Cross-Class Feature Augmentation for Class Incremental Learning

Taehoon Kim[1], **Jaeyoo Park**[1], **Bohyung Han**[1,2]

[1]Department of Electrical and Computer Engineering, Seoul National University
[2]Interdisciplinary Program in Artificial Intelligence, Seoul National University
{kthone, bellos1203, bhhan}@snu.ac.kr

Abstract

We propose a novel class incremental learning approach, which incorporates a feature augmentation technique motivated by adversarial attacks. We employ a classifier learned in the past to complement training examples of previous tasks. The proposed approach has an unique perspective to utilize the previous knowledge in class incremental learning since it augments features of arbitrary target classes using examples in other classes via adversarial attacks on a previously learned classifier. By allowing the Cross-Class Feature Augmentations (CCFA), each class in the old tasks conveniently populates samples in the feature space, which alleviates the collapse of the decision boundaries caused by sample deficiency for the previous tasks, especially when the number of stored exemplars is small. This idea can be easily incorporated into existing class incremental learning algorithms without any architecture modification. Extensive experiments on the standard benchmarks show that our method consistently outperforms existing class incremental learning methods by significant margins in various scenarios, especially under an environment with an extremely limited memory budget.

Introduction

Recent deep learning techniques have shown remarkable progress in various computer vision tasks including image classification (He et al. 2016; Hu, Shen, and Sun 2018), object detection (Liu et al. 2016; Redmon et al. 2016; Zhu et al. 2021c), semantic segmentation (Chen et al. 2017; Long, Shelhamer, and Darrell 2015; Noh, Hong, and Han 2015), and many others. Behind this success is an implicit assumption that the whole dataset with a predefined set of classes should be given in a batch. However, this assumption is unlikely to hold in the real-world scenarios which change dynamically over time. This limits the applicability to real-world problems because deep neural networks trained under changing data distribution often suffer from catastrophic forgetting, meaning that the models lose the ability to maintain knowledge about old tasks. While a straightforward way to handle the critical challenge is to retraining the model with an integrated dataset, this is too expensive or even impossible due to the limitation of computational resources and the inaccessibility of training data.

Class incremental learning is a framework that progressively increases the number of classes while combating the catastrophic forgetting issue. Among many existing approaches (Aljundi et al. 2018; Kirkpatrick et al. 2017; Li and Hoiem 2017; Rebuffi et al. 2017; Wu et al. 2019; Zenke, Poole, and Ganguli 2017), the techniques based on knowledge distillation with exemplars (Douillard et al. 2020; Hou et al. 2019; Li and Hoiem 2017), allowing new models to mimic previous ones, have demonstrated promising performance in alleviating the feature drift issue. Yet, these methods still suffer from data deficiency for old tasks and data imbalance between tasks, as only few training examples are available for the previous tasks. To alleviate these limitations, some existing approaches generate either data samples (Ostapenko et al. 2019; Shin et al. 2017) or feature representations (Liu et al. 2020a). However, they require additional generative models, hampering the stability of convergence and increases the complexity of models.

This paper presents a novel feature augmentation technique, referred to as Cross-Class Feature Augmentation (CCFA), which effectively tackles the aforementioned limitations in class incremental learning. By leveraging the representations learned in the past, we aim to augment the features at each incremental stage to address data deficiency in the classes belonging to old tasks. To this end, inspired by adversarial attacks, we adjust the feature representations of training examples to resemble representations from specific target classes that are different from their original classes. These perturbed features allow a new classifier to maintain the decision boundaries for the classes learned up to the previous stages. Note that this is a novel perspective different from conventional adversarial attack methods (Carlini and Wagner 2017; Goodfellow, Shlens, and Szegedy 2017; Madry et al. 2018; Moosavi-Dezfooli, Fawzi, and Frossard 2016; Zhao, Dua, and Singh 2018), which focus on deceiving models. One may consider generating additional features for each class using the exemplars with the same class labels. However, this strategy is prone to generate redundant or less effective features for defending class boundaries, especially when the number of exemplars in each class is small. On the contrary, the proposed approach exploits exemplars in various classes observed in the previous tasks as well as a large number of training data in the current task, which is helpful for synthesizing features with heterogeneous properties.

The contributions of this paper are summarized below:

- We propose a novel class incremental learning technique, which effectively increases training examples for old tasks via feature augmentation. The proposed method prevents catastrophic forgetting without modifying architectures or introducing generative models.
- Cross-Class Feature Augmentation (CCFA) synthesizes augmented features across class labels using a concrete objective motivated by adversarial attacks.
- Our algorithm is easily incorporated into existing class incremental learning methods and improves performance consistently on multiple datasets with diverse scenarios, especially under minimal memory budgets.

Related Works

Class Incremental Learning

Most of the existing class incremental learning algorithms address the catastrophic forgetting issue using the following techniques: 1) parameter regularization, 2) architecture expansion, 3) bias correction, 4) knowledge distillation, and 5) rehearsal.

Parameter regularization The methods that belong to this category (Aljundi et al. 2018; Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017) measure the importance of each parameter and determine its flexibility based on its importance. The popular metrics to determine the plasticity of models include the Fisher information matrix (Kirkpatrick et al. 2017), the path integral along parameter trajectory (Zenke, Poole, and Ganguli 2017), and the output vector changes (Aljundi et al. 2018). However, their empirical performances are not satisfactory in class incremental learning scenarios (Hsu et al. 2018; van de Ven and Tolias 2019).

Architecture expansion Architectural methods (Rusu et al. 2016; Yoon et al. 2018) typically focus on expanding network capacity dynamically to handle a sequence of incoming tasks. To this end, (Yan, Xie, and He 2021) proposes to expand the network for each incoming task while compressing the network after the end of each task for computational efficiency. Recently, (Liu, Schiele, and Sun 2021) adopts two network blocks to balance plasticity and stability, which are optimized by a bi-level optimization. Although these approaches show remarkable performances even with a long sequence of tasks, their computational complexity at both training and inference time increases linearly as the number of tasks grows, which leads to a significant computational burden and challenges in inference.

Bias correction There exist several approaches (Hou et al. 2019; Wu et al. 2019) that tackle the bias towards new classes incurred by class imbalance. To be specific, (Wu et al. 2019) reduces the bias by introducing additional scale and shift parameters for an affine transformation of the logits for new classes. Moreover, (Zhao et al. 2020) matches the scale of the weight vectors for the new classes with the average norm of the old weight vectors.

Knowledge distillation The methods based on knowledge distillation (Hinton, Vinyals, and Dean 2015; Romero et al. 2015; Zagoruyko and Komodakis 2017; Park, Kang, and Han 2021; Kang, Park, and Han 2022) aim to encourage a model to learn new tasks while mimicking the representations of the old model trained for the previous tasks. To this end, (Li and Hoiem 2017; Rebuffi et al. 2017; Wu et al. 2019; Castro et al. 2018; Hou et al. 2019) match their outputs after the classification layers with old models to preserve the representations of input examples. PODNet (Douillard et al. 2020) controls the balance between the previous knowledge and the new information by preserving the relaxed representations. Here, relaxed representations are obtained by applying the sum pooling along the width and height dimensions to the original intermediate feature maps. Recently, (Simon, Koniusz, and Harandi 2021) proposes to optimize the model considering the concept of the geodesic flow and AFC (Kang, Park, and Han 2022) introduces the way to handle catastrophic forgetting by minimizing the upper bound of the loss increases caused by the representation change.

Rehearsal Rehearsal-based methods utilize a limited number of representative examples stored from old tasks or replay old examples using generative models while training new tasks. Incremental Classifier Representation Learning (iCaRL) (Rebuffi et al. 2017) keeps a small number of samples per class to approximate the class centroid and makes predictions based on the nearest class mean classifiers. On the other hand, pseudo-rehearsal techniques (Ostapenko et al. 2019; Shin et al. 2017) generate examples in the previously observed classes using generative adversarial networks (GANs) (Goodfellow et al. 2014; Liu et al. 2020a; Odena, Olah, and Shlens 2017). However, these methods need to train and store generative models, which incurs an extra burden for class incremental learning.

Adversarial attacks One can mislead deep neural networks trained on natural datasets by injecting minor perturbation into input data, which is called an adversarial attack (Carlini and Wagner 2017; Goodfellow, Shlens, and Szegedy 2017; Madry et al. 2018; Moosavi-Dezfooli, Fawzi, and Frossard 2016; Zhao, Dua, and Singh 2018). A well-known category in adversarial attacks relies on gradient-based optimization (Carlini and Wagner 2017; Goodfellow, Shlens, and Szegedy 2017; Madry et al. 2018). These methods deceive a network by adding a small amount of noise imperceptible by humans, in the direction of increasing the loss corresponding to the ground-truth label.

Proposed Approach

Problem Setup

Class incremental learning trains a model in an online manner given a sequence of tasks, which is denoted by $T_{1:K} \equiv \{T_1, \cdots, T_k, \cdots, T_K\}$. Each task T_k is defined by a training dataset $\mathcal{D}_k$ composed of examples with labels $y \in \mathcal{Y}_k$, where $(\mathcal{Y}_1 \cup \cdots \cup \mathcal{Y}_{k-1}) \cap \mathcal{Y}_k = \emptyset$. At the k^{th} incremental stage, the model is trained on $\mathcal{D}'_k = \mathcal{D}_k \cup \mathcal{M}_{k-1}$, where $\mathcal{M}_{k-1}$ is a small subset of all previously seen training datasets, which is called a memory buffer. The performance of the trained

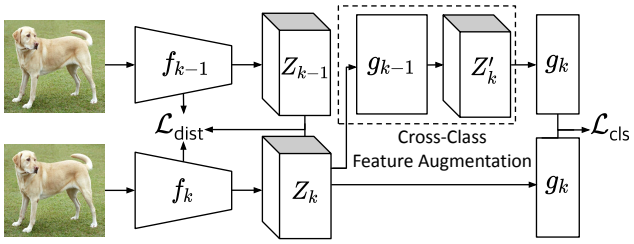

Figure 1: Overall class incremental learning framework with the proposed Cross-Class Feature Augmentation (CCFA). Our model minimizes classification loss $\mathcal{L}_{\text{cls}}$ on training examples in a mini-batch sampled from the union of current task dataset and a small set of exemplars from the previous tasks while minimizing the distillation loss $\mathcal{L}_{\text{dist}}$. To deal with the catastrophic forgetting issue induced by data imbalance between the previous and current tasks, we employ the CCFA to generate diverse features supporting the decision boundaries of the old classifier $g_{k-1}(\cdot)$.

model is evaluated on the test data sampled from a collection of all the encountered tasks without task boundaries.

Overall Framework

We incorporate our feature augmentation technique into existing class incremental learning framework based on knowledge distillation (Douillard et al. 2020; Rebuffi et al. 2017; Kang, Park, and Han 2022), and Figure 1 illustrates the concept of our approach. At the k^{th} incremental stage, we train the current model parametrized by Θ_k, which consists of a feature extractor $f_k(\cdot)$ and a classifier $g_k(\cdot)$, using $\mathcal{D}'_k$. The model optimizes Θ_k on a new task, T_k, initialized by the model parameters in the previous stage, Θ_{k-1}, while preserving the information learned from old tasks, $T_{1:k-1}$, by using knowledge distillation. To further enhance the generalization ability on the previously learned classes, we introduce Cross-Class Feature Augmentation (CCFA), which will be further discussed in the next subsection. After each incremental stage, we sample exemplars from $\mathcal{D}_k$ by a herding strategy (Rebuffi et al. 2017) and augment the memory buffer from $\mathcal{M}_{k-1}$ to $\mathcal{M}_k$.

Cross-Class Feature Augmentation (CCFA)

Using a set of features extracted from the available training data, CCFA conducts feature augmentation to supplement training examples for the previous tasks. The feature augmentation is done in a similar way as adversarial attacks and does not require introducing and training a generator. For CCFA, we utilize the classifier learned in the previous stage, which is readily accessible. The main idea of CCFA is to perturb a feature representation in a direction such that the perturbed feature *crosses* the decision boundary in the previous classifier $g_{k-1}(\cdot)$ to the target class. Here, target classes are different from the original ones, as depicted in Figure 2.

To be specific, for input images and the corresponding labels in a mini-batch $(X, Y) \sim \mathcal{D}'_k$, we first extract normalized feature vectors $Z_k = f_k(X)$. Then, we compute confidences of individual in-batch examples with respect to the

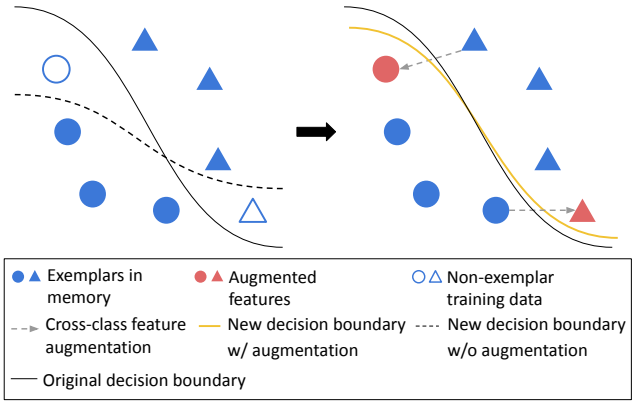

Figure 2: Illustration of Cross-Class Feature Augmentation (CCFA). CCFA perturbs a feature representation in a direction such that the perturbed feature crosses the decision boundary in the previous classifier to the target class, which is different from the original class, and complements the features for the target classes learned up to the previous stages.

class labels appearing in previous stages and define a collective confidence matrix $W \in \mathbb{R}^{b \times c_{\text{old}}}$ as follows:

$$W_{ij} = \begin{cases} 0, & \text{if } Y_i = j \\ W'_{ij}, & \text{if } Y_i \neq j, \end{cases} \tag{1}$$

where b is the batch size, $c_{\text{old}} = |\mathcal{Y}_{1:k-1}|$ is the number of classes in previous stage, and $W' = g_k(Z_k)$ is the output of current classifier. Note that the entries for the ground-truth classes are set to zeros.

We formulate an optimization problem—selecting the target classes as evenly distributed as possible while maximizing the confidence scores—which is given by

$$\begin{aligned} &\max_S \sum_{j=1}^{c_{\text{old}}} \sum_{i=1}^{b} W_{ij} S_{ij} - \sum_{j=1}^{c_{\text{old}}} \Big| \sum_{i=1}^{b} S_{ij} - u \Big| \\ &\text{such that} \quad S \in \{0,1\}^{b \times c_{\text{old}}}, \ \sum_{j=1}^{c_{\text{old}}} S_{ij} = 1 \ \ \forall\, i, \end{aligned} \tag{2}$$

where $u = \frac{b}{c_{\text{old}}}$ is a constant. The solution of Eq. (2), a matrix consisting of b one-hot row vectors, provides a collection of target classes, Y_{target}.

Due to the computational complexity, we leverage a continuous relaxation of S and reduce the number of active variables by selecting the top-K classes with the highest confidence in each row of W. Note that the entries in S_i except the ones corresponding to the top-K classes are simply set to zeros and target classes are to be sampled from the matrix S whose rows act as sampling distributions.

Once the target classes are determined, we apply the Projected Gradient Descent (PGD) (Madry et al. 2018) algorithm iteratively using the following equation:

$$Z_k^{n+1} = Z_k^n - \alpha \operatorname{sign}(\nabla_{Z_k^n} \ell_{\text{cls}}(g_{k-1}(Z_k^n), Y_{\text{target}})), \tag{3}$$

where $Z_k^0 = Z_k$, $\ell_{\text{cls}}(\cdot,\cdot)$ is the algorithm-specific classification loss function of each baseline, and α denotes the step

size for the adversarial attack. Note that we attack the normalized features just before the classification layer, unlike the original adversarial attack operating on the image space.

We obtain an adjusted features Z'_k after performing the above process for N steps. By passing Z'_k through the old classifier $g_{k-1}(\cdot)$, we obtain pseudo-labels Y_{pseudo} for Z'_k as

$$Y_{\text{pseudo}} = \arg\max g_{k-1}(Z'_k). \tag{4}$$

We train the model with the pseudo-labeled augmented features in addition to the original ones, where the classification loss is given by

$$\mathcal{L}_{\text{cls}} = \ell_{\text{cls}}([g_k(Z_k), g_k(Z'_k)], [Y, Y_{\text{pseudo}}]), \tag{5}$$

where $[\cdot, \cdot]$ denotes the concatenation operator along the batch dimension.

The unique aspect of CCFA is that it augments features across class boundaries. Feature augmentation within the same classes could be another reasonable option to approximate the distribution of the old training dataset. However, according to our experiments, this strategy suffers from the lack of diversity in the source of augmentation. On the other hand, CCFA generates heterogeneous features from diverse sources in other classes, which handle sample deficiency in the previous tasks effectively.

Training Objective

The proposed approach is generic and can be integrated into various class incremental learning algorithms. In this paper, we are interested in the methods based on knowledge distillation and the loss function at each stage is given by

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{cls}} + \lambda \cdot \mathcal{L}_{\text{dist}}, \tag{6}$$

where $\mathcal{L}_{\text{dist}}$ is the distillation loss for each baseline algorithm, and λ is the balancing weight. The augmented features given by CCFA only affect $\mathcal{L}_{\text{cls}}$, not $\mathcal{L}_{\text{dist}}$.

Experiments

Datasets and Evaluation Protocol

We evaluate the proposed method on two datasets for class incremental learning, CIFAR-100 (Krizhevsky, Nair, and Hinton 2009), and ImageNet-100/1000 (Russakovsky et al. 2015). We arrange the classes in three different orders for CIFAR-100 and in a single order for ImageNet-100 and ImageNet-1000 as provided by (Douillard et al. 2020). Following the previous works (Douillard et al. 2020; Hou et al. 2019; Liu, Schiele, and Sun 2021), we train the model using a half of the classes in the initial stage and split the remaining classes into groups of 50, 25, 10 and 5 for CIFAR-100, and 10 and 5 for ImageNet-100/1000. At each incremental stage, we evaluate the model with the test examples in all the encountered classes. We report the average of the accuracies aggregated from all the incremental stages, which is also referred to as *average incremental accuracy* (Douillard et al. 2020; Hou et al. 2019; Rebuffi et al. 2017).

Number of tasks	50	25	10	5
iCaRL	44.20	50.60	53.78	58.08
BiC	47.09	48.96	53.21	56.86
Mnemonics	-	60.96	62.28	63.34
GDumb*	59.76	59.97	60.24	60.70
TPCIL	-	-	63.58	65.34
GeoDL*	52.28	60.21	63.61	65.34
UCIR	49.30	57.57	61.22	64.01
PODNet	57.98	60.72	63.19	64.83
PODNet*	57.84	60.50	62.77	64.62
PODNet* + CCFA	**60.69**	**62.91**	**65.50**	**67.24**
AANet	-	62.31	64.31	66.31
AANet*	60.91	62.34	64.49	66.34
AANet* + CCFA	**62.20**	**63.74**	**66.16**	**67.37**
AFC	62.18	64.06	64.29	65.82
AFC*	61.74	63.78	64.63	66.02
AFC* + CCFA	**63.11**	**64.59**	**65.61**	**66.47**

Table 1: Class incremental learning performance on CIFAR-100. CCFA consistently improves the performance of the existing methods. Models with $*$ are our reproduced results. Note that we run 3 experiments with 3 different orders for CIFAR-100 and report the average performance. The boldfaced numbers indicate the best performance.

Implementation Details

We follow the implementation settings of the existing methods (Douillard et al. 2020; Liu, Schiele, and Sun 2021; Kang, Park, and Han 2022). We adopt ResNet-32 for CIFAR-100 and ResNet-18 for ImageNet as the backbone network architectures. The hyperparameters including learning rates, batch sizes, training epochs, distillation loss weight (λ) and herding strategies are identical to the baseline algorithms (Douillard et al. 2020; Liu, Schiele, and Sun 2021; Kang, Park, and Han 2022). The size of the memory buffer is set to 20 per class as a default for all experiments unless specified otherwise.

For the feature augmentation, we set the number of iterations for adversarial attack to 10 for all experiments. Empirically, we find that $K = 1$ is sufficient for top-K in target selection which is equivalent to selecting classes with highest confidence as Y_{target} without solving LP optimization problem. In the CIFAR-100 experiments, we randomly sample the attack step size α from a uniform distribution $\mathcal{U}(\frac{2}{255}, \frac{5}{255})$ and generate 640 features, which is 5 times the batch size, using 5 randomly sampled values of α for each real example. For ImageNet, the attack step size α is randomly sampled from the uniform distribution $\mathcal{U}(\frac{2}{2040}, \frac{5}{2040})$ and 128 features, which is equal to the batch size, are generated using a random sample of α. We select the step size α in a way that the same amount of feature updates occurs in a single iteration for backbone networks with different feature dimensions. For the number of augmented features, we consider the stability-plasticity trade-off.

Results on CIFAR-100

We compare the proposed approach, referred to as Cross-Class Feature Augmentation (CCFA), with existing state-of-the-art class incremental learning methods. We incorporate CCFA into four baseline models including PODNet (Douil-

Dataset	Number of tasks	Memory per class (m)	m=1	m=5	m=**20**
ImageNet-100	5	PODNet	—	—	75.54
		PODNet*	50.18	67.03	74.47
		PODNet* + CCFA	**64.28**	**72.42**	**75.78**
		AANet	—	—	76.96
		AANet*	71.25	75.22	78.05
		AANet* + CCFA	**74.48**	**76.75**	**78.14**
		AFC	–	–	76.87
		AFC*	56.53	72.75	**76.91**
		AFC* + CCFA	**64.45**	**74.35**	76.75
	10	PODNet	—	—	74.33
		PODNet*	37.64	63.21	72.37
		PODNet* + CCFA	**49.80**	**65.62**	**73.00**
		AANet	—	—	74.33
		AANet*	60.74	71.21	75.90
		AANet* + CCFA	**66.62**	**73.51**	**76.71**
		AFC	–	–	75.75
		AFC*	51.37	70.69	75.10
		AFC* + CCFA	**61.85**	**71.35**	**75.39**
ImageNet-1000	5	PODNet	—	—	66.95
		PODNet*	51.20	66.08	69.70
		PODNet* + CCFA	**59.78**	**69.03**	**69.97**
		AFC	–	–	67.02
		AFC*	57.37	66.47	67.93
		AFC* + CCFA	**65.11**	**68.78**	**69.55**
	10	PODNet	—	—	64.13
		PODNet*	36.66	59.34	67.11
		PODNet* + CCFA	**48.41**	**63.74**	**67.82**
		AFC	–	–	67.02
		AFC*	52.36	63.62	65.96
		AFC*+ CCFA	**63.42**	**67.07**	**67.88**

Table 2: Class incremental learning performance on ImageNet-100/1000 with varying memory sizes. CCFA consistently improves the performance when plugged into the baselines, especially with an extremely limited memory budget. Models with an asterisk ($*$) are our reproduced results. The bold-faced numbers indicate the best performance.

lard et al. 2020)[1], AANet (Liu, Schiele, and Sun 2021)[2], and AFC (Kang, Park, and Han 2022)[3]. Note that we run three experiments with three different orders for CIFAR-100 and report the average performance. Table 1 demonstrates that the proposed method consistently improves accuracy on three baseline models in various scenarios, and achieves the state-of-the-art performance.

Results on ImageNet

We test the performance of CCFA on ImageNet-100/1000 with varying size of the memory buffer. Table 2 presents the results of our method compared to the baselines, PODNet (Douillard et al. 2020), AANet (Liu, Schiele, and Sun 2021), and AFC (Kang, Park, and Han 2022) by varying the number of tasks and the allocated memory per task. CCFA boosts the performances consistently, especially when the size of the memory buffer is extremely small. These results show that the features augmented by CCFA play a crucial role in complementing the lack of training examples in old tasks and alleviating the collapse of the decision boundaries. Moreover, CCFA is helpful for regularizing the model by augmenting diverse samples and reducing overfitting issue commonly observed when the memory size is small.

Ablation Study and Analysis

We perform several ablation studies on CIFAR-100 to analyze the effectiveness of CCFA. For all the ablation studies, we utilize PODNet as the baseline. The number of incremental stages is set to 50 unless specified otherwise.

Initialization for augmentation Table 3(a) presents the comparison between random noise and extracted feature, z_k, as an initialization point for perturbation. To implement the random initialization, we replace Z_k in Equation (3) by a sample from a Gaussian distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. According to our experiment, the augmented feature from Z_k outperforms the random noise while the random initilization is still helpful for boosting performance.

Target class selection strategy We evaluate the proposed cross-class feature augmentation with different target selection criterion. Table 3(b) demonstrates that CCFA boosts the baseline even with the random target class selection strategy since random target classes provide diverse augmentation directions. However, the gradients toward low-confidence classes may be unstable, reducing the benefit of CCFA. Performance drop with the farthest target class supports this

[1]https://github.com/arthurdouillard/incremental_learning.pytorch
[2]https://github.com/yaoyao-liu/class-incremental-learning
[3]https://github.com/kminsoo/AFC

Ablation types	Variations	Acc (%)
(a) Initialization	Random noise	59.90
	Z_k (ours)	**60.69**
(b) Target class	Random	59.91
	Farthest	57.38
	Nearest (ours)	**60.69**
(c) Top-K (5 stages)	$K = 10$	**67.56**
	$K = 5$	67.39
	$K = 3$	67.38
	$K = 1$ (ours)	67.24
(d) Augmentation methods	PODNet	57.98
	CutMix	58.92
	MixUp	55.96
	Manifold-MixUp	55.14
	CCFA	**60.69**
(e) Exemplar-free	IL2A	58.42
	IL2A + CCFA	**60.78**

Table 3: Ablation study results on the variations of CCFA.

Number of tasks	50	25	10	5
PODNet + Mixup	55.96	59.22	63.97	66.26
PODNet + Mixup+ CCFA	**58.47**	**64.05**	**67.77**	**69.19**
PODNet + CutMix	58.92	62.69	66.18	68.30
PODNet + CutMix+ CCFA	**61.41**	**64.88**	**67.81**	**69.28**

Table 4: Compatibility of CCFA with existing data augmentation techniques.

assumption. In addition, we show the results with varying K for top-K used in target selection process. Table 3(c) illustrates that setting $K = 1$, *i.e.*, no optimization process added, shows comparable result with the results on larger K's. While increased K shows improved results, introducing LP in batch training incurs excessive extra training costs. On a single NVIDIA Titan Xp GPU, training with $K = 1$ runs at 5.15 iteration/sec, which is 10 times faster than the case with $K = 3$, when the ResNet-32 backbone is employed with batch size of 128.

Comparison with other data augmentation techniques Data augmentation is a widely used method to increase the number of training examples and learn a robust model. We show the superiority of CCFA to the standard data augmentation methods. We employ CutMix (Yun et al. 2019), Mixup (Zhang et al. 2018) and Manifold-MixUp (Verma et al. 2019). Table 3(d) exhibits that CCFA clearly outperforms the existing augmentation methods for class incremental learning.

Results on exemplar-free setting Table 3(e) presents the results of IL2A (Zhu et al. 2021a) on CIFAR-100 with 10 incremental stages. IL2A (Zhu et al. 2021a) is an exemplar-free method that employ two different augmentation strategies, class and semantic augmentations. Table 3(e) shows that CCFA boosts the performance of the baseline in exemplar-free settings. Note that semantic augmentation of IL2A (Zhu et al. 2021a) samples features from a class-wise Gaussian distribution as in PASS (Zhu et al. 2021b).

Compatibility with the existing augmentation methods We investigate the compatibility of CCFA with the existing augmentation methods, Mixup (Zhang et al. 2018) and CutMix (Yun et al. 2019), to show broad applicability of CCFA. Table 4 shows that CCFA benefits from the existing input-level augmentation techniques in every class incremental learning scenario.

# of augmented features	1	3	5	7	9
Forgetting ↓	28.13	26.19	24.97	25.00	23.85
Average new accuracy ↑	76.58	72.11	68.94	66.63	65.15
Overall accuracy	59.14	60.17	**60.69**	60.51	60.51

Table 5: Forgetting vs adaptivity by varying the number of augmented features (Z'_k).

Memory per class (m)	m=5	m=10	m=20	m=50
iCaRL	16.44	28.57	44.20	48.29
BiC	20.84	21.97	47.09	55.01
UCIR	22.17	42.70	49.30	57.02
PODNet	35.59	48.54	57.98	63.69
PODNet + CCFA (ours)	**39.70**	**52.25**	**60.69**	**65.99**

Table 6: Comparative results by varying the memory budget for each class on CIFAR-100 with 50 stages.

Initial task size	20	30	40	50
Number of stages	80	70	60	50
iCaRL	41.28	43.38	44.35	44.20
BiC	40.95	42.27	45.18	47.09
UCIR	41.69	47.85	47.51	49.30
PODNet	47.68	52.88	55.42	57.98
PODNet + CCFA (ours)	**50.32**	**55.51**	**57.59**	**60.69**

Table 7: Performance comparison between the proposed CCFA and the state-of-the-art frameworks on CIFAR-100 by varying the number of classes in the initial task while each of the remaining tasks only contains a single class. The bold-faced numbers represent the best performance.

Number of augmented features We conduct experiments by varying the number of augmented features. Since the quantity of the features affects the relative data ratio between the old and new tasks, we focus on the balance between learning new classes and forgetting old classes. We measure adaptivity of the model to new classes using the average new accuracy. For robustness of the model to old classes, we measure the forgetting metric (Lee et al. 2019). Table 5 illustrates the performance of PODNet with CCFA in terms of the two metrics and the overall accuracy by varying the number of augmented features. We observe that increasing the number of augmented features tends to reduce both forgetting and average new accuracy since they impose more weight on old classes.

Memory size We claim that the proposed method effectively compensate for the lack of training data at the feature space level. To validate this hypothesis, we analyze how the size of the memory budget affect the performance of our algorithm in Table 6. The results show that the accuracy gains obtained from CCFA indeed increase by decreasing the memory size, which is consistent with our assumption.

Memory per class (m)	m=1	m=5	m=10	m=20
PODNet	35.18	55.30	59.16	63.19
PODNet + CCFA towards GT	30.07	57.13	62.09	64.84
PODNet + CCFA (ours)	**38.06**	**58.78**	**62.85**	**65.50**

Table 8: Analysis regarding the direction of the augmentation process on CIFAR-100 with 10 stages. The results demonstrate effectiveness of setting the target class in a cross-class manner with various memory budgets. The boldfaced numbers represent the best performance.

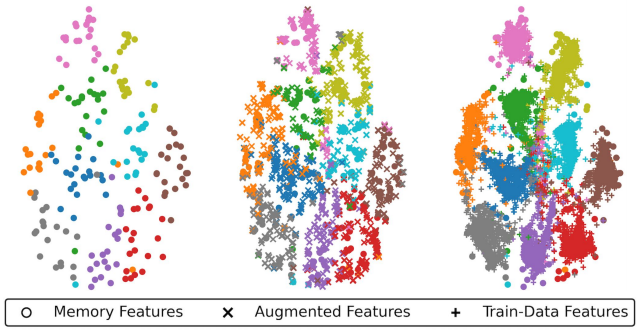

Figure 3: t-SNE of the features from the memory buffer of the random 10 classes after training initial stage (Memory Features), features generated by CCFA (Augmented Features) and features from randomly selected 1000 images from training dataset (Train-Data Features). By allowing CCFA, each class in the old tasks populates samples in the feature space, which alleviates the collapse of the decision boundaries caused by sample deficiency for the old tasks.

Initial task size We evaluate CCFA in the settings, where the initial task is small and the initially learned features are not sufficiently robust. Table 7 illustrates the results with respect to diverse initial task sizes; our approach outperforms the baselines regardless of initial task sizes.

Cross-class augmentation strategy As discussed earlier, one reasonable augmentation strategy is to adopt the gradient direction that preserves the original class label and reduces the classification loss. We compare CCFA with CCFA towards the ground-truth labels (CCFA towards GT) to demonstrate the efficiency of the proposed algorithm with various memory budgets in Table 8. Note that CCFA outperforms CCFA towards GT in all settings and their performance gap increases as the memory size gets smaller. This result implies that the cross-class augmentation strategy in CCFA yields more diversity in the synthesized representations and alleviates the lack of augmentation sources in a limited memory environment.

Computational complexity CCFA incurs a small amount of additional computation because the gradients are computed with respect to the classification layer only. On a single NVIDIA RTX GPU, PODNet with CCFA requires 1.53 seconds per iteration while PODNet requires 1.50s with batch size 128 under the ResNet-18 backbone.

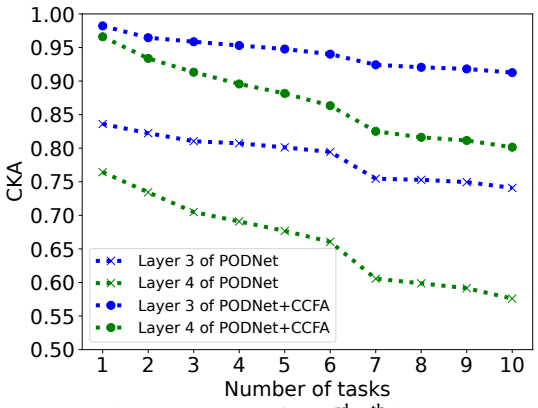

Figure 4: CKA after 3rd/4th layer

Figure 5: CKA between the features from the feature extractors of the first and each incremental stage for PODNet and PODNet + CCFA after 3rd/4th residual layer.

Visualization of CCFA

To support our argument that augmented features given by CCFA resolve the sample deficiency problem of class incremental learning, we visualize the features from the memory buffer, augmented features, and features from training data that are inaccessible in subsequent stages. Figure 3 demonstrates that CCFA effectively approximates the structure of feature space given by full training data, which is consistent with our quantitative results..

Effects of CCFA on the Feature Extractor

To understand the impact of CCFA on the feature extractor, we measure the Centered Kernel Alignment (CKA) (Cortes, Mohri, and Rostamizadeh 2012; Kornblith et al. 2019; Kim and Han 2023) between the intermediate features of the feature extractor after the first incremental stage and each sequential incremental stage, with and without CCFA. We utilize the test set of the first stage to measure the drift of the feature representations learned in the first incremental stage. Figure 5 illustrates the change of CKA values as the number of stages increases. The results clearly demonstrate that CCFA alleviates forgetting at intermediate feature level even though it only operates at the classifier level.

Conclusion

We presented a novel feature augmentation technique for class incremental learning. The proposed method augments the features of the target class by utilizing the samples from other classes, adversarially attacking the previously learned classifier. The augmented features play a role in complementing the data for the previous tasks. This idea is generally applicable to existing frameworks without any modification on the architecture. Our extensive experimental results demonstrate that the proposed method consistently improves performance on multiple datasets when applied to existing class incremental learning frameworks, especially in an environment with extremely limited memory constraints.

Acknowledgements

This work was partly supported by Samsung Advanced Institute of Technology, and the National Research Foundation of Korea grant [No. 2022R1A2C3012210, Knowledge Composition via Task-Distributed Federated Learning] and the IITP grants [No.2022-0-00959, (Part 2) Few-Shot Learning of Causal Inference in Vision and Language for Decision Making; 2021-0-02068, Artificial Intelligence Graduate School Program (Seoul National University); 2021-0-01343, Artificial Intelligence Innovation Hub] funded by the Korean government (MSIT).

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory Aware Synapses: Learning what (not) to forget. In *ECCV*.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-End Incremental Learning. In *ECCV*.

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2017. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *TPAMI*.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2012. Algorithms for learning kernels based on centered alignment. In *JMLR*.

Douillard, A.; Cord, M.; Ollion, C.; and Robert, T. 2020. PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning. In *ECCV*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *NIPS*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2017. Explaining and Harnessing Adversarial Examples. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a Unified Classifier Incrementally via Rebalancing. In *CVPR*.

Hsu, Y.-C.; Liu, Y.-C.; Ramasamy, A.; and Kira, Z. 2018. Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. *arXiv preprint arXiv:1810.12488*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In *CVPR*.

Kang, M.; Park, J.; and Han, B. 2022. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*.

Kim, D.; and Han, B. 2023. On the Stability-Plasticity Dilemma of Class-Incremental Learning. In *CVPR*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the national academy of sciences*.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *ICML*.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2009. Learning Multiple Layers of Features from Tiny Images. Technical report.

Lee, K.; Lee, K.; Shin, J.; and Lee, H. 2019. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 312–321.

Li, Z.; and Hoiem, D. 2017. Learning without Forgetting. *TPAMI*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. SSD: Single Shot MultiBox Detector. In *ECCV*.

Liu, X.; Wu, C.; Menta, M.; Herranz, L.; Raducanu, B.; Bagdanov, A. D.; Jui, S.; and de Weijer, J. v. 2020a. Generative Feature Replay for Class-Incremental Learning. In *CVPR Workshops*.

Liu, Y.; Schiele, B.; and Sun, Q. 2021. Adaptive Aggregation Networks for Class-Incremental Learning. In *CVPR*.

Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B.; and Sun, Q. 2020b. Mnemonics Training: Multi-Class Incremental Learning without Forgetting. In *CVPR*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. DeepFool: a Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*.

Noh, H.; Hong, S.; and Han, B. 2015. Learning Deconvolution Network for Semantic Segmentation. In *ICLR*.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional Image Synthesis with Auxiliary Classifier Gans. In *ICML*.

Ostapenko, O.; Puscas, M.; Klein, T.; Jahnichen, P.; and Nabi, M. 2019. Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning. In *CVPR*.

Park, J.; Kang, M.; and Han, B. 2021. Class-incremental learning for action recognition in videos. In *ICCV*.

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *ECCV*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. FitNets: Hints For Thin Deep Nets. In *ICLR*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual Learning with Deep Generative Replay. In *NIPS*.

Simon, C.; Koniusz, P.; and Harandi, M. 2021. On Learning the Geodesic Path for Incremental Learning. In *CVPR*.

Tao, X.; Chang, X.; Hong, X.; Wei, X.; and Gong, Y. 2020. Topology-Preserving Class-Incremental Learning. In *ECCV*.

van de Ven, G. M.; and Tolias, A. S. 2019. Three Scenarios for Continual Learning. *arXiv preprint arXiv:1904.07734*.

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold mixup: Better representations by interpolating hidden states. In *ICML*.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large Scale Incremental Learning. In *CVPR*.

Yan, S.; Xie, J.; and He, X. 2021. DER: Dynamically Expandable Representation for Class Incremental Learning. In *CVPR*.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2018. Lifelong Learning with Dynamically Expandable Networks. In *ICLR*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*.

Zagoruyko, S.; and Komodakis, N. 2017. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. In *ICML*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond empirical risk minimization. *ICLR*.

Zhao, B.; Xiao, X.; Gan, G.; Zhang, B.; and Xia, S.-T. 2020. Maintaining Discrimination and Fairness in Class incremental Learning. In *CVPR*.

Zhao, Z.; Dua, D.; and Singh, S. 2018. Generating Natural Adversarial Examples. In *ICLR*.

Zhu, F.; Cheng, Z.; Zhang, X.-Y.; and Liu, C.-l. 2021a. Class-incremental learning via dual augmentation. In *NeurIPS*.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F.; and Liu, C.-L. 2021b. Prototype Augmentation and Self-Supervision for Incremental Learning. In *CVPR*.

Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2021c. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR*.