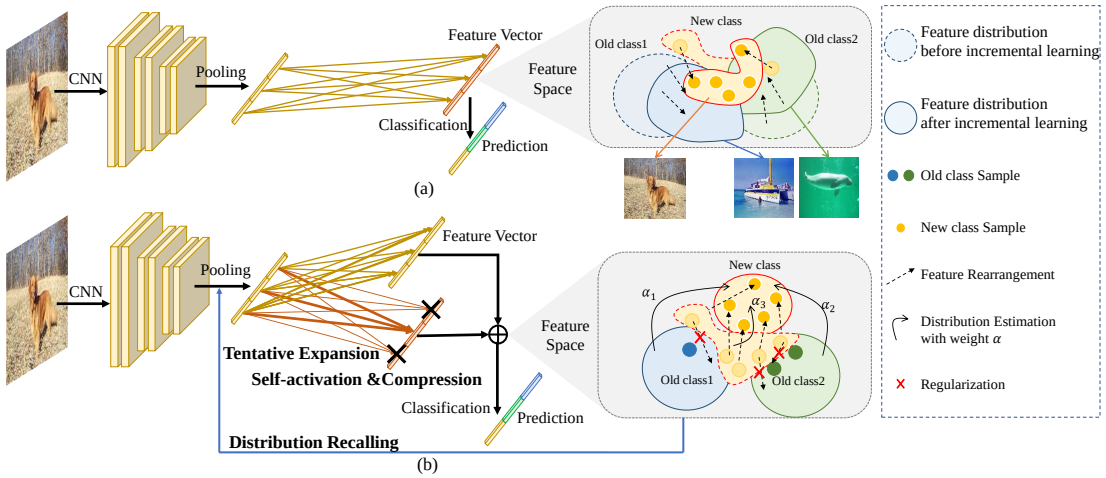


Fig. 2. Dynamic support network (DSN). (a) A scale-fixed network with a feature space of limited representation capacity. Embedding examples of new classes into the feature space causes distribution crumbling. (b) DSN refers to an adaptively updating network with compressive node expansion. When the network is trained upon new classes, it tentatively expands network nodes to learn new class features and then compresses the redundant nodes to provide compact feature representation. Meanwhile, the old class distribution in this feature space is recalled to regularize incremental learning. DSN dynamically "supports" the feature space and feature distribution to avoid class confusion.

importance of network nodes. The predicted vector then updates network outputs. Through iterative node activation, DSN pursues a sparse indicator vector with minimal node expansion. Meanwhile, DSN selectively recalls old classes in each incremental session. To this end, it first estimates new class distributions in the current session. In the next session, the incremental new classes change to old classes and a sampling procedure is performed to generate feature vectors from old class distributions for model training. In this manner, feature distributions are supported and thereby avoid class confusion, Fig. 2b.

The contributions of this study include:

- We propose the dynamic support network (DSN), which provides a feature space regularized solution to forgetting and over-fitting observed in FSCIL.
- We design a node self-activation mechanism to produce sparse indicator vectors and pursue minimal node expansion to alleviate over-fitting.
- We design a distribution recalling strategy, which adaptively samples feature vectors from class distributions and supports feature distribution to avoid class confusion.
- We substantially improve FSCIL performance and achieve new state-of-the-art performance on commonly used benchmarks.

2 Related Works

Few-shot Learning. Metric learning [6]–[11] trained two-branch networks to compare the few-shot training images with query (test) images and determine categories of the query images. Meta learning [12]–[14] pursued faster adaptation of models to new categories with few training images. Data augmentation [15]–[17] generated examples of rich transformations to enhance feature representation. As a sophisticated data augmentation approach, distribution calibration [18] estimated few-shot sample distributions from similar base classes.

However, when mitigated to new classes the performance of base classes degenerate, *i.e.*, catastrophic forgetting. To solve, Gidaris *et al*. [19] introduced the few-shot classification weight generator to comprise new and old classes. Openmix [20] built the connection between old classes and new classes using data augmentation. A classification weight generator based on the attention mechanism [6] was proposed to retain the representation of base classes when learning new classes. Meta-learning [21], [22] and feature alignment methods [23] were explored to regularize the learning procedure of new classes. Memory networks are effective to overcome catastrophic forgetting. LRRE [24] involved a life-long memory module, which performs predictions by leveraging the knowledge from past data with similar activations. The compound memory network (CMN) [25], [26] leveraged the label independent memory (LIM) to cache label related features and encode the large amount of unlabeled data. The LIM mechanism solved catastrophic forgetting by encoding all the old-class examples with a limited number of examples.

Incremental Learning. Researches can be broadly classified as either task- or class-incremental learning (CIL) [27]. The former included rehearsal methods [28]–[32], data distillation [29], [33]–[37], and architecture configuration [38]–[41]. Rehearsal methods recalled exemplars from a previous session to prevent forgetting. Data distillation Regularization [33], [37] methods introduced regularization loss functions, which constrain network outputs to prevent forgetting. Architecture configuration methods leveraged hard attention [39], prunning and pack mechanisms [38] to choose parameters during inference. The dynamic expansion network [40] used a three-step strategy with node retraining, node expansion, and node splitting. When task IDs are not accessible during inference, task-incremental learning evolves to class-incremental learning [28]. The primary challenge is catastrophic forgetting, which was elaborated by data distillation [33], [42], memory schemes [34], [43], and transfer learning [44].

Network growing [45]–[47] not only learned resource-efficient neural architectures but also alleviated catastrophic forgetting in incremental learning. Splitting Network [45] optimized the network structure by partitioning existing neurons to multiple off-springs. ConvexNetwork [46] converted network node selection as a convex optimization problem, which were solved by incrementally inserting a hidden unit at a time and each time finding a linear classifier that minimizes a weighted sum of errors. ArchitectureDescent [47] defined a general framework to flexibly grow networks, both wider and deeper, by jointly optimizing the networks' parameters and architectures.

Existing methods have advanced the study of class incremental learning, but unfortunately ignored feature space expansion, which cause class confusion when more and more categories are embedded to a fixed feature space.

Few-shot Class Incremental Learning. FSCIL faces challenges which appear beyond conventional learning paradigms. Compared with incremental learning, it is puzzled by over-fitting brought about by few-shot examples. Compared with few-shot learning, it has catastrophic forgetting and requires to avoid overwhelming the original feature space when adding few samples from incremental classes. The neural gas method [2] attempted to solve this problem by preserving feature topology. The dynamic few-shot learning method [19] introduced a classification weight generator to learn feature representations which generalized well on "unseen" categories. However, neither of the methods systematically considers the feature distribution support problem, which is the focus of this study.

3 Preliminary

FSCIL trains a model upon base classes C_{base} with sufficient data and continually generates the model to novel classes C_{novel} with only few samples from streaming datasets $\{D^{(t)}, t = 0, 1, 2, ...\}$, where $D^{(t)}$ denotes data for the t-th incremental class set $C^{(t)}$. For $\forall_{t_1 \neq t_2}$, we have $C^{(t_1)} \cap C^{(t_2)} = \varnothing$, $C^{(0)} = C_{\text{Base}}$, and $\cup_t C^{(t)} = C_{\text{novel}}$ with $t > 0$. In the t-th session, only dataset $D^{(t)}$ of class set $C^{(t)}$ is available to train the model while all seen classes $\{C^{(0)}, ..., C^{(t)}\}$ require to be tested. In this procedure, the old classes $\{C^{(0)}, \cdots, C^{(t-1)}\}$ cannot be forgotten while the few-shot examples from new classes should not be overfitted.

A naive solution for FSCIL is first to train a network with C_{base}, and then fine-tune the network using data $D^{(t)}$ from incremental classes $C^{(t)}$. For the base training session (0-th session), each image $\boldsymbol{I} \in D^{(0)}$ is fed to a network to extract a feature vector $\boldsymbol{x} = f(\boldsymbol{I}; \theta_b) \in \mathbb{R}^c$, where $f(\cdot)$ denotes the backbone network parameterized with θ_b. A network layer parameterized using θ_o is then introduced to project extracted feature vectors to a feature space $f(\boldsymbol{x}; \theta_o)$. Taking $g(\cdot)$ parameterized by $\theta_c^{(0)}$ as the classifier, we have the network prediction $\hat{\boldsymbol{y}}^{(0)} = g\big(f(\boldsymbol{x}; \theta_o); \theta_c^{(0)}\big)$. During base model training (t=0), with image ground-truth $\boldsymbol{y}$, the following classification loss function can be optimized, using

$$\arg\min_{\theta} \mathcal{L}_{cls}(\hat{\boldsymbol{y}}^{(0)}, \boldsymbol{y}; \theta), \tag{1}$$

where $\theta = \{\theta_b, \theta_o, \theta_c\}$, $\theta_c = \theta_c^{(0)}$, and $\mathcal{L}_{cls} = \boldsymbol{y} \log(\hat{\boldsymbol{y}}^{(0)})$ is the cross entropy loss function.

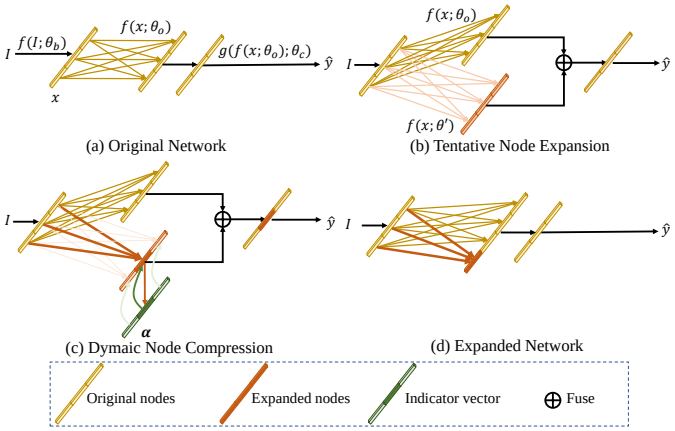


Fig. 3. Network expansion and compression.

For incremental learning sessions ($t > 0$), to classify new classes $C^{(t)}$, additional parameters $\theta_c^{(t)}$ are added to the network's classification layer. Accordingly, the network prediction becomes $\hat{\boldsymbol{y}}^{(t)} = g\big(f(\boldsymbol{x}; \theta_o); \theta_c\big)$, where $\theta_c = \{\theta_c^{(0)}, ..., \theta_c^{(t)}\}$. To avoid forgetting old classes, a knowledge distillation method [33] is adopted to retain the parameters learnt from old classes. The distillation loss is defined as $\mathcal{L}_{dis} = \hat{\boldsymbol{y}}^{(t-1)} \log \hat{\boldsymbol{y}}^{(t)}$, where $\hat{\boldsymbol{y}}^{(t-1)}$ is the output of the network trained in the $(t-1)$-th session. With knowledge distillation, FSCIL's objective function is defined as

$$\begin{aligned} &\arg\min_{\theta} \mathcal{L}_{inc}(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}; \theta) \\ &\qquad = \mathcal{L}_{cls}(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}; \theta) + \lambda_1 \mathcal{L}_{dis}(\hat{\boldsymbol{y}}^{(t)}, \hat{\boldsymbol{y}}^{(t-1)}; \theta), \end{aligned} \tag{2}$$

where $\theta = \{\theta_o, \theta_c\}$. $\theta_c = \{\theta_c^{(0)}, ..., \theta_c^{(t)}\}$. λ_1 is a regularization factor.

With knowledge distillation, the above approach reduces the catastrophic forgetting, but remains challenged by the over-fitting issue when only few training examples are available. However, the few samples from new classes make it hard to retain the old class distributions, Fig 2(a), which would be solved by DSN introduced below.

4 Dynamic Support Network

DSN refers to an adapatively expanded network, which "supports" the feature space when performing few-shot class incremental learning, Fig. 2b. In each incremental session, DSN leverages compressive network expansion to enrich feature representation, as well as recalling old class distributions to dynamically regularize the feature space.

4.1 Compressive Expansion: Feature Space Support

Before the incremental classes are trained, it is unknown about how many nodes should be expanded. We propose to tentatively expand the network so that it can accommodate the incremental classes. A self-activation mechanism is then introduced to judge whether the expanded nodes require to be preserved or not, $i.e.$, self-activated network compression. In this way, DSN constructs a minimal network to sup-

port the distributions of all classes, as well as regularizing the feature space, Fig. 3. This is implemented by optimizing

$$\arg\min_{\theta,\theta'} \mathcal{L}_{inc}(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}; \theta, \theta') + \lambda_2 \mathcal{L}_{exp}(\boldsymbol{\alpha}^{(t)}; \theta'), \quad (3)$$

where $\mathcal{L}_{inc}(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}; \theta, \theta')$ is the FSCIL loss defined in Eq. 2. θ' is the expanded network parameter introduced below. $\boldsymbol{\alpha}^{(t)}$ is a vector indicating node activation. $\mathcal{L}_{exp}$ is the loss function defined for node expansion and activation, as

$$\mathcal{L}_{exp} = ReLU(\frac{\|\boldsymbol{\alpha}^{(t)}\|_1}{c} - \tau_t), \quad (4)$$

where $ReLU$ is the active function [48]. $\tau_t = 0.1$ is a threshold determined by linear search and fixed across datasets. The right hand of Eq. 4 puts sparse constraints on $\boldsymbol{\alpha}^{(t)}$ with the L1-norm, which is used to make the network sparse.

Tentative Node Expansion. This is performed by tentatively adding nodes to a network layer, Fig. 3(b). The expanded layer can be either the last convolutional layer or the fully connected (FC) layer. The expanded node number is set to be equal to that of the layer before expansion so that the feature of each node is enriched. Denoting $\theta'^{(t)}$ parameters of the expanded layer in the t-th training session, the features extracted by the expanded nodes are $f(\boldsymbol{x}; \theta'^{(t)})$. The feature output $f(\boldsymbol{x}; \theta'^{(t)})$ from the expanded nodes are fused with those $f(\boldsymbol{x}; \theta_o)$ from the nodes before expansion using a plus operation, as $f(\boldsymbol{x}; \theta'^{(t)}) \oplus f(\boldsymbol{x}; \theta_o)$. The outputs of the expanded network are

$$\hat{\boldsymbol{y}}^{(t)} = g\big(f(\boldsymbol{x}; \theta'^{(t)}) \oplus \gamma f(\boldsymbol{x}; \theta_o)); \theta_c\big), \quad (5)$$

where $g(\cdot)$ denotes the classifier.

Self-Activated Node Compression. To minimize the number of expanded nodes, a variable vector $\boldsymbol{\alpha}^{(t)} \in \{0, 1\}^c$, is used to indicate which nodes are representative for the new classes, Fig. 3(c). With the indicator vector, the expanded features are calculated as

$$\big(\boldsymbol{\alpha}^{(t)} \odot f(\boldsymbol{x}; \theta'^{(t)})\big) \oplus \big(\gamma f(\boldsymbol{x}; \theta_o)\big), \quad (6)$$

where $\odot$ denotes the element-wise multiplication and $\oplus$ the plus operation. γ is a regularization factor. Accordingly, the network prediction is updated to

$$\hat{\boldsymbol{y}}^{(t)} = g\big(\big(\boldsymbol{\alpha}^{(t)} \odot f(\boldsymbol{x}; \theta'^{(t)})\big) \oplus \big(\gamma f(\boldsymbol{x}; \theta_o)\big); \theta_c\big). \quad (7)$$

According to Eq. 6, $\boldsymbol{\alpha}(i)^{(t)} \rightarrow 1$ implies that the i-th expanded node is important for the incremental classes while $\boldsymbol{\alpha}(i)^{(t)} \rightarrow 0$ implies that the expanded node is trivial. A similar indicator vector is used in the DART method [49]. Differently, the indicator vector for DART is defined as additional learnable network parameters, which aggravate the risk of overfitting. To be adaptive, $\boldsymbol{\alpha}^{(t)}$ is designed to be dependent on the outputs of the nodes in an positive self-activation fashion, as

$$\boldsymbol{\alpha}^{(t)} = \frac{1}{1 + e^{-\beta f(\boldsymbol{x}; \theta'^{(t)})}}, \quad (8)$$

where β is a magnification factor and the features are updated as

$$f(\boldsymbol{x}; \theta'^{(t)}) \leftarrow \boldsymbol{\alpha}^{(t)} \odot f(\boldsymbol{x}; \theta'^{(t)}). \quad (9)$$

According to Eq. 8, the indicator vector $\alpha^{(t)}$ is positively dependent on features $f(\boldsymbol{x}; \theta'^{(t)})$ of the expanded nodes. Although SE-Net [50] and GLU [51] defined node activation mechanisms upon indicator vectors, both of them require additional network parameters to learn the indicator vectors. When only few new class samples are available, additional network parameters aggregate the risk of overfitting. In this study, the indicator vector $\boldsymbol{\alpha}^{(t)}$ is designed to be parameter-free, $i.e.$, they are defined upon network node values in an positive self-activation fashion. According to Eq. 9, $f(\boldsymbol{x}; \theta'^{(t)})$ are positively dependent on the indicator vector $\boldsymbol{\alpha}^{(t)}$. When training proceeds, elements of $\boldsymbol{\alpha}^{(t)}$ and $f(\boldsymbol{x}; \theta'^{(t)})$ would be amplified or dwindled. According to Eq. 3 and Eq. 4, $\alpha^{(t)}$ would have sparse non-zero elements, which infers that $f(\boldsymbol{x}; \theta'^{(t)})$ would be sparse. Sparse network nodes corresponding to significant features are preserved while those corresponding to trivial features are discarded. In this way, DSN realizes network compression towards compact feature representation, Fig. 3(d).

4.2 Old Class Recalling: Feature Distribution Support

Although compressive network expansion facilitates supporting the feature space, lacking the constraint of old class distributions could lead to confusion between new and old classes. According to FSCIL settings, the old class samples are unavailable during incremental learning. Meanwhile, memorizing all the old class data is expensive. To solve, we propose to recall old class distributions to support the feature space, Fig. 4.

New Class Distribution Estimation. To obtain old class distributions, the new class distribution requires an estimation in the current learning session. Assume that each base class follows a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. The distribution ($i.e.$, mean μ and covariance Σ) for a base class can be estimated by all samples belonging to this class. Supposing that the few samples follow an unbiased distribution, the mean μ vector can be calculated upon these examples. Denote $\boldsymbol{x}_i$ the mean feature vector of the i-th new class generated by the network in the t-th training session and the network prediction of x_i is $\hat{\boldsymbol{y}}_i^{(t)}$. The few-shot samples are insufficient to precisely estimate the covariances of the new classes. Inspired by the distribution calibration method [18], we leverage the distributions of similar old classes to estimate the covariances of the new classes. For the i-th new class, the covariance is calculated by

$$\Sigma_i = \bar{\boldsymbol{y}}_i^{(t)\intercal} \boldsymbol{\Sigma}, \quad (10)$$

where $\boldsymbol{\Sigma} = [..., \Sigma_m, ...]^\intercal$ is a matrix by concatenating Σ_m. Σ_m in $\boldsymbol{\Sigma}$ is a covariance for either a new or an old class. $\bar{\boldsymbol{y}}_i^{(t)}$ denotes the mean vector of network predictions for the i-th new class data. This is used to estimate the similarity between new and old classes. In the early training stages, the predictions for new class samples are inaccurate so that Σ_i largely depends upon the old class distributions. When training proceeds, the predictions for new class samples become accurate ($\hat{\boldsymbol{y}}_i^{(t)} \in \hat{\boldsymbol{y}}^{(t)}$ becomes significant) and the dependency upon old classes decreases.

Old Class Distribution Sampling. When incremental learning proceeds, the new classes, of which the distributions have been estimated in the last session, change to old classes. The old classes with estimated distributions are

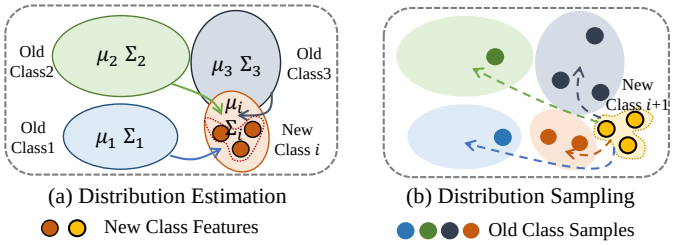


Fig. 4. Distribution recalling for feature space support. (a) In the current learning session, the new class distribution is estimated by the old class distributions. (b) In the next learning session, feature vectors are sampled from old classes to regularize the incremental learning.

then recalled during the next incremental session to avoid class confusion, which is performed by sampling N_j feature vectors from the j-th old class distributions, as

$$\boldsymbol{x}'_j = Sample(\mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j), N_j), \tag{11}$$

where $Sample(\cdot)$ denotes a sampling function, which draws random samples from a given Guassian distribution.

To reduce feature distribution confusion, more feature vectors are sampled from the old classes which are close to new classes. The sampling number N_j of j-th class is determined by the mean vector, $\bar{\boldsymbol{y}}_i^{(t)}$, of predictions for the i-th new class data, as

$$N_j = \begin{cases} N_j + 1 & \bar{y}_i^{(t)}(j) \in topk(\bar{\boldsymbol{y}}_i^{(t)}) \\ N_j & otherwise \end{cases}, \tag{12}$$

where $topk(\cdot)$ is a function to select top-k largest elements from a vector. $\bar{y}_i^{(t)}(j)$ denotes j-th element of $\bar{\boldsymbol{y}}_i^{(t)}$.

Denote $\boldsymbol{x}'$ the sampled feature vectors, the network predictions of $\boldsymbol{x}'$ are

$$\hat{\boldsymbol{y}}'^{(t)} = g\big(\big(\boldsymbol{\alpha}^{(t)} \odot f(\boldsymbol{x}'; \theta'^{(t)})\big) \oplus \big(\gamma f(\boldsymbol{x}'; \theta_o)\big); \theta_c\big). \tag{13}$$

With the sampled feature vectors $\boldsymbol{x}'$ and their corresponding predictions $\hat{\boldsymbol{y}}'$, DSN is trained by optimizing the following loss function, as

$$\begin{aligned} \arg\min_{\theta,\theta'} & \mathcal{L}_{inc}(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}; \theta, \theta') \\ & + \lambda_2 \mathcal{L}_{exp}(\boldsymbol{\alpha}^{(t)}; \theta') + \lambda_3 \mathcal{L}_{rec}(\hat{\boldsymbol{y}}'^{(t)}, \boldsymbol{y}'; \theta, \theta'), \end{aligned} \tag{14}$$

where $\mathcal{L}_{exp}(\boldsymbol{\alpha}^{(t)}; \theta')$ defined in Eq. 4 is for compressive network expansion. $\mathcal{L}_{rec}(\hat{\boldsymbol{y}}'^{(t)}, \boldsymbol{y}'; \theta, \theta') = \mathcal{L}_{cls}(\hat{\boldsymbol{y}}'^{(t)}, \boldsymbol{y}'; \theta, \theta') + \lambda_1 \mathcal{L}_{dis}(\hat{\boldsymbol{y}}'^{(t)}, \hat{\boldsymbol{y}}'^{(t-1)}; \theta, \theta')$ is defined for distribution recalling. $\boldsymbol{y}'$ is the ground-truth label of corresponding few-shot samples. λ_1 and λ_2 are empirically defined regularization factors, which are set to 1.0 and 0.1 respectively in our experiments.

5 Experiments

5.1 Experimental Setting

Dataset. We evaluate DSN on CIFAR 100, CUB200 and miniImageNet. The categories in the datasets are divided into base ones with adequate annotations and new ones with K-shot annotated images. For FSCIL, the network is trained upon base classes for the first session. New classes are gradually added to train DSN in T incremental sessions. In each incremental session, N-way new classes are added. CIFAR100 and miniImageNet consist of 100 classes, where 60 classes are set as base classes and 40 as new classes. Each new class has 5-shot annotated images (K = 5). The new classes are divided into 8 sessions (T = 8), each of which has 5 classes (N = 5). CUB200 consists of 200 classes where 100 classes are set as base classes and the other 100 classes as new classes under the settings of K = 5, T = 10, N = 10.

Training and Evaluation. DSN is built upon the Resnet18 network and optimized with the SGD algorithm. Four data augmentation strategies, including normalization, horizontal flipping, random cropping, and random resizing, are used. For the first incremental session, we train the network using $D^{(0)}$ upon the base classes. When $t > 0$, the network is trained upon $D^{(t)}$ with new classes. DSNs trained for all T sessions are respectively evaluated on all of the seen classes. The performance drop ("PD") is defined as $PD = ACC^{(0)} - ACC^{(T)}$, where $ACC^{(0)}$ denotes the 0-th session performance and $ACC^{(T)}$ denotes the T-th session performance. The performance retention ("PR") is defined as $PR = ACC^{(T)}/ACC^{(0)}$.

5.2 Performance

CUB200. Table 1 presents DSN's performance and comparison with the state-of-the-art methods with the Resnet-18 backbone. In the final session, DSN outperforms the state-of-the-art CEC method [53] by 10.93% (63.21% *vs.* 52.28%), which is a substantial improvement. This demonstrates DSN's plausibility to alleviate both the catastrophic forgetting and overfitting issues in FSCIL. **CIFAR100.** Comparisons provided in Table 2 also show that DSN is equivalent to state-of-the-art methods in early sessions, but significantly outperforms them when learning proceeds. **miniImageNet.** Table 3 depicts DSN's performance on miniImageNet dataset with the ResNet18 backbone. DSN offers the lowest PD rate and the highest PR rate. Particularly, the PD of DSN outperforms that of state-of-the-art CEC by 3.6% (21.06% *vs.* 24.37%) and PR by 2.39% (68.54% *vs.* 66.15%). Again, these are substantial margins for such a challenging task. The performance gain on CUB200 is larger than those on CIFAR100 and miniImagenet datasets, which can be attributed to the following reasons. On the one hand, the class number of CUB200 is 200 while the others are 100. The larger class number implies more serious class confusion in the feature space. On the other hand, CUB200 is defined for fine-grained classes (bird species), which have smaller inter-class distances and larger feature distribution crumbling. On CUB200, DSN can play out its advantages by supporting the feature space to avoid class confusion and feature distribution crumbling.

5.3 Model Analysis

Feature Space/Distribution Support. When new class samples are embedded to the baseline feature space, feature distribution crumbling is originally observed, Fig. 5. In contrast, when training proceeds and more classes are embedded to the feature space, DSN appears to support the feature space to alleviate feature distribution crumbling.

Evolution of $\boldsymbol{\alpha}$. In Fig. 6, in early training epochs, $\boldsymbol{\alpha}_t$ falls into $[0, 1]$ and all expanded nodes are activated to

TABLE 1
Performance comparison on CUB200 using the Resnet18 backbone. "PD" denotes the performance drop. "PR" denotes the performance retention.

Method	sessions											PD ↓	PR ↑
	0	1	2	3	4	5	6	7	8	9	10		
Ft-CNN	68.68	44.81	32.26	25.83	25.62	25.22	20.84	16.77	18.82	18.25	17.18	51.50	25%
iCaRL [28]	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	47.52	30.80%
EEIL [52]	68.68	53.63	47.91	44.20	36.30	27.46	25.93	24.70	23.95	24.13	22.11	46.57	32.19%
NCM [53]	68.68	57.12	44.21	28.78	26.71	25.66	24.62	21.52	20.12	20.06	19.87	48.81	28.92%
TOPIC [2]	68.68	62.49	54.81	49.99	45.25	41.40	38.35	35.36	32.22	28.31	26.28	42.40	38.26%
SKW [4]	68.28	60.45	55.70	50.45	45.72	42.90	40.89	38.77	36.51	34.87	32.96	35.32	47.99%
FSLL [3]	68.72	65.67	62.33	58.10	55.44	52.66	51.17	50.27	48.31	47.25	45.55	23.17	66.28%
CEC [5]	**75.85**	71.94	68.50	63.5	62.43	58.27	57.73	55.81	54.83	53.52	52.28	23.57	68.92%
DSN (ours)	**80.86**	**78.18**	**75.57**	**72.68**	**71.42**	**70.12**	**69.16**	**67.94**	**66.99**	**65.10**	**63.21**	**17.65**	**78.17%**

TABLE 2
Performance comparison on CIFAR100 using the ResNet18 backbone.

Method	session									PD ↓	PR ↑
	0	1	2	3	4	5	6	7	8		
Ft-CNN	64.10	36.91	15.37	9.80	6.67	3.80	3.70	3.14	2.65	61.45	4.13%
iCaRL [28]	64.10	53.28	41.69	34.13	27.93	25.06	20.41	15.48	13.73	50.37	21.41%
EEIL [52]	64.10	53.11	43.71	35.15	28.96	24.98	21.01	17.26	15.85	48.25	24.72%
NCM [53]	64.10	53.05	43.96	36.97	31.61	26.73	21.23	16.78	13.54	50.56	21.12%
TOPIC [2]	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	34.73	45.81%
CEC [5]	73.07	68.88	65.26	61.19	58.09	55.57	53.22	51.34	49.14	23.93	67.25%
DSN (ours)	73.00	68.83	64.82	**62.64**	**59.36**	**56.96**	**54.04**	**51.57**	**50.00**	**23.00**	**68.49%**

TABLE 3
Performance Comparison on miniImageNet using the ResNet18 backbone.

Method	session									PD ↓	PR ↑
	0	1	2	3	4	5	6	7	8		
Ft-CNN	61.31	27.22	16.37	6.08	2.54	1.56	1.93	2.60	1.40	59.91	2.28%
iCaRL [28]	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	44.1	28.07%
EEIL [52]	61.31	46.58	44.00	37.29	33.14	27.12	24.10	21.57	19.58	41.73	31.93%
NCM [53]	61.31	47.80	39.31	31.91	25.68	21.35	18.67	17.24	14.17	47.14	23.11%
TOPIC [2]	61.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	36.89	39.83%
CEC [5]	72.00	66.86	62.97	59.43	56.70	53.73	51.19	49.24	47.63	24.37	66.15%
DSN (ours)	66.95	62.46	58.78	55.64	52.85	51.23	48.9	46.78	45.89	**21.06**	**68.54%**

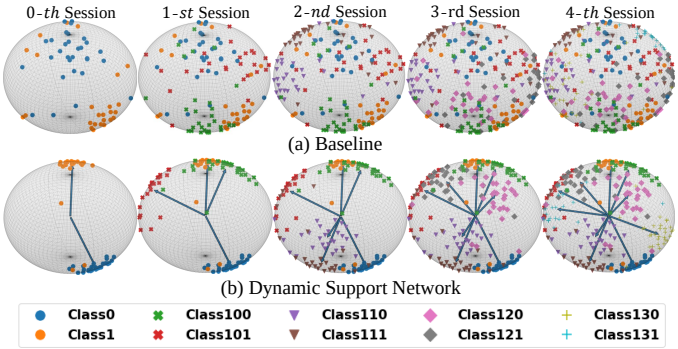


Fig. 5. Feature spaces of the baseline approach and our DSN approach. Adding new class samples to the feature space projected by the baseline causes distribution crumbling. The feature space of DSN is better supported to avoid distribution crumbling.

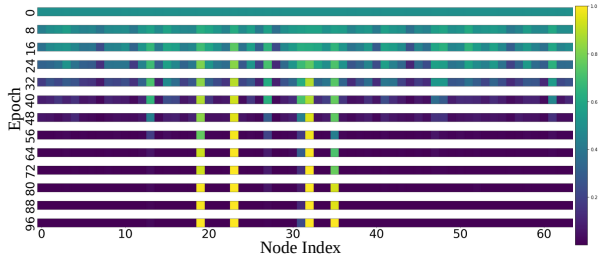


Fig. 6. Evolution of the indicator vector (α) for self-activated node compression. (Lighter color denotes larger values)

construct redundant feature representation. When training proceeds, α approaches 0 or 1and $\|\alpha\|_1$ saturated to τ so that few (approximately 10%) expanded nodes are activated

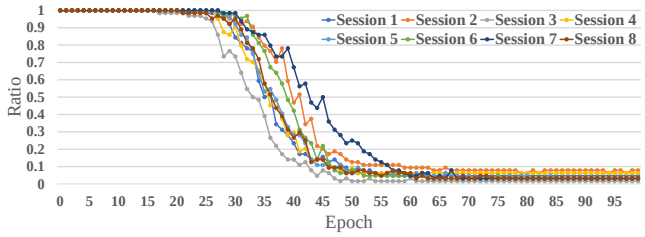
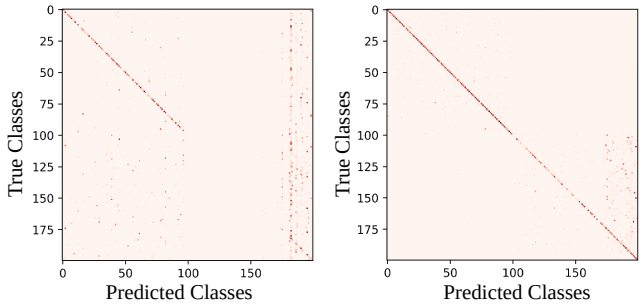


Fig. 7. Activation ratio of the extended nodes during training.



(a) Baseline (b) DSN

Fig. 8. Confusion matrices between old classes (0-189) and new classed (190-199). (Best viewed in color with zoom)

to learn compact feature representation. The experiments validate that the performance gain does not attribute to the huge additional node cost.

Class Confusion. In Fig. 8, we compare the classification confusion matrices of the baseline method and DSN. For the baseline approach, the diagonal elements of the confusion matrix have small values, particularly for the old classes (0-189) and larger values for the new classes (190-199), which implies that training with few samples overfits the new classes and forgets the old ones. With DSN, the diagonal elements of the confusion matrix become larger (particularly for the old classes), which means less class confusion.

5.4 Ablation Study

5.4.1 Compressive Network Expansion

In Table 4, we validate the components of DSN on the CUB dataset using the performance for the last session. With compressive node expansion, the performance improves by 13.59% (37.90% *vs.* 24.31%). Table 5 evaluates the performance by applying compressive node expansion to different network layers, $e.g.$, the 3×3 conv. layer, 1×1 conv. layer and the fully connected layer. DSN achieves comparable performance for the three network layers. The best results are from the fully connected (fc) layer, as this is more close to the feature representation and can be better optimized.

5.4.2 Old Class Recalling

In Table 4, with old class recalling, the performance further increases by 25.31% (from 63.21% to 37.90%). Such a large margin attributes to the reduction of overfitting and the inter-session confusion. While the feature recalling strategy significantly outperforms the image recalling strategy.

Class Distribution Estimation. In Table 6, we compare the proposed distribution estimation (DE) approach with the distribution calibration (DC) approach [18]. DSN with

TABLE 4
Ablation study of DSN on CUB Dataset using the Resnet18 backbone. "Images" denotes the Old Class Recalling using the old class images. "Features" denotes the Old Class Recalling using the sampled features.

Baseline	✓				
Expansion		✓	✓	✓	✓
Compression			✓	✓	✓
Image Recalling				✓	
Feature Recalling					✓
ACC	24.31	28.41	37.90	48.19	63.21

TABLE 5
Comparison of compressive node expansion on different network layers. "FC" denotes the fully connected layer.

Net Layer	3×3 conv.	1×1 conv.	FC
Acc.	37.05	35.76	**37.90**

DE achieves a marked improvement in terms of performance (59.22% $vs.$ 51.84%). The reason for such an improvement is that DE makes full use of the learnt distributions to precisely estimate new class distributions. When using all old classes for distribution estimation (ODE), DSN further improves the accuracy by 2.16% (from 59.22% to 61.38%).

Class Distribution Sampling. In Table 7 we compare two feature sampling strategies. The adaptive sampling strategy (ASS), which can determine the sampling number from each old classes according to the network prediction, outperforms the equal sampling strategy (ESS), which samples equal numbers of features from all classes. The reason lies in that ASS focuses on the old classes which are close to new classes and this facilitates feature discriminability.

6 Conclusion

We proposed Dynamic Support Network (DSN), providing a systematic approach to solve the catastrophic forgetting and over-fitting issues in few-shot class incremental learning (FSCIL). DSN constructed a scalable framework for feature representation expansion. On a simple baseline, DSN improved the FSCIL performance, in striking contrast with the state-of-the-art approaches. From the perspective of feature space support and feature distribution support, DSN provides fresh insights to the challenging FSCIL problem.

References

[1] A. G. Greenwald, "Cognitive learning, cognitive response to persuasion, and attitude change," *Psychological Foundations of Attitudes*, 1968.
[2] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, "Few-shot class-incremental learning," in *IEEE CVPR*, pp. 12180–12189, 2020.
[3] P. Mazumder, P. Singh, and P. Rai, "Few-shot lifelong learning," in *AAAI*, 2021.
[4] A. Cheraghian, S. Rahman, P. Fang, S. K. Roy, L. Petersson, and M. Harandi, "Semantic-aware knowledge distillation for few-shot class-incremental learning," in *IEEE CVPR*, 2021.
[5] C. Zhang, N. Song, G. Lin, Y. Zheng, P. Pan, and Y. Xu, "Few-shot incremental learning with continually evolved classifiers," in *IEEE CVPR*, 2021.

TABLE 6
Comparison of distribution estimation strategies. "DE" estimates new class distributions using base class distributions (classes of the 0-th session). "DEO" estimates new class distributions using old class distributions (classes from 0-th to $(t-1)$-th sessions).

Method	Acc.
Distribution Calibration [18]	51.84
Distribution Estimation (DE)	59.22
Distribution Estimation with old classes(DEO)	**61.38**

TABLE 7
Comparison of distribution sampling strategies.

Method	Acc.
Equal Sampling Strategy (ESS)	61.38
Adaptive Sampling Strategy (ASS)	**63.21**

[6] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NeurIPS*, pp. 3630–3638, 2016.
[7] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, pp. 4077–4087, 2017.
[8] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *IEEE CVPR*, pp. 1199–1208, 2018.
[9] C. Zhang, Y. Cai, G. Lin, and C. Shen, "Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *IEEE CVPR*, pp. 12200–12210, 2020.
[10] B. Yang, C. Liu, B. Li, J. Jiao, and Q. Ye, "Prototype mixture models for few-shot semantic segmentation," in *ECCV*, pp. 763–778, 2020.
[11] B. Liu, J. Jiao, and Q. Ye, "Harmonic feature activation for few-shot semantic segmentation," *IEEE Trans. Image Process.*, vol. 30, pp. 3142–3153, 2021.
[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, pp. 1126–1135, 2017.
[13] T. Elsken, B. Staffler, J. H. Metzen, and F. Hutter, "Meta-learning of neural architectures for few-shot learning," in *IEEE CVPR*, pp. 12362–12372, 2020.
[14] Q. Sun, Y. Liu, T. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *IEEE CVPR*, pp. 403–412, 2019.
[15] H. Zhang, J. Zhang, and P. Koniusz, "Few-shot learning via saliency-guided hallucination of samples," in *IEEE ICCV*, pp. 2770–2779, 2019.
[16] K. Li, Y. Zhang, K. Li, and Y. Fu, "Adversarial feature hallucination networks for few-shot learning," in *IEEE ICCV*, pp. 13467–13476, 2020.
[17] J. Kim, H. Kim, and G. Kim, "Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning," in *ECCV*, pp. 599–617, 2020.
[18] S. Yang, L. Liu, and M. Xu, "Free lunch for few-shot learning: Distribution calibration," in *ICLR*, 2021.
[19] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *IEEE CVPR*, pp. 4367–4375, 2018.
[20] Z. Zhong, L. Zhu, Z. Luo, S. Li, Y. Yang, and N. Sebe, "Openmix: Reviving known knowledge for discovering novel visual categories in an open world," in *IEEE CVPR*, pp. 9462–9470, 2021.
[21] M. Ren, R. Liao, E. Fetaya, and R. S. Zemel, "Incremental few-shot learning with attention attractor networks," in *NeurIPS*, pp. 5276–5286, 2019.
[22] S. W. Yoon, D. Kim, J. Seo, and J. Moon, "Xtarnet: Learning to extract task-adaptive representation for incremental few-shot learning," in *ICML*, pp. 10852–10860, 2020.
[23] Q. Liu, O. Majumder, A. Achille, A. Ravichandran, R. Bhotika, and S. Soatto, "Incremental few-shot meta-learning via indirect discriminant alignment," in *ECCV*, pp. 685–701, 2020.
[24] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *ICLR*, 2017.
[25] L. Zhu and Y. Yang, "Compound memory networks for few-shot video classification," in *ECCV*, pp. 782–797, 2018.
[26] L. Zhu and Y. Yang, "Label independent memory for semi-supervised few-shot video classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 273–285, 2022.
[27] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: survey and performance evaluation," *CoRR*, vol. abs/2010.15277, 2020.
[28] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *IEEE CVPR*, pp. 5533–5542, 2017.
[29] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *ECCV*, pp. 556–572, 2018.
[30] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *IEEE CVPR*, pp. 374–382, 2019.
[31] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *NeurIPS*, pp. 2990–2999, 2017.
[32] Y. Xiang, Y. Fu, P. Ji, and H. Huang, "Incremental learning using conditional adversarial networks," in *IEEE ICCV*, pp. 6618–6627, 2019.
[33] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, 2018.
[34] P. Dhar, R. V. Singh, K. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *IEEE CVPR*, pp. 5138–5146, 2019.
[35] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*, pp. 3987–3995, 2017.
[36] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE ICCV*, pp. 618–626, 2017.
[37] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *IEEE CVPR*, pp. 831–839, 2019.
[38] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *IEEE CVPR*, pp. 7765–7773, 2018.
[39] J. Serrà, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *ICML*, pp. 4555–4564, 2018.
[40] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *ICLR*, 2018.
[41] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *ECCV*, pp. 72–88, 2018.
[42] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *IEEE ICCV*, pp. 3420–3429, 2017.
[43] Y. Liu, Y. Su, A. Liu, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *IEEE CVPR*, pp. 12242–12251, 2020.
[44] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *ICLR*, 2019.
[45] L. Wu, D. Wang, and Q. Liu, "Splitting steepest descent for growing neural architectures," in *NeurIPS 2019*, pp. 10655–10665, 2019.
[46] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *NeurIPS*, pp. 123–130, 2005.
[47] L. Wu, B. Liu, P. Stone, and Q. Liu, "Firefly neural architecture descent: a general approach for growing neural networks," in *NeurIPS*, 2020.
[48] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.
[49] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *ICLR*, 2019.
[50] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE CVPR*, pp. 7132–7141, 2018.
[51] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *ICML*, pp. 933–941, 2017.
[52] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *ECCV*, pp. 241–257, 2018.
[53] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *IEEE CVPR*, pp. 831–839, 2019.