S3C: Self-Supervised Stochastic Classifiers for Few-Shot Class-Incremental Learning

Jayateja Kalla and Soma Biswas

Department of Electrical Engineering,
Indian Institute of Science, Bangalore, India.
{jayatejak, somabiswas}@iisc.ac.in

Abstract. Few-shot class-incremental learning (FSCIL) aims to learn progressively about new classes with very few labeled samples, without forgetting the knowledge of already learnt classes. FSCIL suffers from two major challenges: (i) *over-fitting* on the new classes due to limited amount of data, (ii) *catastrophically forgetting* about the old classes due to unavailability of data from these classes in the incremental stages. In this work, we propose a self-supervised stochastic classifier (S3C)[1] to counter both these challenges in FSCIL. The stochasticity of the classifier weights (or class prototypes) not only mitigates the adverse effect of absence of large number of samples of the new classes, but also the absence of samples from previously learnt classes during the incremental steps. This is complemented by the self-supervision component, which helps to learn features from the base classes which generalize well to unseen classes that are encountered in future, thus reducing catastrophic forgetting. Extensive evaluation on three benchmark datasets using multiple evaluation metrics show the effectiveness of the proposed framework. We also experiment on two additional realistic scenarios of FSCIL, namely where the number of annotated data available for each of the new classes can be different, and also where the number of base classes is much lesser, and show that the proposed S3C performs significantly better than the state-of-the-art for all these challenging scenarios.

Keywords: few-shot class-incremental learning, stochastic classifiers, self-supervised learning

1 Introduction

In recent years, Deep Neural Networks (DNN) have shown significant performance improvement on various computer vision applications [19, 27, 29]. Usually, the DNN models require enormous amount of annotated data from all the classes of interest to be available for training. In real-world, since data from different classes may become available at different instants of time, we want the model to learn about the new classes incrementally without forgetting about the old classes, which is precisely the task addressed in Class-Incremental Learning

[1] code: https://github.com/JAYATEJAK/S3C

(CIL). CIL approaches are very useful and practical, not only because it is computationally expensive and time-consuming to retrain the model from scratch, but also because data from the previous classes may not be available due to storage and privacy issues.

Since collecting large number of annotated data from all the new classes is also very difficult, recently, the more challenging but realistic few-shot class-incremental learning (FSCIL) is gaining increasing attention, where the new classes have few labeled samples per class [35]. In FSCIL, a model is first learnt using a set of base classes with large number of labeled examples per class. At each incremental step (task), the model has access to a few labeled samples of the new classes and a single prototype for each of the previously learnt classes. The goal is to learn a unified classifier to recognize the old as well as the new classes, without having access to any task labels. This helps the model to quickly learn about the new classes without requiring to collect and annotate large amounts of data for the new classes. FSCIL faces two major challenges, namely overfitting due to limited samples for the new classes, and catastrophic forgetting of the already learnt classes due to absence of old classes data at the incremental steps.

In this work, we propose a novel framework, S3C (**S**elf-**S**upervised **S**tochastic **C**lassifier) to simultaneously address both these challenges in the FSCIL setting. Unlike the standard classifiers, stochastic classifiers (SC) are represented by weight distributions, i.e. a mean and variance vector [24]. Thus, each classifier weight sampled from this distribution is expected to correctly classify the input samples. We show for the first time, that SC learnt for both the base and new classes can significantly reduce the over-fitting problem on the new classes for FSCIL task. It can also arrest the catastrophic forgetting of the previously learnt classes to a certain extent. As is common in most FSCIL approaches [44, 25], we propose to freeze the feature extractor and learn only the SC at each incremental step. In order to compute features from the base classes which generalize to unseen classes, inspired by recent works [22, 47], we use self-supervision along with SC giving our final S3C framework. As expected, this helps to significantly mitigate the effect of catastrophic forgetting, while at the same time retaining the advantage on the new classes. To this end, our contributions are as follows:

1. We propose a novel framework, termed S3C (Self-Supervised Stochastic Classifier) to address the FSCIL task.
2. We show that stochastic classifiers can help to significantly reduce overfitting on the new classes with limited amount of data for FSCIL.
3. We also show that self-supervision with stochastic classifier can be used to better retain the information of the base classes, without hindering the enhanced performance of the stochastic classifiers for the new classes.
4. We set the new state-of-the-art for three benchmark datasets, namely CIFAR100 [18], CUB200 [37] and miniImageNet [44].
5. We also propose and evaluate on two additional, realistic FSCIL settings, namely FSCIL-im (FSCIL-imbalanced) - where the new classes may have different number of samples/class and (ii) FSCIL-lb (FSCIL-less base) - where there are less number of base classes, which further justifies the effectiveness of the proposed S3C framework.

2 Related Works

Here, we provide some pointers to the related work in literature.
Class-Incremental Learning (CIL): The goal of CIL is to learn new classes progressively without any task information. Due to plenty of annotated new class data, mitigating catastrophic forgetting is a challenging problem. LwF [23] proposed to use knowledge distillation [15] to alleviate catastrophic forgetting. iCaRL [31] showed that nearest classifier mean (NCM) using old class exemplars can generate robust classifiers for CIL. EEIL [7] used knowledge distillation to remember old classes and cross-entropy to learn new classes in an end-to-end training. UCIR [16] proposed cosine-based classifiers and used feature-space distillation and inter-class separation margin loss to mitigate catastrophic forgetting. Several state-of-art-works [38, 45, 2, 12, 3] proposed different techniques to address the class imbalance problem in CIL like rescaling scores or balanced fine-tuning of classifiers, etc. Some of the recent works [47, 41, 46] have focused on non-exemplar based methods, with no access to exemplars from the old classes.

Few-Shot Class-Incremental Learning (FSCIL): Recently, there has been a significant focus on the more realistic and challenging FSCIL task, where very few samples per class are available for training at each incremental task. Tao *et al.* [35] proposed this protocol and used neural network gas architecture to preserve the feature topologies of the base and new classes. Mazumder *et al.* [25] proposed to identify unimportant parameters in the model based on their magnitudes and learn only these parameters during the incremental tasks. The works proposed in [9, 8, 1, 10, 48, 21, 32] focus on learning robust manifolds by regularizing feature space representations. The works in [11, 34, 44] used graph-based networks for old classes' knowledge retention. Recently, CEC [44] proposed a meta-learning strategy and achieved state-of-art results for the FSCIL setting.

Self-Supervised Learning (SSL): SSL uses predefined pretext tasks to learn features from unlabeled data. Different pretext tasks have been proposed like image rotations [17], image colourization [20], clustering [6], and solving jigsaw puzzles from image patch permutations [28]. These features can notably improve the performance of downstream tasks like few-shot learning [13], semi-supervised learning [43], to improve the model robustness [14], class imbalance [40], etc. Recently, Lee *et al.* [22] used SSL to improve the performance for supervised classification, by augmenting the original labels using the input transformations. In this work, we show that SSL [22] can be used very effectively for the FSCIL task.

Stochastic Neural Networks: Traditional neural networks cannot model uncertainty well due to their deterministic nature [5]. Stochastic neural networks [26] give robust representations in the form of distributions. Subedar *et al.* [33] proposed uncertainty aware variational layers for activity recognition. Recently, it has been used for person re-identification [42] and unsupervised domain adaptation [24] tasks.

3 Problem Definition and Notations

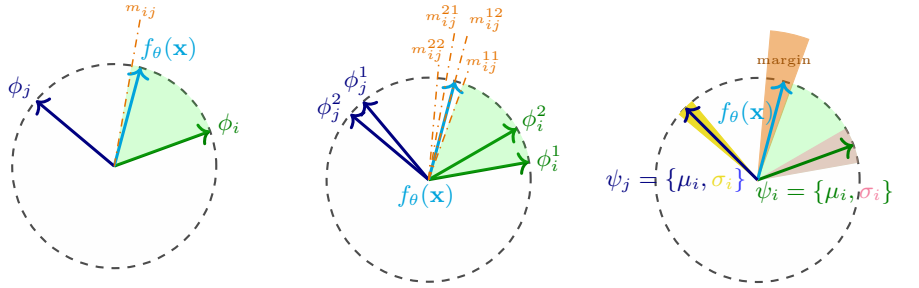
Here, we explain the FSCIL task, which consists of a base task and several incremental stages, and also the notations used in the rest of the paper. In the base task, the goal is to learn a classifier using large number of labeled samples from several base classes. At each incremental step, using a few labeled samples per new class and a single class prototype of the old (previously learnt) classes, the model needs to be updated such that it can classify both the old and the new classes. Let $\mathcal{D}^{(0)}$ denote the base task which contains large number of annotated data from classes $\mathcal{C}^{(0)}$. Let the incremental task data be denoted as $\{\mathcal{D}^{(1)}, .., \mathcal{D}^{(t)}, .., \mathcal{D}^{(\mathcal{T})}\}$, and the corresponding label spaces be denoted as $\mathcal{C}^{(t)}$, where $t = 1, \ldots, \mathcal{T}$. Thus, the model will learn a total of $\mathcal{T}$ tasks incrementally and there is no overlap in the label space between the different tasks, i.e. $C^{(t)} \cap C^{(s)} = \phi$; $(t \neq s)$. Once the model has learned on the data $\mathcal{D}^{(t)}$, it has to perform well on all the classes seen so far i.e $\{C^{(0)} \cup C^{(1)} \cup \cdots \cup C^{(t)}\}$.

4 Proposed Method

Here, we describe the proposed S3C framework for the FSCIL task. In many of the initial FSCIL approaches [35, 25, 8], the main focus was to develop novel techniques for the incremental step to prevent catastrophic forgetting and overfitting. Recently, CEC [44] showed that the base network training has a profound effect on the performance of the incremental tasks. Using appropriate modifications while learning the base classifier can significantly enhance not only the base class accuracies, but also the performance for the incrementally added classes. Even without any fine-tuning during the incremental steps, CEC reports the state-of-the-art results for FSCIL. In the proposed S3C framework, we combine the advantages of both these techniques and propose to not only improve the base classifier training, but also update all the classifiers during the incremental steps. First, we describe the two main modules of S3C, namely Stochastic Classifier and Self-Supervision and then discuss how to integrate them.

Stochastic Classifier: One of the major challenges in FSCIL is the few number of annotated samples that is available per class at each incremental step. This may result in overfitting on the few examples and learning classification boundaries which do not generalize well on the test data. Now, we discuss how stochastic classifiers can be used to mitigate this problem.

In this work, we use cosine similarity between the features and the classifier weights to compute the class score for that particular feature. For a given input image $\mathbf{x}$ from class C_i, let us denote its feature vector as $f_\theta(\mathbf{x})$, where the parameters of the feature extractor f is denoted by θ. Let the classifier weights corresponding to class C_i be denoted as ϕ_i. Then the cosine similarity of the feature with this classifier weight can be computed as $\langle \overline{\phi_i}, \overline{f_\theta(\mathbf{x})} \rangle$, where $\overline{u} = u/||u||_2$ denotes the l_2 normalized vector. Fig. 1(a) shows the normalized feature extractor, and classifier weights for two classes, C_i and C_j. The green shaded area



a). Standard Classifier (ϕ) b). 2-Classifiers/Class (ϕ) c). Stochastic Classifier (ψ)

Fig. 1: Figure shows the classification boundary between two classes in (a) standard classifier, (b) two-classifiers per class and (c) stochastic classifier. The margin in (c) results in more discriminative classification boundaries.

denotes the region where $f_\theta(\mathbf{x})$ will be correctly classified to class C_i, and m_{ij} is the classification boundary between the two classifiers (considering only the upper sector between ϕ_i and ϕ_j).

Now, instead of a single classifier, let us learn two different classifiers for each class (eg. ϕ_i^1 and ϕ_i^2 for class C_i). In Fig. 1 (b), $\{m_{ij}^{11}, m_{ij}^{12}, m_{ij}^{21}, m_{ij}^{22}\}$ are the four classification boundaries for four combination of classifiers. To ensure that the input data is correctly classified using all the classifiers, the feature embedding $f_\theta(\mathbf{x})$ has to move closer to the classifier of its correct class, thus making the samples of a class better clustered and further from samples of other classes. But it is difficult to choose (and compute) how many classifiers should be used. By using a stochastic classifier (Fig. 1 (c)), we can ensure that we have infinite such classifiers around the mean classifier.

Using a stochastic classifier $\psi = \{\mu, \sigma\}$ at the classification head resembles the use of multiple classifiers, where μ and σ denotes the mean and variance of the classifier ψ. For a given input image $\mathbf{x}$, the output score of the stochastic classifiers is proportional to $\langle \hat{\mu}, f_\theta(\mathbf{x}) \rangle$ ($\hat{\mu} = \mu + \mathcal{N}(0,1) \odot \sigma$), where the classifier is sampled from the distribution. This has similarity with feature augmentations which are also commonly used [47]. There are two main advantages of using a stochastic classifier instead of feature augmentations: (1) Instead of using a fixed variance for the features (which has to be manually calculated), the means and variances used in the proposed framework are automatically learnt in an end-to-end manner. (2) The means and variances learnt using the base classes also help to initialize the corresponding parameters for the new classes in a semantically meaningful manner as explained later.

Self-supervision: At the incremental stages, due to presence of few examples from the new classes, in general, most of the FSCIL approaches either fix the feature extractor after learning the base classes [44, 9] or fine-tune it with a very small learning rate [35, 25, 8], so that it does not change significantly. This reduces catastrophic forgetting as well as overfitting. In our work, we fix the fea-

ture extractor after learning the base classes and only fine-tune the classifiers. To make the base feature extractor generalize well to unseen data, we propose to use self-supervision for the base classifier training as well as during the incremental learning stages. Since self-supervised training does not use class labels, more generic features can be learnt, which can generalize well to unseen classes. SSL has been used successfully for several tasks [43, 14, 40, 13, 47], including the standard class-incremental setting [47]. Here, we use the recently proposed SSL approach [22], where image augmentations are used to generate artificial labels, which are used to train the classification layer. For a given input image $\mathbf{x}$, let the augmented versions be denoted as $\widetilde{\mathbf{x}}_r = t_r(\mathbf{x})$, where $\{t_r\}_{r=1}^{M}$ denotes pre-defined transformations. In this work, we use images rotated by $\{0°, 90°, 180°, 270°\}$, i.e. (M=4) as the augmented images. We show that the feature extractor learnt using self-supervision performs very well in the incremental stages. First, we describe the integrated S3C loss which is used in the training process.

Construction of S3C loss: At task t, $C_i^{(s)}$ denotes the i^{th} class in task $s \in \{0, 1, .., t\}$. Then its corresponding stochastic classifier is denoted as $\psi_i^{(s)}$ with mean $\mu_i^{(s)}$ and variance $\sigma_i^{(s)}$. To integrate the stochastic classifiers with self-supervision, for each class, we create four classifier heads corresponding to each of the four rotations as in [22]. In this work, we want to jointly predict the class and its rotation $r = \{0°, 90°, 180°, 270°\}$, thus we denote the final classifiers as $\psi_{i,r}^{(s)}$, with individual means $(\mu_{i,r}^{(s)})$, but with the same class-wise variance $(\sigma_i^{(s)})$. Since the same data is present in different rotations, we enforce that the classifiers for the same class share the same variances, which reduces the number of parameters to be computed. Thus, the joint softmax output of a given sample $\mathbf{x}$ for $C_i^{(s)}$ class at r^{th} rotation is given by

$$\rho_{ir}^{(s)}(\mathbf{x}; \theta, \psi^{(0:t)}) = \frac{exp(\eta \ \langle \ \overline{\hat{\mu}_{ir}^{(s)}}, \overline{f_\theta(\mathbf{x})} \ \rangle)}{\sum_{j=0}^{t} \sum_{k=0}^{|C^{(t)}|} \sum_{l=0}^{M} exp(\eta \ \langle \ \overline{\hat{\mu}_{kl}^{(j)}}, \overline{f_\theta(\mathbf{x})} \ \rangle)} \tag{1}$$

Where $\hat{\mu}_{ir}^{(j)} = \mu_{ir}^{(j)} + \mathcal{N}(0, 1) \odot \sigma_i^{(j)}$ represents the sampled weight from the stochastic classifier $\psi_{ir}^{(j)}$, η is a scaling factor used to control peakiness of the softmax distribution. Finally, the S3C training objective for a training sample $\mathbf{x}$ with label y from task s can be written as

$$\mathcal{L}_{S3C}(\mathbf{x}, y; \theta, \psi^{(0:t)}) = -\frac{1}{M} \sum_{r=1}^{M} \log(\rho_{yr}^{(s)}(\widetilde{\mathbf{x}}_r; \theta, \psi^{(0:t)})) \tag{2}$$

This implies that the input image is transformed using the chosen image transformations (4 rotations in this work) and the loss is combined for that input. Note that the first transformation corresponding to 0° is the identity transformation (i.e. the original data itself). We now describe the base and incremental stage training of the S3C framework (Fig. 2).

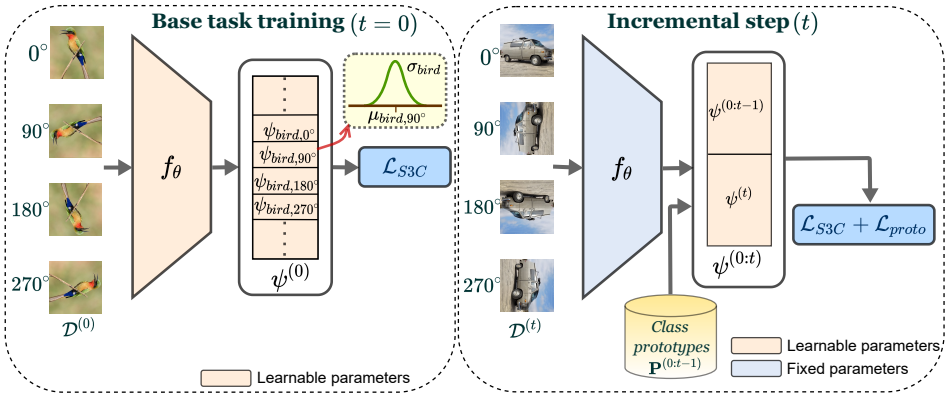


Fig. 2: Illustration of the proposed S3C framework: Left: Base network training, Right: Training at each incremental step.

4.1 Base Network Training of S3C

In FSCIL setting, we assume that we have access to several base classes with sufficient number of annotated data for base training. Given the data from the base classes $C^{(0)}$, we use a base network (ResNet20 for CIFAR100 and ResNet18 for CUB200 and miniImageNet) along with a Graph Attention Network inspired by [44][36]. We train the base network, i.e. the feature extractor with parameters θ and the stochastic classifiers corresponding to the base classes ($\psi^{(0)}$) with S3C objective $\mathcal{L}_{base} = \mathcal{L}_{S3C}(\mathbf{x}, y; \theta, \psi^{(0)})$, with the base training data given by $\{\mathbf{x}, y\} \in \mathcal{D}^{(0)}$. The proposed objective improves the performance of the base classes, in addition to that of the new classes that will be encountered in the incremental stages as we will observe in the experimental evaluation.

4.2 Preparing for the incremental step

After the base classifier training, the training data of the base classes may not be available any longer. This may be due to limited storage capacity, privacy issues, etc. After the first incremental step, we want the unified classifier to perform well on the base as well as on the new classes. For this, to mitigate catastrophic forgetting of the base classes, their class prototypes are stored as is the common practice [35][44]. These stored class prototypes can be treated as class representatives of the base classes and thus can be used for updating the network at the incremental step. The class prototypes are computed by averaging the training features given by the feature extractor ($f_\theta(\cdot)$) for each class. This is done not only at the end of the base training, but after each incremental step as well, i.e. after incremental step t, we store the class prototypes of all the classes that the model has encountered till step t. The class prototype set $\mathbf{P}^{(t)}$ contains the classes prototypes encountered in task t. The class prototype $P_i^{(t)}$ after task

t for i^{th} class is calculated as

$$P_i^{(t)} = \frac{1}{N_i^{(t)}} \sum_{n=1}^{N^{(t)}} \mathbb{I}_{(y_n=i)} \; f_\theta(\mathbf{x}_n) \tag{3}$$

Where $N^{(t)}$ is the number of samples in the dataset $\mathcal{D}^{(t)}$, $N_i^{(t)}$ is number of samples in i^{th} class of task t, and $\{\mathbf{x}_n, y_n\}_{n=1}^{N^{(t)}} \in \mathcal{D}^{(t)}$. The indicator variable $\mathbb{I}_{(y_n=i)}$ will be 1 if the sample belongs to the i^{th} class (i.e. $y_n = i$). Thus, the class prototype set is updated at the end of each task.

4.3 Incremental Step

Here, we will discuss the training process involved in each incremental step. As in [44] [9], we propose to freeze the already learnt feature extractor, since the self-supervision has ensured that it will generalize well to previously unseen classes. This also helps in mitigating the catastrophic forgetting and over-fitting problems. In our work, we propose to update the classifiers of the previous as well as the new classes with the stored class-prototypes and the few examples of the new classes. This will help the model better adapt to the new set of classes. Now, we discuss how to initialize the stochastic classifiers for the new classes.
Initialization of the Stochastic Classifiers of the new classes: For the new classes, we need to initialize the stochastic classifiers before fine-tuning. The means are initialized with the centroid of the features for that class (calculated using the previous model). We initialize the variances of the new classes using that of the most semantically similar class from the base set. Semantic similarity is computed using GloVE embeddings [30] of the base and new class names.

Fine-tuning the classifiers: With this initialization, we fine-tune the classifiers of the new as well as the previous classes using the few labeled examples of the new classes and the stored class-prototypes of the previous classes. Let $q \in \mathbf{P}^{(0:t-1)}$ be a prototype from any old class, then the joint softmax output of the stochastic classifier for i^{th} class and r^{th} rotation (task s) is

$$\zeta_{ir}^{(s)}(q; \psi^{(0:t)}) = \frac{exp(\eta \; \langle \; \overline{\hat{\mu}_{ir}^{(s)}}, \overline{q} \; \rangle)}{\sum_{j=0}^{t} \sum_{k=0}^{|C^{(t)}|} \sum_{l=0}^{M} exp(\eta \; \langle \; \overline{\hat{\mu}_{kl}^{(j)}}, \overline{q} \; \rangle)} \tag{4}$$

For fair comparison with the state-of-the-art approaches, we only store a single class-prototype per class corresponding to the original images (i.e. 0° rotation). Thus for the previous classes, only the parameters of the stochastic classifier corresponding to the 0° rotation are updated. To mitigate catastrophic forgetting, we use cross entropy loss based on the class prototypes as

$$\mathcal{L}_{proto}(q, \check{y}, \psi^{(0:t)}) = -\log(\zeta_{\check{y}r}^{(s)}(q; \psi^{(0:t)})) \tag{5}$$

where $\check{y}$ is the class label of the prototype in task s.

For the new classes, very few labeled samples per class is available. Since the few examples cannot cover the entire distribution, generalization to new classes is quite challenging. As discussed before, we propose to use stochastic classifiers which mitigates the problem of overfitting and generalizes well to the new classes even with few examples. To this end, we calculate a loss as in equation (2) on the new task data using stochastic classifiers. Finally, the total loss at each incremental task is given by

$$\mathcal{L}_{inc}^{(t)} = \lambda_1 \cdot \mathcal{L}_{proto}(q, \check{y}, \psi^{(0:t)}) + \lambda_2 \cdot \mathcal{L}_{S3C}(\mathbf{x}, y; \theta, \psi^{(0:t)}) \tag{6}$$

where $\{\mathbf{x}, y\} \in \mathcal{D}^{(t)}$ and $t > 0$. λ_1, λ_2 are hyper-parameters to balance the performance between old and new classes. At the end of task t, we have the learnt classifiers for all the classes seen so far, namely $\psi^{(0)}, \ldots, \psi^{(t)}$.

5 Testing Phase

At inference time, the test image $\mathbf{x}$ can belong to any of the classes seen so far. To utilize the learnt classifiers effectively, we generate transformed versions of $\mathbf{x}$ and aggregate all the corresponding scores. Thus, the aggregate score for the i^{th} class in task s is computed as $z_i^{(s)} = \frac{1}{M} \sum_{r=1}^{M} \eta \ \langle \ \overline{\mu_{ir}^{(s)}}, \overline{f_\theta(\widetilde{\mathbf{x}}_r)} \rangle$. Then the aggregated probability used for predicting the class is given by

$$P_{agg}(i, s/\mathbf{x}, \theta, \psi^{(0:t)}) = \frac{\exp\left(z_i^{(s)}\right)}{\sum_{j=0}^{t} \sum_{k=1}^{|C^{(j)}|} \exp\left(z_k^{(j)}\right)} \tag{7}$$

Thus, the final prediction for the test sample $\mathbf{x}$ is

$$\hat{i}, \hat{s} = arg \max_{i,s} P_{agg}(i, s/\mathbf{x}) \tag{8}$$

which implies that the input $\mathbf{x}$ belongs to $\hat{i}^{th}$ class of task $\hat{s}$. This aggregation scheme improves the model performance significantly.

6 Experimental Evaluation

Here, we describe the extensive experiments performed to evaluate the effectiveness of the proposed S3C framework. Starting with a brief introduction of the datasets, we will discuss the performance of the proposed framework on three standard benchmark datasets. In addition, we also discuss its effectiveness on two real and challenging scenarios, where (i) the data may be imbalanced at each incremental step and (ii) fewer classes may be available during base training. We also describe the ablation study to understand the usefulness of each module.

Datasets Used: To evaluate the effectiveness of the proposed S3C framework, we perform experiments on three benchmark datasets, namely CIFAR100 [18], miniImageNet [19] and CUB200 [37].
CIFAR100 [18] contains 32×32 RGB images from 100 classes, where each class contains 500 training and 100 testing images. We follow the same FSCIL dataset splits as in [44], where the base task is trained with 60 classes and the remaining 40 classes is trained in eight incremental tasks in a 5-*way* 5-*shot* setting. Thus, there are a total of 9 training sessions (i.e., base + 8 incremental).
MiniImageNet [19] is a subset of the ImageNet dataset and contains 100 classes with images of size 84×84. Each class has 600 images, 500 for training and 100 for testing. We follow the same task splits as in [44], where 60 classes are used for base task training and the remaining 40 classes are learned incrementally in 8 tasks. Each task contains 5 classes with 5 images per class.
CUB200 [37] is a fine-grained birds dataset with 200 classes. It contains a total of 6000 images for training and 6000 images for testing. All the images are resized to 256×256 and then cropped to 224×224 for training. We used the the same data splits proposed in [44], where there are 100 classes in the base task, and each of the 10 incremental tasks are learned in a 10-*way* 5-*shot* manner.

Implementation details: For fair comparison, we use the same backbone architecture as the previous FSCIL methods [44]. We use ResNet20 for CIFAR100 and ResNet18 for miniImageNet and CUB200 as in [44]. Inspired by CEC [44], we used the same GAT layer at the feature extractor output for better feature representations. We trained the base network for 200 epochs with a learning rate of 0.1 and reduced it to 0.01 and 0.001 after 120 and 160 epochs for CIFAR100 and miniImageNet datasets. For CUB200, the initial learning rate was 0.03 and was decreased to 0.003, 0.0003 after 40 and 60 epochs. We freeze the backbone network and fine-tune the stochastic classifiers for 100 epochs with a learning rate of 0.01 for CIFAR100 and miniImageNet and 0.003 for CUB200 at each incremental step. The base network was trained with a batch size of 128, and for the newer tasks, we used all the few-shot samples in a mini-batch for incremental learning. All the experiments are run on a single NVIDIA RTX A5000 GPU using PyTorch. We set $\eta = 16$, $\lambda_1 = 5$ and $\lambda_2 = 1$ for all our experiments.

Evaluation protocol: We evaluate the proposed framework using the following three evaluation metrics as followed in the FSCIL literature: (1) First, at the end of each task, we report the **Top1 accuracy** [35, 25, 44, 8] of all the classes seen so far, which is the most commonly used metric; (2) To be practically useful, the model needs to perform well on all the tasks seen so far (i.e. have a good performance balance between the previous and new tasks). To better capture this performance balance, inspired from [39], recent FSCIL works [4, 9] propose to use the **Harmonic Mean** (HM) of the performance of the previous and new classes at the end of each incremental task. If t denotes the task id, $t \in \{0, 1, ..., \mathcal{T}\}$, let Acc_n^t denote the model accuracy on test data of task n after learning task t, where $n \in \{0, 1, 2, ..., t\}$. Then at the end of task t, to analyze

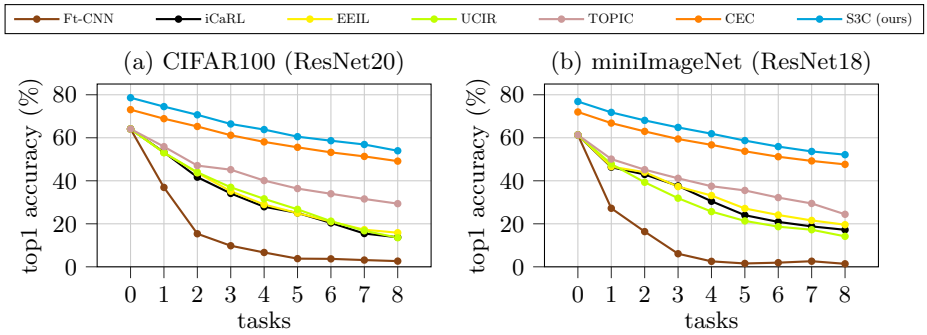


Fig. 3: Comparison of S3C with the state-of-art approaches on CIFAR100 and miniImageNet datasets using the backbone given in the caption.

Dataset	Method	Harmonic Mean (%) ↑							
		1	2	3	4	5	6	7	8
CIFAR100	CEC [44]	41.57	38.75	32.36	31.53	32.55	32.40	32.25	31.27
	S3C (Ours)	**61.60**	**54.57**	**48.94**	**47.60**	**47.00**	**46.75**	**45.96**	**45.22**
miniImageNet	CEC [44]	31.68	30.86	29.52	29.01	26.75	24.46	26.14	26.24
	S3C (Ours)	**35.30**	**38.18**	**40.62**	**38.86**	**35.02**	**34.49**	**36.06**	**36.20**

Table 1: Comparison of S3C with the state-of-the-art CEC in terms of Harmonic Mean on CIFAR100 and miniImageNet datasets. On both datasets S3C outperforms CEC by a considerable margin.

the contribution of base and novel classes in the final accuracy, harmonic mean is calculated between Acc_0^t and $Acc_{1:t}^t$. Inspired by CEC, we also report **performance dropping rate** ($PD = Acc_0^0 - Acc_{0:\mathcal{T}}^{\mathcal{T}}$) that measures the absolute difference between initial model accuracy after task 0 and model accuracy at the end of all tasks $\mathcal{T}$. Here, we report the performance of S3C framework for the standard FSCIL setting on all the three benchmark datasets. Note that all the compared approaches have used the same backbone architecture, i.e. ResNet20 for CIFAR100 and ResNet18 for miniImageNet and CUB200 datasets. As mentioned earlier, most of the FSCIL approaches like TOPIC [35], Ft-CNN [35], EEIL [7], iCaRL [31], UCIR [16], adopted this classifier as it is and proposed different techniques in the incremental stage. Thus they have the same base task accuracy as can be observed from the results. The current state-of-the-art in FSCIL, CEC [44] showed that using the same backbone along with appropriate modifications for learning the base classifier can significantly enhance not only the base class accuracies, but also the performance on the incrementally added classes. We combine the advantages of both these techniques, i.e. making the base classifier better (using the same backbone), and at the same time, effectively fine-tuning the stochastic-classifiers in S3C. Thus the base accuracy of CEC and the proposed S3C is better than the other approaches.

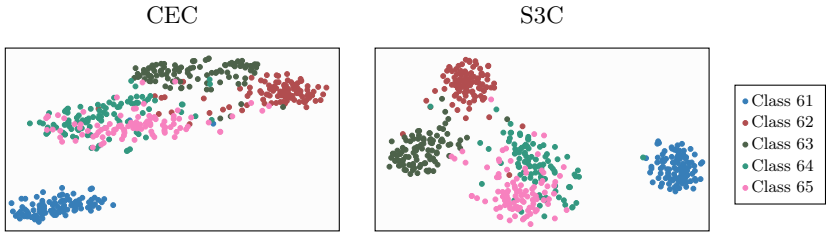


Fig. 4: Figure shows t-SNE plot (of test samples) from 5 new classes after task 1 for CIFAR100 dataset.

6.1 Results on standard FS-CIL propocol

Here, we report the results on the three benchmark datasets. Fig. 3 compares the proposed SC3 framework with the state-of-the-art approaches in terms of top1 accuracy on CIFAR100. We observe that the modifications while learning the base classifier improves the performance for both CEC and S3C significantly. At the end of all tasks, S3C achieves a top1 accuracy of 53.96% compared to 49.14% obtained by the state-of-art CEC (relative improvement is 4.82%). The performance of all the compared approaches are directly taken from [44]. Table 1 shows the HM of S3C at the end of each incremental task. We observe that S3C obtains a relative improvement of 13.95% compared to CEC in terms of HM. This shows the effectiveness of S3C in achieving a better balance between the base and new class performance. Fig. 4 shows the t-SNE plot for new classes after task 1, where we observe that the new classes in S3C are relatively well clustered compared to CEC. In terms of PD, S3C is close to CEC (higher by 0.7%), but it outperforms CEC in terms of the other two metrics, namely top1 accuracy and HM.

From Fig. 3 (right), we observe that S3C achieves 52.14% top1 accuracy on minImageNet, with a relative improvement of 4.51% over the second best of 47.63% obtained by CEC. In terms of HM (Table 1) S3C achieves 9.96% relative improvement over CEC. Performance dropping rate (PD) of CEC is slightly lower (0.35%) than S3C.

We observe from Table 2 and Table 3 that S3C outperforms CEC by 6.67% and 11.72% respectively in terms of top1 accuracy and HM for CUB200 dataset.

Method	Accuracy in each session (%) ↑											PD ↓	Our relative improvement
	0	1	2	3	4	5	6	7	8	9	10		
Ft-CNN [35]	68.68	43.7	25.05	17.72	18.08	16.95	15.1	10.6	8.93	8.93	8.47	60.21	**+39.83**
iCaRL [31]	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	47.52	**+26.69**
EEIL [7]	68.68	53.63	47.91	44.2	36.3	27.46	25.93	24.7	23.95	24.13	22.11	46.57	**+25.74**
UCIR [16]	68.68	57.12	44.21	28.78	26.71	25.66	24.62	21.52	20.12	20.06	19.87	48.81	**+27.98**
TOPIC [35]	68.68	62.79	54.81	49.99	45.25	41.4	38.35	35.36	32.22	28.31	26.28	42.40	**+21.97**
CEC [44]	75.85	71.94	68.50	63.50	62.43	58.27	57.73	55.81	54.83	53.52	52.28	23.57	**+2.74**
S3C (Ours)	**80.62**	**77.55**	**73.19**	**68.54**	**68.05**	**64.33**	**63.58**	**62.07**	**60.61**	**59.79**	**58.95**	**20.83**	

Table 2: Comparison of S3C with other approaches on CUB200 dataset. All the compared results are directly taken from [44].

Method	Harmonic Mean (%) ↑									
	1	2	3	4	5	6	7	8	9	10
CEC [44]	57.63	52.83	45.08	45.97	44.44	45.63	45.10	43.76	45.77	44.69
S3C (Ours)	**76.29**	**65.12**	**57.30**	**60.63**	**56.59**	**57.79**	**56.73**	**55.43**	**55.48**	**56.41**

Table 3: Harmonic mean comparison of S3C with CEC on CUB200 dataset.

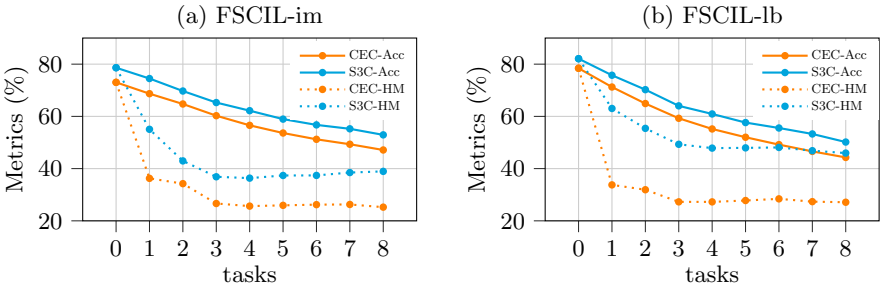


Fig. 5: Comparison of S3C and CEC in terms of top1 accuracy and harmonic mean for two challenging scenarios, namely (a) FSCIL-im and (b) FSCIL-lb.

For this dataset, the proposed S3C has the least performance dropping (PD) rate compared to all the other approaches.

6.2 Analysis and Ablation

Here, we perform additional experiments and ablation studies on the CIFAR100 dataset to evaluate the effectiveness of the proposed S3C framework.

Experiments on More Realistic and Challenging Scenarios: First, we show the effectiveness of S3C for two realistic scenarios, (i) where there is class imbalance at each incremental task; (ii) where the number of base classes is less. ***1. FSCIL-im (imbalance in new classes):*** The standard FSCIL setting assumes that equal number of images per new class are available at each incremental task. For example, 5 images for each of the 5 new classes are available at each incremental task in a 5-way 5-shot setting, In real-world, number of samples per class can vary, since for some classes, it is easier to collect data compared to others. Obviously, one can collect more samples from the minority classes, or select a sub-set from the majority classes. But it is more practical if the algorithm can satisfactorily work without this constraint.

To create the data imbalance, at each incremental step, we consider the number of training samples for the 5 new classes as $\{5, 4, 3, 2, 1\}$. Few samples along with the imbalance makes this setting very challenging. Fig. 5 (left) shows the top 1 accuracy and HM of S3C and CEC for this scenario without any modification of the algorithms. We observe that S3C performs very well for both the metrics, thus showing its effectiveness in handling imbalanced new class data.

self-supervision	classifier	After task 0 base task accuracy	After task $\mathcal{T}$ top1 accuracy	After task $\mathcal{T}$ harmonic mean
✗	linear	74.70	48.98	26.76
✓	linear	76.14	53.55	41.80
✓	stochastic	78.03	53.96	45.22

Table 4: Ablation Study: We observe that both self-supervision and stochastic classifiers help to improve the performance significantly.

2. FSCIL-lb (fewer base classes): The standard FSCIL setting assumes that the number of base classes is quite high, with many annotated samples per class. Here, we analyze the performance of S3C when the number of base classes is lower. A similar setting has been explored in [31] for CIL. The advantage of having lesser number of base classes is that the base learner becomes ready for incremental learning quickly (with fewer classes requiring many annotated samples) and the remaining classes can be learnt incrementally with fewer number of labeled samples per class. For the CIFAR100 experiments conducted so far, there were 60 base and 40 new classes. For this experiment, we use only 40 base classes, and keep the incremental tasks unchanged. From Fig. 5 (right), we observe that S3C obtains a relative improvement of 5.29% in top1 accuracy (18.83% in HM) over CEC. This shows that S3C can start learning incrementally at an early stage of data collection, which makes it more suited for real-world scenarios.

Ablation studies: Table 4 shows the effect of self-supervision and type of classifier on CIFAR100 base task accuracy. The top 1 accuracy and HM after all the incremental stages are also reported. We observe that both the modules help in improving the performance of the base and incremental classes. Though the top 1 accuracy of both linear and stochastic classifiers are close after the incremental stages, there is significant improvement in HM with the stochastic classifier. This implies that both the modules help in achieving very good performance on the new classes, in addition to retaining the performance on the base, thus achieving a great performance balance between the two.

7 Conclusions

In this paper, we proposed a novel S3C framework, which integrates self-supervision with stochastic classifiers seamlessly for the FSCIL task. We show that this framework not only reduces overfitting on the few labeled samples of the new classes, but also mitigates catastrophic forgetting of the previously learnt classes. Extensive experiments on three benchmark datasets, namely CIFAR100, CUB200 and miniImageNet and additional analysis show that the proposed S3C significantly outperforms the state-of-art approaches.

Acknowledgements: This work is partly supported through a research grant from SERB, Department of Science and Technology, Govt. of India and Google Research, India.

References

1. Akyürek, A.F., Akyürek, E., Wijaya, D., Andreas, J.: Subspace regularizers for few-shot class incremental learning. ICLR (2022)
2. Belouadah, E., Popescu, A.: Il2m: Class incremental learning with dual memory. In: ICCV. pp. 583–592 (2019)
3. Belouadah, E., Popescu, A.: Scail: Classifier weights scaling for class incremental learning. In: WACV. pp. 1266–1275 (2020)
4. Bhat, S.D., Banerjee, B., Chaudhuri, S.: Semgif: A semantics guided incremental few-shot learning framework with generative replay. In: BMVC (2021)
5. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: ICML. pp. 1613–1622 (2015)
6. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV. pp. 132–149 (2018)
7. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
8. Chen, K., Lee, C.G.: Incremental few-shot learning via vector quantization in deep embedded space. In: ICLR (2020)
9. Cheraghian, A., Rahman, S., Fang, P., Roy, S.K., Petersson, L., Harandi, M.: Semantic-aware knowledge distillation for few-shot class-incremental learning. In: CVPR. pp. 2534–2543 (2021)
10. Cheraghian, A., Rahman, S., Ramasinghe, S., Fang, P., Simon, C., Petersson, L., Harandi, M.: Synthesized feature based few-shot class-incremental learning on a mixture of subspaces. In: ICCV. pp. 8661–8670 (2021)
11. Dong, S., Hong, X., Tao, X., Chang, X., Wei, X., Gong, Y.: Few-shot class-incremental learning via relation knowledge distillation. In: AAAI. pp. 1255–1263 (2021)
12. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: ECCV 2020. pp. 86–102 (2020)
13. Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., Cord, M.: Boosting few-shot visual learning with self-supervision. In: ICCV. pp. 8059–8068 (2019)
14. Hendrycks, D., Mazeika, M., Kadavath, S., Song, D.: Using self-supervised learning can improve model robustness and uncertainty. NeurIPS **32** (2019)
15. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. NeurIPS Workshop (2014)
16. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR. pp. 831–839 (2019)
17. Komodakis, N., Gidaris, S.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NeurIPS **25**, 1097–1105 (2012)
20. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: ECCV. pp. 577–593 (2016)
21. Lee, E., Huang, C.H., Lee, C.Y.: Few-shot and continual learning with attentive independent mechanisms. In: ICCV. pp. 9455–9464 (2021)
22. Lee, H., Hwang, S.J., Shin, J.: Self-supervised label augmentation via input transformations. In: ICML. pp. 5714–5724. PMLR (2020)
23. Li, Z., Hoiem, D.: Learning without forgetting. TPAMI **40**(12), 2935–2947 (2017)

24. Lu, Z., Yang, Y., Zhu, X., Liu, C., Song, Y.Z., Xiang, T.: Stochastic classifiers for unsupervised domain adaptation. In: CVPR. pp. 9111–9120 (2020)
25. Mazumder, P., Singh, P., Rai, P.: Few-shot lifelong learning. In: AAAI (2021)
26. Neal, R.M.: Bayesian learning for neural networks, vol. 118. Springer Science & Business Media (2012)
27. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV. pp. 1520–1528 (2015)
28. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV. pp. 69–84 (2016)
29. Ouyang, W., Zeng, X., Wang, X., Qiu, S., Luo, P., Tian, Y., Li, H., Yang, S., Wang, Z., Li, H., et al.: Deepid-net: Object detection with deformable part based convolutional neural networks. TPAMI **39**(7), 1320–1334 (2016)
30. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543 (2014)
31. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 2001–2010 (2017)
32. Shi, G., Chen, J., Zhang, W., Zhan, L.M., Wu, X.M.: Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. NeurIPS **34**, 6747–6761 (2021)
33. Subedar, M., Krishnan, R., Meyer, P.L., Tickoo, O., Huang, J.: Uncertainty-aware audiovisual activity recognition using deep bayesian variational inference. In: ICCV. pp. 6301–6310 (2019)
34. Tan, Z., Ding, K., Guo, R., Liu, H.: Graph few-shot class-incremental learning. WSDM (2022)
35. Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., Gong, Y.: Few-shot class-incremental learning. In: CVPR. pp. 12183–12192 (2020)
36. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2017)
37. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011), http://www.vision.caltech.edu/visipedia/CUB-200.html
38. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
39. Xian, Y., Schiele, B., Akata, Z.: Zero-shot learning-the good, the bad and the ugly. In: CVPR. pp. 4582–4591 (2017)
40. Yang, Y., Xu, Z.: Rethinking the value of labels for improving class-imbalanced learning. NeurIPS **33**, 19290–19301 (2020)
41. Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S., Weijer, J.v.d.: Semantic drift compensation for class-incremental learning. In: CVPR. pp. 6982–6991 (2020)
42. Yu, T., Li, D., Yang, Y., Hospedales, T.M., Xiang, T.: Robust person re-identification by modelling feature uncertainty. In: ICCV. pp. 552–561 (2019)
43. Zhai, X., Oliver, A., Kolesnikov, A., Beyer, L.: S4l: Self-supervised semi-supervised learning. In: ICCV. pp. 1476–1485 (2019)
44. Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., Xu, Y.: Few-shot incremental learning with continually evolved classifiers. In: CVPR. pp. 12455–12464 (2021)
45. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.T.: Maintaining discrimination and fairness in class incremental learning. In: CVPR. pp. 13208–13217 (2020)
46. Zhu, F., Cheng, Z., Zhang, X.y., Liu, C.l.: Class-incremental learning via dual augmentation. NeurIPS **34** (2021)
47. Zhu, F., Zhang, X.Y., Wang, C., Yin, F., Liu, C.L.: Prototype augmentation and self-supervision for incremental learning. In: CVPR. pp. 5871–5880 (2021)

48. Zhu, K., Cao, Y., Zhai, W., Cheng, J., Zha, Z.J.: Self-promoted prototype refinement for few-shot class-incremental learning. In: CVPR. pp. 6801–6810 (2021)