Feature Expansion and enhanced Compression for Class Incremental Learning

Quentin Ferdinand, Gilles Le Chenadec, Benoit Clement, Panagiotis Papadakis, Quentin Oliveau

Feature Expansion and enhanced Compression for Class Incremental Learning

Quentin Ferdinand[a,b,*], Benoit Clement[b,d], Panagiotis Papadakis[c], Quentin Oliveau[a], Gilles Le Chenadec[b]

[a]*Naval Group, France*
[b]*ENSTA Bretagne, Lab-STICC UMR 6285, Brest, France*
[c]*IMT Atlantique, Lab-STICC UMR 6285, team RAMBO, Brest, France*
[d]*CROSSING IRL 2010, Adelaide, Australia*

Abstract

Class incremental learning consists in training discriminative models to classify an increasing number of classes over time. However, doing so using only the newly added class data leads to the known problem of catastrophic forgetting of the previous classes. Recently, dynamic deep learning architectures have been shown to exhibit a better stability-plasticity trade-off by dynamically adding new feature extractors to the model in order to learn new classes followed by a compression step to scale the model back to its original size, thus avoiding a growing number of parameters. In this context, we propose a new algorithm that enhances the compression of previous class knowledge by cutting and mixing patches of previous class samples with the new images during compression using our Rehearsal-CutMix method. We show that this new data augmentation reduces catastrophic forgetting by specifically targeting past class information and improving its compression. Extensive experiments performed on the CIFAR and ImageNet datasets under diverse incremental learning evaluation protocols demonstrate that our approach consistently outperforms the state-of-the-art . The code will be made available upon publication of our work[1].

Keywords: Class incremental learning, Catastrophic forgetting, Rehearsal memory, Knowledge distillation, CutMix, Dynamic networks, Convolutional neural networks, Deep learning

1. Introduction

In recent years, deep learning has undergone a remarkable evolution, demonstrating impressive achievements in various domains [1, 2, 3]. In particular, in the field of visual classification, convolutional neural networks have been shown to attain and even exceed human performance [4, 5, 6, 7]. Despite reaching human-like performance on specific vision tasks, however, these models encounter limitations in their ability to continually learn and adapt to novel concepts [8], a capability inherent in human cognition. In fact, this inability to adapt incrementally poses significant challenges in real-world applications like face recognition[9, 10], robotics[11, 12] up to autonomous driving [13]. The field of class incremental learning specifically studies the process of incrementally training a classification model on newly acquired data to accommodate new classes or concepts over time. The main challenge in class incremental learning is called catastrophic forgetting and refers to the tendency of models trained incrementally to forget previously learned classes when learning new ones, leading to a degradation in performance on earlier tasks or classes. Constituting an open research problem, this has led to a plurality of approaches that seek to alleviate its effects. Among them, memory rehearsal [14, 15, 16, 17, 18, 19, 20, 21] stands out as one of the most widely adopted methods. This technique involves maintaining a fixed-size memory containing a few exemplars from previously learned classes. During model finetuning, this memory is incorporated alongside the new class data to retain a subset of samples from prior classes within the dataset, thus preventing complete forgetting. This rehearsal strategy leads to an initial performance gain that many approaches [22, 15, 19, 23, 18, 24] seek to extend by combining it with knowledge distillation so as to further reduce forgetting. Knowledge distillation consists in transferring "knowledge" from one model to another, which in incremental learning translates to transferring knowledge from the previous model to the one being finetuned on new data. This enables the training model to preserve knowledge of previous classes while adapting to new ones, thereby mitigating forgetting.

While effective at reducing forgetting these methods also reduce the adaptation of the model and therefore induce a stability/plasticity trade-off [25, 20] dilemma between retaining past classes and adapting to new classes. Recently, a new paradigm [20, 21, 26] has emerged that proposes to freeze previous feature extractors and dynamically expand the feature space of the model by training new extractors during each incremental step. This novel technique has demonstrated a superior trade-off between stability and plasticity compared to conventional incremental methods [20]. This performance gain, however, comes at the cost of an increasing number of network parameters over time.

To address this issue, a plausible solution involves compressing the model back to its initial size upon the completion of each

*corresponding author. Email: quentin.ferdinand@ensta-bretagne.org. This research was supported by Naval Group and the National Association of Research and Technology (ANRT).

[1]https://github.com/QFerdi/FECIL

incremental step [21]. This compression training step is done on the incremental dataset that is biased towards new classes and therefore impedes the compression of previous class knowledge. To address this limitation, we propose a novel method termed as **Rehearsal-CutMix** that is shown to enhance the compression step. Specifically, the key contributions of this work can be summarized as follows:

- We present a novel incremental algorithm based on the expansion/compression paradigm for incremental learning.
- We introduce a new hybrid data augmentation technique that combines the widely adopted CutMix augmentation [27] with the rehearsal memory employed in incremental learning as illustrated in 1.
- Through comprehensive experimentation on multiple datasets and popular incremental evaluation protocols we demonstrate the effectiveness of this new rehearsal-CutMix augmentation when used during the compression step of our algorithm. Furthermore, our method attains the best performance against the state-of-the-art on all evaluation datasets.

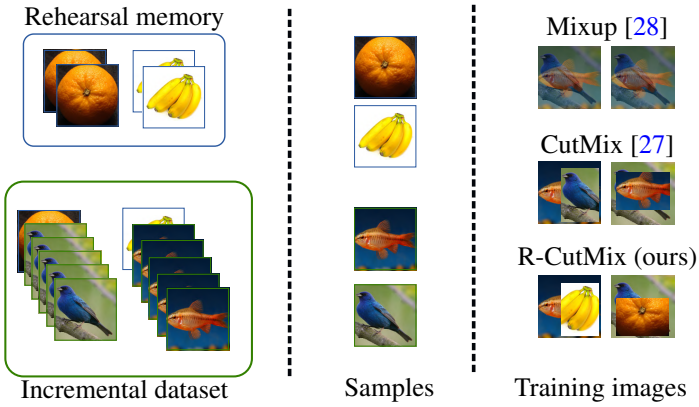

Figure 1: Overview of the differences between Mixup [28], CutMix [27], and our Rehearsal-CutMix procedure. Our method specifically samples one image from the incremental dataset containing mostly new classes and one image from the rehearsal memory containing only previous classes before mixing them together for training.

This work follows up on our previous study [24] that leveraged contrastive learning methods to improve the quality of the features learned incrementally. Going beyond that earlier work, we take advantage of the improved stability/plasticity trade-off of dynamic models and enhance the necessary compression step with a new data augmentation based on the popular CutMix [27] technique, recognized for its proven efficacy in enhancing model generalization.

The remainder of the article is organized as follows. In section 2 we conduct a comprehensive review of related works in the class incremental field, providing some context to the different components of our method. In section 3 we then describe in detail our methodology and the integration of the Rehearsal-CutMix data augmentation technique to the compression step. The performance evaluation of our approach is unfolded in section 4, where we present experimental results and comparative analyses across multiple datasets and methods, demonstrating the superiority of our framework. Finally, section 5 concludes our work and identifies directions for future investigations.

2. Related works

In this section, we first review the main approaches used to alleviate catastrophic forgetting during incremental training. Subsequently, we describe the emerging paradigm that dynamically expands the trained network and has been shown to lead to better plasticity/stability trade-offs during incremental steps. Finally, a description of the mixup data-augmentation strategy and its importance in the context of incremental learning is explored.

2.1. Conventional methods

Class incremental learning methods aim to train models able to learn new classes incrementally in a way that controls the forgetting of previous ones. We refer the interested reader to the following surveys [29, 30] for a detailed presentation of the state-of-the-art , allowing us to focus on the main components in the following paragraphs.

In the past years, numerous methods have been proposed to alleviate catastrophic forgetting during incremental learning. Notably, knowledge distillation, initially introduced as a means to transfer knowledge from a larger teacher model to a smaller student model [31], found its early application in incremental learning through the method *Learning without Forgetting*(LwF) [32]. By using the probabilistic model output of the previous incremental step as a soft target for the model being trained during an incremental step, Li et al. were able to transfer knowledge from previous classes into the model learning new classes, thus mitigating forgetting.

Subsequent developments instigated by the *iCarL* method [14], expanded on this basis by combining it with a rehearsal memory mechanism to further reduce forgetting [22, 15, 18]. This procedure consists in storing the most representative exemplars of each seen class within a fixed-size memory in order to be able to jointly replay them with new data during incremental steps.

This approach posed a novel challenge by inducing an imbalance in the incremental dataset towards new classes that is known to be particularly problematic for the training of deep neural networks [33] and was shown to bias classification towards the most represented classes of the dataset [29]. In *EtEIL* [22], the authors proposed fine-tuning the classification layer on a balanced subset of the training data while *DER* [20] trains a new classification layer from scratch on this subset. The alternative method of rescaling weights of new and past classes within the biased classification layer has also been explored. In *Bic* [15], authors apply an affine rescaling of the weights and learn its parameters on a balanced validation set. Finally, attaining comparable performance, the method *WA* [18] greatly

simplified this rebalancing step by simply rescaling the classification layer based on the norm ratio of past and new class weights.

2.2. Dynamic models

An emerging concept in incremental learning advocates freezing the learned weights after each incremental step and adding new ones for adaptation to new classes. Early approaches considered using completely different sets of neurons [34, 35] for each incremental step but were inherently bounded by the network capacity, depleted only after training on a few incremental tasks.

Towards overcoming this limitation, recent works explored the addition of new neurons [35] or even new complete feature extractors incrementally to accommodate new classes [20, 26, 21]. These approaches typically induce a considerable increase in network parameters with each incremental step that some works try to mitigate with complex pruning-based approaches [35, 20, 26]. These approaches typically involve training models with sparsity losses to then be able to prune many useless neurons from the model in order to reduce significantly the number of parameters with a minimal loss of performance. This type of approach mitigates but does not solve completely the issue of the number of required parameters growing with the incremental steps and generally requires specific hyper-parameters depending on the dataset and application to find the right balance between performance and parameter growth.

The recent method *FOSTER* [21], however, considered a concurrent approach making use of the knowledge distillation loss to compress the dynamic network with added neurons back to its original size after each incremental step. This compression necessitates a dedicated training step on the incremental dataset and therefore faces the typical challenges of incremental learning such as forgetting of past class information and training on an imbalanced dataset.

To alleviate the previous shortcoming, in this work we introduce a novel data augmentation based on the popular *CutMix* augmentation [27] and designed specifically to improve the distillation of previous classes during this compression step.

2.3. Mixup-based data augmentation

Mixup [28] is a data augmentation technique that consists in generating and training on random interpolations between samples. Despite its simplicity this augmentation has been shown to improve significantly the generalization of state-of-the-art neural network architectures. Specifically, training classification models on interpolations between samples leads to smoother decision boundaries between classes therefore improving the generalization of the model.

Recently, many variants of this mixup procedure have been proposed [27, 36, 37] by changing the method used to mix different samples and labels. *CutMix* [27], the main method considered in this paper, combines *Mixup* with the *Cutout* data augmentation [38] by cropping and mixing patches of different images instead of interpolating between them. This method of mixing images together was shown to retain the benefits of both *Cutout* and *Mixup* and further reduce the over-confidence issue that can arise when training deep neural networks.

Manifold *Mixup* [36] on the other hand considers interpolations between hidden representations of the samples to learn class manifolds with less variance to further improve the robustness of models trained in this way. While superior to the base *Mixup* augmentation, Manifold *Mixup* requires two images to go through the feature extractor part of the network before mixing them to obtain the final training sample and therefore induces an increased training time and gpu memory cost when training the model.

3. Proposed Method

In this section, we describe in detail our method called Feature Expansion and enhanced Compression for Incremental Learning (FECIL). A schema representing the overall pipeline is shown in Figure 2 while the details of the expansion and compression stages are explained in sections 3.2 and 3.3 respectively.

3.1. Method overview and problem setting

In the class incremental learning problem that we consider, a model is trained sequentially for T classification tasks, wherein each task incorporates a set of C_{new} new classes that the model is required to classify. For each classification task t, a new dataset $\mathcal{D}_{new} = \{\mathcal{X}, \mathcal{Y}\}$ is considered, where $\mathcal{X}$ and $\mathcal{Y}$ represent a set of labeled samples (images in our case) belonging to C_{new} classes that were previously unknown. Our method makes use of the rehearsal strategy introduced in [14] that stores the most representative samples of each class using Herding sampling [39]. These representative samples are stored in a small fixed-size memory $\mathcal{M}_t$ for future incremental training steps which leads to an augmented training dataset $\mathcal{D}_t = \{D_{new} \bigcup \mathcal{M}_t\}$ containing a part of all C_t classes including those of the previous classes.

Each incremental step is split into two training phases. The first one is the expansion phase where we create and train an expanded network Φ^t_{big} by adding a new feature extractor φ^t_{new} and weights for the new classes in the classification layer $\mathcal{H}^t$ before updating the models' parameters on the new data task. By freezing the previous feature extractor and training a new one, it is ensured that no forgetting of previous features happens in this expansion phase, however the training dataset is still imbalanced towards new classes, therefore the prominent weight alignment (WA) technique [18] is used after this training step to remove the bias from the classification layer.

The resulting dynamic network contains two feature extractors and one classification layer allowing to classify both new and past classes. In order to prevent the number of parameters from growing after each incremental step, this network is then compressed into a more compact network Φ^t that is then saved for the next incremental step. Knowledge distillation is a technique that has witnessed particular success in non-incremental scenarios for this kind of network compression problems [31, 40, 41, 42, 43, 44]. In the context of incremental learning, however, it has been shown by the FOSTER

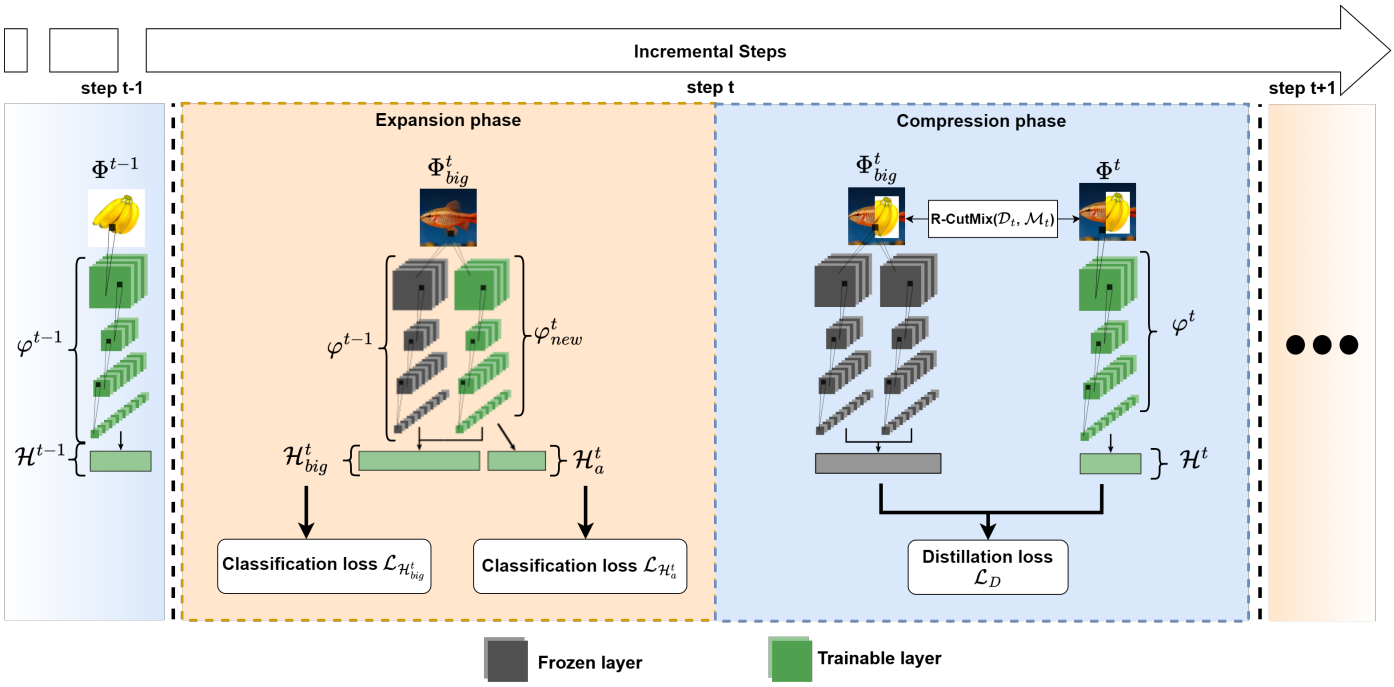

Figure 2: Pipeline of the proposed approach. Each incremental step consists of two training phases, first the expansion phase where we dynamically expand Φ^{t-1} to learn new classes, and second, the phase where we compress the expanded model Φ^t_{big} back to its original size with minimal performance drop using our Rehearsal-CutMix distillation mechanism.

method [21] to struggle with previous classes compression due to the imbalance of the incremental dataset. We therefore introduce a new data-augmentation scheme based on the CutMix method that we call Rehearsal-CutMix, specifically designed to improve the distillation of past classes knowledge in this compression step. Finally, similarly to our expansion step, any bias resulting from the dataset imbalance is removed from the classification layer with the WA technique after this training step.

Upon compression of Φ^t_{big} into Φ^t, Φ^t_{big} is then discarded and only Φ^t is retained and used in the next incremental step, which ensures the model size does not grow with the incremental steps.

3.2. Feature expansion

This phase bears similarities with the one introduced in DER [20]. At incremental task t, like DER, we create the dynamic model Φ^t_{big} that expands the previous feature space with a new feature extractor φ^t_{new} to accommodate new classes. Unlike DER, however, our dynamic model does not require all previous extractors but only two, φ^t_{new} and φ^{t-1} thanks to the compression step we will detail in section 3.3. Both the previous and new feature extractors are then fed into a new classifier $\mathcal{H}^t_{big}$ with C_t (the total number of old and new categories) outputs.

More specifically, given an input image x from the incremental dataset $\mathcal{D}_t$, the feature vector ϕ of the model becomes the concatenation of both feature extractor outputs:

$$\phi = \{\varphi^{t-1}(x), \varphi^t_{new}(x)\} \tag{1}$$

In order to mitigate the forgetting of past classes, the previous feature extractor φ^{t-1} as well as the statistics of its batch normalization layers [45] are frozen for this expansion training step. On the other hand, in order to encourage the model to adapt the extracted features to new classes, the weights of φ^t_{new} are initialized with those of φ^{t-1} but optimized along with the classification layer on the incremental dataset $\mathcal{D}_t$.

The feature vector ϕ is then fed into the dense classification layer $\mathcal{H}^t_{big}$ and softmax is applied to the output logits of this classifier to make the prediction for each class:

$$p_{\mathcal{H}^t_{big}}(y|x) = \text{Softmax}(\mathcal{H}^t_{big}(x))$$

Finally, the weights of the new classifier $\mathcal{H}^t_{big}$ corresponding to the old classes are initiated with those of $\mathcal{H}_{t-1}$ to retain old knowledge while the newly added weights are randomly initialized.

Feature adaptation to new classes. During the expansion step, the model Φ^t_{big} is trained on the incremental dataset $\mathcal{D}_t$ to classify correctly both new and old classes by minimizing a cross entropy loss function :

$$\mathcal{L}_{\mathcal{H}^t_{big}} = -\frac{1}{B}\sum_{i=1}^{B}\log(p_{\mathcal{H}^t_{big}}(y = y_i|x_i)) \tag{2}$$

where B is the size of the batch of images sampled from $\mathcal{D}_t$, x_i is one image of the batch and y_i is its label.

However, training solely with this loss function would lead the new feature extractor φ^t_{new} to learn features that discriminate between past classes. These features are not only redundant with those of the feature extractor φ^{t-1} kept frozen but used by the model, but also tend to over-fit on the memory data, which negatively impacts performance as demonstrated in [20].

For this reason we enforce the model to learn discriminating features only for new classes by employing the auxiliary classifier $\mathcal{H}_a^t$ introduced in DER. The features from φ_{new}^t are fed into $\mathcal{H}_a^t$ made with $C_{new}+1$ outputs in order to classify all new classes and treat all past classes as one category. Initialized randomly, this classifier is then trained with a cross-entropy loss $\mathcal{L}_{\mathcal{H}_a^t}$:

$$\mathcal{L}_{\mathcal{H}_a^t} = -\frac{1}{B}\sum_{i=1}^{B}\log(p_{\mathcal{H}_a^t}(y=\tilde{y}_i|x_i)) \tag{3}$$

where $\tilde{y}_i$ is the modified one-hot target vector of size $C_{new}+1$. This classifier's only purpose is to moderate the training of φ_{new}^t and is therefore discarded at the end of the training step.

Our total loss for the expansion training phase therefore becomes the following linear combination of the previously explained losses:

$$\mathcal{L}_{exp} = \mathcal{L}_{\mathcal{H}_{big}^t} + \mathcal{L}_{\mathcal{H}_a^t}$$

3.3. Feature compression

Upon completion of the feature expansion phase, our model Φ_{big}^t composed of two feature extractors obtains excellent performances, as will be demonstrated in section 4. This performance gain comes at the cost of an increased number of parameters that we solve by compressing Φ_{big}^t back to its original size while controlling the loss of information.

Specifically, a model Φ^t composed of only one feature extractor φ^t and a classifier $\mathcal{H}^t$ is initialized with Φ^{t-1} and trained on $\mathcal{D}_t$ with the standard knowledge distillation loss using Φ_{big}^t probability distributions as a soft targets:

$$\begin{aligned} q_c^{\mathcal{H}^t}(x) &= \frac{e^{o_c(x)/\tau}}{\sum_{i=1}^{C^t} e^{o_i(x)/\tau}} \\ \mathcal{L}_D(x) &= \sum_{c=1}^{C_t} -q_c^{\mathcal{H}_{big}^t}(x)\ \log(q_c^{\mathcal{H}^t}(x)) \end{aligned} \tag{4}$$

with $q_c^{\mathcal{H}^t}(x)$ the softened softmax probability obtained from output node o_c of the model, τ a temperature hyper-parameter, and $q_c^{\mathcal{H}_{big}^t}(x)$ the equivalent softened softmax probability but obtained from the outputs of the big model Φ_{big}^t.

Due to the imbalance of $\mathcal{D}_t$, however, such a compression scheme performs better in new than past classes [21], leading to a compressed model with poor performances on previous classes. We therefore devise a new method called Rehearsal-CutMix to improve the distillation of past classes.

Rehearsal-CutMix. CutMix [27] is a data augmentation strategy building upon the famous Mixup augmentation [28] by mixing patches of different samples to generate synthetic training samples. This augmentation has been shown to enhance the model generalization and robustness and alleviate its overconfidence when making predictions.

Specifically, considering an image $x \in \mathbb{R}^{W\times H\times C}$ with W representing the width, H the height, and C the number of channels (3 for RGB images); and its label y from the sampled mini-batch in onehot format, CutMix generates a training sample $(\tilde{x},\tilde{y})$ by cropping and replacing patches of one image x_i with patches of another image x_j :

$$\begin{aligned} \tilde{x} &= \mathrm{M}\odot x_i + (\mathbb{1}-\mathrm{M})\odot x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{5}$$

where $\mathrm{M}\in\{0,1\}^{W\times H\times C}$ represents a binary mask, $\mathbb{1}$ a similar mask but filled with ones, and $\odot$ is the element-wise multiplication. The parameter λ controls the ratio of combination between the two images and is sampled for each generated sample $(\tilde{x},\tilde{y})$ from a beta distribution $\lambda \sim \mathrm{Beta}(\alpha,\alpha)$ defined by the hyper-parameter α. In order for M to crop a patch of x_i and replace it with a patch of x_j according to the ratio parameter λ, a bounding box defined by its center $(C_x,\ C_y)$ and its size $(r_w,\ r_h)$ is uniformly sampled in the following manner :

$$\begin{aligned} C_w &\sim \mathrm{Unif}(0,W), \quad & r_w &= W\sqrt{1-\lambda} \\ C_h &\sim \mathrm{Unif}(0,H), \quad & r_h &= H\sqrt{1-\lambda} \end{aligned} \tag{6}$$

Since the area of such a bounding box is $\frac{r_w r_h}{WH} = 1-\lambda$, filling M with 0 inside of this box and 1 elsewhere makes λ control the strength of the combination of both the images (x_i, x_j) and labels (y_i, y_j).

While the original implementation of CutMix considered mixing samples within the same training mini-batch [27], we here propose instead to sample x_i from the training mini-batch taken from the incremental dataset $\mathcal{D}_t$ and x_j randomly sampled from the rehearsal memory $\mathcal{M}_t$. We call this variant Rehearsal-CutMix or R-CutMix in short.

$\mathcal{D}_t$ already contains $\mathcal{M}_t$, however, $\mathcal{D}_t$ is biased towards new classes, resulting in mini-batches sampled from it that contain more images from new than past classes. By taking x_j from a second mini-batch sampled solely from $\mathcal{M}_t$, we ensure the generated images are either a new class mixed with a previous class, or two previous classes mixed together which gives three main benefits for our compression step.

On one hand the benefits of the original CutMix augmentation are retained, training on mixed images partly cropped improves the model generalization and robustness [28, 38, 27]. On the other hand, specifically in our incremental context, R-CutMix rebalances the training labels and the overall dataset towards past classes which reduces the bias that is learned by the classification layer. Most importantly, it ensures that most training samples contain information about both new and past classes which allows the knowledge distillation loss to transfer better the overall knowledge of the model and especially at the boundaries between each old and new class.

4. Experiments

This section provides an extensive evaluation of our approach within three datasets that are widely used in class incremental

Methods	CIFAR-100 B0						CIFAR-100 B50			
	5 steps		10 steps		20 steps		5 steps		10 steps	
	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last
Joint Training	80.41	-	81.49	-	81.74	-	79.89	-	79.91	-
iCaRL [14]	71.14	-	65.27	50.74	61.20	43.75	65.06	-	58.59	-
BiC [15]	73.10	-	68.80	53.54	66.48	47.02	66.62	-	60.25	-
WA [18]	72.81	-	69.46	53.78	67.33	47.31	64.01	-	57.86	-
DER [20]	75.55	-	74.64	64.35	73.98	62.55	72.60	-	72.45	-
DER w/o P [20]	76.80	-	75.36	65.22	74.09	62.48	73.21	-	72.81	-
DyTox [46]	-	-	73.66	60.67	72.27	56.32	-	-	-	-
DyTox+ [46]	-	-	75.54	62.06	75.04	60.03	-	-	-	-
FOSTER B4 [21]	78.01	68.96	75.74	62.03	73.04	57.3	74.14	66.14	69.82	58.9
FOSTER [21]	77.47	68.68	75.19	62.08	72.36	57.67	74.61	66.36	70.21	59.07
FECIL B4 (ours)	79.16	71.52	78.48	67.31	76.12	61.51	75.48	68.23	71.18	61.3
FECIL (ours)	78.32	69.35	77.49	66.1	75.0	60.41	73.99	66.53	70.07	60.63

Table 1: Results on the CIFAR-100 B0 and B50 benchmarks averaged over three different class orders. The Best and second best methods are displayed respectively with gray and light gray background color. FOSTER was run with the official released implementation, changing only the backbone architecture to a 18-layers ResNet (which increases performance) for fair comparison with other methods. Dytox [46] performances are grayed out because it is the only method that cannot be run with a 18-layer ResNet and instead uses a Vision Transformer.

learning, namely, CIFAR-100, ImageNet-100, and ImageNet-1000. We first present the details of our implementation and the hyper-parameters used for our experiments and then compare the performance of our method against several state-of-the-art algorithms (see sections 4.2 and 4.3) using popular evaluation protocols and incremental datasets. Finally, in section 4.4, we conduct ablation studies so as to assess the contribution of each component of our method.

4.1. Experimental Setup and Implementation Details

Datasets and protocols. The CIFAR-100 dataset [47] is composed of 32x32 pixel color images representing 100 classes with 500 training images and 100 evaluation images for each class. The ImageNet-1000 dataset [4] on the other hand is a large scale dataset containing 1.2 million training images and 50,000 validation images representing 1000 different categories. Lastly, the ImageNet-100 dataset consists of a subset of the large scale ImageNet-1000 dataset containing only 100 randomly sampled classes. Following standard practice, we validate the proposed method on CIFAR-100 and ImageNet-100 with two widely used [20, 21, 46] protocols :

B0 (base 0) : In this protocol, all the classes of the dataset are separated equally in 5, 10 or 20 incremental steps and the model uses a rehearsal memory of 2000 exemplars.

B50 (base 50) : The model is first trained on 50 initial classes, the remaining 50 classes are then separated equally in 5 or 10 incremental steps and the model is trained with a memory of 20 samples per class in each incremental step.

We compare the top-1 accuracy obtained after the last incremental step as well as the average incremental accuracy as defined in [14] for all methods trained with these protocols.

For ImageNet-1000, we validate our approach with the protocol known as **ImageNet-1000 B0** [14, 20, 21] that trains the model on all 1000 classes in 10 incremental steps of 100 classes using a memory size of 20000 samples. Moreover, following [21], we report the top-1 and top-5 last step and average incremental accuracies on this protocol.

Implementation details. Our method is implemented in PyTorch [48] with the framework PyCIL [49]. Following [20, 46] we chose the 18-layers ResNet [5] backbone architecture to evaluate our model on all the considered datasets.

We note that some previous works [14, 15, 18, 21] instead used a modified 32-layers ResNet [14] for the CIFAR-100 dataset which has been shown in [20] to underestimate their performance because it cannot achieve competitive results on CIFAR100 compared with the standard 18-layers ResNet [20]. Authors from [20], re-implemented or used the officially released implementations when possible for all the baseline methods [14, 15, 18] to obtain their performances with a standard 18-layers ResNet architecture. We therefore follow [46] and report these published results in our tables, and use the official FOSTER implementation to compare all methods using a the same 18-layers ResNet model.

Following standard practice, we used the optimizer SGD with a momentum of 0.9, a weight decay of 0.0005 and use a batch size of 128 for CIFAR-100 and 256 for ImageNet. During each training step, our models where trained for 200 epochs, with a learning rate starting at 0.1 and gradually decaying to 0 with a cosine annealing schedule [50]. Following [21], the data augmentation applied to training images consists in random cropping, horizontal flip, AutoAugment [51], and normalization. For the compression phase we add our rehearsal-CutMix augmentation, set the beta distribution parameter α to 0.2, and set τ to 2 for the distillation loss. Finally, the exemplars stored in rehearsal memory are selected via the Herding selection strategy [39] following previous works [14].

4.2. Evaluation on CIFAR-100

In order to properly evaluate the effectiveness of our method we compare it to several state-of-the-art methods including

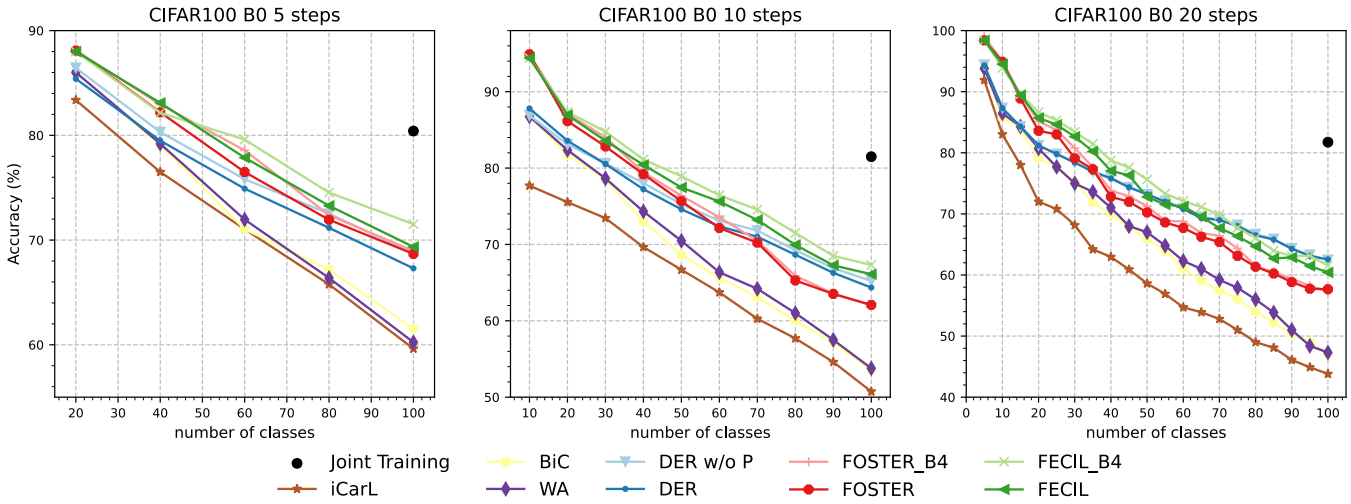

Figure 3: Performance evolution on CIFAR-100. The top-1 accuracy (%) is reported after each incremental step. Left is evaluated with 5 steps, middle with 10 steps, and right with 20 steps.

other dynamic network based approaches such as DER [20], FOSTER [21], and Dytox [46], and several methods that do not rely on dynamic networks such as iCarL [14], BiC [15], and WA [18].

DER uses a pruning mechanism to ensure the number of parameters does not grow too much and DER w/o P denotes the performance obtained without the pruning part. Our method and FOSTER, on the other hand, use a compression step instead and we therefore denote by FECIL B4 and FOSTER B4 the performances obtained before the compression step. Finally, the method Dytox is the only method that does not use a 18-layers ResNet model and instead leverage a transformer architecture with a similar number of parameters that allows them to control the expansion of parameters during the incremental training.

Table 1 summarizes the main results obtained on the CIFAR-100 dataset. As can be seen in this table our method surpasses significantly the other methods in most experimental settings. When compared to the other expansion and compression-based method FOSTER, our method generally achieves both a higher average incremental accuracy and accuracy after the last step, which demonstrates the effectiveness of our R-mixup compression step. Particularly under the incremental setting B0 10 steps, our FECIL model surpasses FOSTER by 2.3% points of accuracy in average and up to 4% in the last incremental step, demonstrating that our R-CutMix compression scales better with the number of incremental steps.

For the B0 benchmark, as can be seen in the figure 3 our method consistently reaches a higher accuracy than FOSTER and FOSTER B4 after each incremental steps. In fact, only DER reaches a slightly higher accuracy in the last few incremental steps of the B0 20 steps protocol, which is to be expected because DER does not compress the model after each incremental step and instead considers a growing number of parameters, thus allowing their performance to scale better for very high numbers of incremental steps.

Results obtained on the CIFAR-100 B50 protocol are displayed in the Table 1, where it can be observed that our method still surpass other methods in most cases.

There are two main differences between the B0 and B50 protocols; on the one hand, since half of the dataset is trained initially, remembering the initial classes is much more important for the B50 setting. On the other hand, the rehearsal memory is limited to 20 exemplars per class instead of the total size being limited to 2000 for the B0 protocol. This low number of samples stored in memory is especially challenging for our method as it directly impacts the variability of images sampled by our R-mixup procedure. For this reason our method does not reach the performance of DER in the B50 10steps setting but still outperforms FOSTER, demonstrating the effectiveness of our compression step even with such a limited amount of samples stored in memory. Moreover, with the B50 5steps protocol, the memory grows faster to 2000 samples which positively impacts the performance of our R-CutMix compression and thus allows our method to surpasses both FOSTER and DER.

4.3. Evaluation on ImageNet

Table 2 summarizes the main results obtained for all approaches on the ImageNet-100 and ImageNet-1000 datasets. Similarly to the results obtained on CIFAR-100 with the B0 protocol, both FECIL B4 and FECIL reach higher average and last step accuracy than FOSTER B4 and FOSTER demonstrating the effectiveness of our R-CutMix compression across several datasets.

On the ImageNet-100 dataset it can be seen that our method consistently reaches significantly higher performance than most other methods on both datasets. Specifically, FECIL B4 and FECIL outperform FOSTER B4 and FOSTER by 2.24% and 1.85% in terms of average top-1 accuracy and 2.17% and 1.71% in terms of last step top-1 accuracy.

Methods	ImageNet-100 10 steps					ImageNet-1000 10 steps				
	#Params	top-1		top-5		#Params	top-1		top-5	
		Avg	Last	Avg	Last		Avg	Last	Avg	Last
Joint Training	11.22	-	81.20	-	95.1	11.68	-	-	-	89.27
iCaRL [14]	11.22	-	-	83.6	63.8	11.68	38.4	22.7	63.7	44.0
BiC [15]	11.22	-	-	90.6	84.4	11.68	62.73	50.1	83.80	72.70
WA [18]	11.22	-	-	91.0	84.1	11.68	65.67	55.60	86.6	81.1
DER w/o P [20]	112.27	77.18	66.70	93.23	87.52	116.89	68.84	60.16	88.17	82.86
DER [20]	-	76.12	66.06	92.79	88.38	-	66.73	58.62	87.08	81.89
DyTox [46]	11.01	77.15	69.10	92.04	87.98	11.36	71.29	63.34	88.59	84.49
FOSTER B4 [21]	22.44	76.54	67.08	93.12	88.89	23.36	68.34	58.53	89.18	81.77
FOSTER [21]	11.22	76.22	66.70	93.08	88.56	11.68	68.29	58.16	89.22	81.49
FECIL B4 (ours)	22.44	78.78	69.25	95.34	91.31	23.36	70.05	61.23	91.37	83.19
FECIL (ours)	11.22	78.07	68.41	94.76	90.28	11.68	69.11	60.4	90.18	81.72

Table 2: Results on the ImageNet-100 and ImageNet-1000 B0 benchmarks. We report the average and last top-1 and top-5 accuracy obtained and display the Best and second best methods respectively with gray and light gray background color."#Params" corresponds to the number of parameters used by the model at the end of the incremental training (in millions).

Finally, On the Imagenet-1000 dataset it can be observed that our method performs better only in terms of top-5 accuracy. In fact, Dytox reaches significantly higher top-1 performance, however our method reaches higher average and last step top-1 accuracy than DER w/o P while using approximately 10 times less parameters due to our compression step.

4.4. Detailed analysis of the method

We further evaluate our method and the contribution of each specific component by conducting a thorough ablation analysis of our method. These ablation experiments were conducted on the 10 steps CIFAR-100 B0 benchmark following standard practice [20].

Specifically, our expansion training step is very similar to the one exhaustively studied in DER[20] apart from the fact that it is followed by a compression step. We therefore focus our analysis on the effectiveness of the different components of our compression step. Specifically, we first remove completely our R-CutMix augmentation and perform a naive distillation-based compression after each incremental step. We then compare the effect of the addition of both the standard Mixup [28] and CutMix [27] augmentations. Finally, we replace these augmentations with the rehearsal-based variant used in our method and explained in detail in section 3.3.

As can be seen in Table 3, when the standard mixup augmentation is used, the performance attained by our FECIL model after the last incremental step only goes from 61.59% to 61.81% while going up to 62.98% with the rehearsal variant. This demonstrates the benefits of training with mixed images representing samples partly from old and new classes. Furthermore, while even the regular CutMix augmentation increases performance significantly, the gap between CutMix and R-CutMix is much larger than between Mixup and R-Mixup. In fact, while replacing Mixup with R-Mixup improves accuracy reached by our compressed model by 1.17%, replacing CutMix with R-CutMix improves the accuracy of the compressed model by 2.88%.

Computational overhead. We further evaluate the overhead induced by our R-CutMix procedure by comparing the average time taken by each ablation to accomplish 1 epoch of the compression step in the table 3. These times are normalized so that the fully ablated method that took 6.55 seconds on our computer corresponds to $1x$ and the increase in percentage is noted for the different ablations. It can be observed that the training time overhead induced by the regular mixup and CutMix methods is negligible, however, it is not the case for the rehearsal variants. This increase comes from the fact that for every mini-batch sampled from the dataset a second one is sampled from the rehearsal memory. These two batches, however, are mixed and merged back into a singular mini-batch before going into the model. This prevents any impact on the time needed for the forward and backward pass and therefore explains why the training time overhead remains limited.

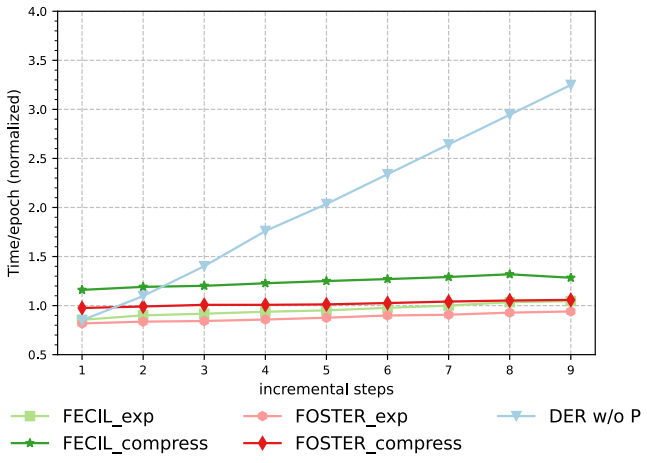

Figure 4: Evolution of the time necessary for an epoch during 10 incremental steps on the CIFAR-100 dataset. FOSTER_exp and FECIL_exp represent the time per epoch during the expansion phase while FOSTER_compress and FECIL_compress illustrate the time per epoch of the compression step.

	Mixup	CutMix	Rehearsal	FECIL B4 Avg	FECIL B4 Last	FECIL Avg	FECIL Last	Avg Time/epoch
Ablations				76.12	63.55	75.07	61.59	1x
	✓			76.89	64.53	75.01	61.81	1.01x
		✓		77.83	66.04	76.19	63.22	1.07x
	✓		✓	77.48	64.47	76.35	62.98	1.22x
		✓	✓	78.49	67.31	77.49	66.1	1.24x

Table 3: Ablation of different key components of our method. The average and last step accuracy are reported for each ablation. We also normalize and report the average time taken by one epoch during the compression step. All experiments were done on the CIFAR-100 B0 10 steps benchmark.

Finally, we compare the training time overhead of our method against FOSTER and DER w/o pruning with the CIFAR-100 B0 10 step protocol in figure 4. The times reported are kept normalized in a similar manner than in table 3 for better readability. As can be observed, since DER w/o pruning does not use a compression step, the time necessary for an epoch drastically increases over the incremental steps. Conversely, our approach introduces a minor overhead in contrast to FOSTER, but one that remains constant over time. This ensures that the epoch duration of FECIL remains stable throughout incremental training, thus preserving scalability even with a high number of incremental steps.

5. Conclusion

In this work, we introduce FECIL, a novel two-stage training procedure for class incremental learning. Our method first proceeds by expanding the features of the model in order to accommodate new classes without forgetting past ones; It then compresses it back to its original size in order to keep the model size fixed over the course of the entire incremental training process. Specifically, we introduce a Rehearsal-CutMix data augmentation that mixes training images of new classes with images from the rehearsal memory so as to greatly improve the distillation of past classes' information during the compression step. Extensive experiments were performed on three major incremental datasets and a variety of evaluation protocols where our method consistently outperformed other state-of-the-art methods.

While the experiments demonstrated the effectiveness of our compression step, its performance remains bounded by the accuracy reached in the expansion step. As this expansion step is done on an imbalanced dataset, we believe there is still room for further improvements. For example, studying the addition of other Mixup or CutMix-based data augmentations specifically designed to reduce the bias learned in this expansion step could be a promising research direction.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (2015) 436–444.

[2] Y. Bengio, I. Goodfellow, A. Courville, Deep learning, volume 1, MIT press Cambridge, MA, USA, 2017.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (2015) 211–252.

[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[6] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI conference on artificial intelligence, volume 31, 2017.

[7] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114.

[8] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, T. E. Boult, Toward open set recognition, IEEE transactions on pattern analysis and machine intelligence 35 (2012) 1757–1772.

[9] S. Ozawa, S. L. Toh, S. Abe, S. Pang, N. Kasabov, Incremental learning of feature space and classifier for face recognition, Neural Networks 18 (2005) 575–584.

[10] S. Madhavan, N. Kumar, Incremental methods in face recognition: a survey, Artificial Intelligence Review 54 (2021) 253–303.

[11] S. Thrun, T. M. Mitchell, Lifelong robot learning, Robotics and autonomous systems 15 (1995) 25–46.

[12] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. Díaz-Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, Information fusion 58 (2020) 52–68.

[13] J. M. Pierre, Incremental lifelong deep learning for autonomous vehicles, in: 2018 21st international conference on intelligent transportation systems (ITSC), IEEE, 2018, pp. 3949–3954.

[14] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[15] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 374–382.

[16] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, R. Chellappa, Learning without memorizing, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 5138–5146.

[17] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, Q. Sun, Mnemonics training: Multi-class incremental learning without forgetting, in: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition, 2020, pp. 12245–12254.

[18] B. Zhao, X. Xiao, G. Gan, B. Zhang, S.-T. Xia, Maintaining discrimination and fairness in class incremental learning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 13208–13217.

[19] A. Douillard, M. Cord, C. Ollion, T. Robert, E. Valle, Podnet: Pooled outputs distillation for small-tasks incremental learning, in: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XX 16, Springer, 2020, pp. 86–102.

[20] S. Yan, J. Xie, X. He, Der: Dynamically expandable representation for class incremental learning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 3014–3023.

[21] F.-Y. Wang, D.-W. Zhou, H.-J. Ye, D.-C. Zhan, Foster: Feature boosting and compression for class-incremental learning, in: European conference on computer vision, Springer, 2022, pp. 398–414.

[22] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 233–248.
[23] S. Hou, X. Pan, C. C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 831–839.
[24] Q. Ferdinand, B. Clement, Q. Oliveau, G. Le Chenadec, P. Papadakis, Attenuating catastrophic forgetting by joint contrastive and incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3782–3789.
[25] S. Grossberg, Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world, Neural networks 37 (2013) 1–47.
[26] Z. Li, C. Zhong, S. Liu, R. Wang, W.-S. Zheng, Preserving earlier knowledge in continual learning with the help of all previous feature extractors, arXiv preprint arXiv:2104.13614 (2021).
[27] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, Y. Yoo, Cutmix: Regularization strategy to train strong classifiers with localizable features, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.
[28] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412 (2017).
[29] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, J. Van De Weijer, Class-incremental learning: survey and performance evaluation on image classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (2022) 5513–5533.
[30] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, Z. Liu, Deep class-incremental learning: A survey, arXiv preprint arXiv:2302.03648 (2023).
[31] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).
[32] Z. Li, D. Hoiem, Learning without forgetting, IEEE transactions on pattern analysis and machine intelligence 40 (2017) 2935–2947.
[33] M. Buda, A. Maki, M. A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, Neural networks 106 (2018) 249–259.
[34] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra, Pathnet: Evolution channels gradient descent in super neural networks, arXiv preprint arXiv:1701.08734 (2017).
[35] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, C.-S. Chen, Compacting, picking and growing for unforgetting continual learning, Advances in neural information processing systems 32 (2019).
[36] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, Y. Bengio, Manifold mixup: Better representations by interpolating hidden states, in: International conference on machine learning, PMLR, 2019, pp. 6438–6447.
[37] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, D.-C. Juan, Remix: rebalanced mixup, in: Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16, Springer, 2020, pp. 95–110.
[38] T. DeVries, G. W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv preprint arXiv:1708.04552 (2017).
[39] M. Welling, Herding dynamical weights to learn, in: Proceedings of the 26th annual international conference on machine learning, 2009, pp. 1121–1128.
[40] F. Tung, G. Mori, Similarity-preserving knowledge distillation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1365–1374.
[41] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16, Springer, 2020, pp. 776–794.
[42] J. Gou, B. Yu, S. J. Maybank, D. Tao, Knowledge distillation: A survey, International Journal of Computer Vision 129 (2021) 1789–1819.
[43] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, A. Kolesnikov, Knowledge distillation: A good teacher is patient and consistent, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10925–10934.
[44] G. Xu, Z. Liu, X. Li, C. C. Loy, Knowledge distillation meets self-supervision, in: European conference on computer vision, Springer, 2020, pp. 588–604.
[45] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, pmlr, 2015, pp. 448–456.
[46] A. Douillard, A. Ramé, G. Couairon, M. Cord, Dytox: Transformers for continual learning with dynamic token expansion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 9285–9295.
[47] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).
[48] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).
[49] D.-W. Zhou, F.-Y. Wang, H.-J. Ye, D.-C. Zhan, Pycil: A python toolbox for class-incremental learning, 2023.
[50] I. Loshchilov, F. Hutter, Sgdr: Stochastic gradient descent with warm restarts, arXiv preprint arXiv:1608.03983 (2016).
[51] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q. V. Le, Autoaugment: Learning augmentation policies from data, arXiv preprint arXiv:1805.09501 (2018).